BARCELONA SUPERCOMPUTING CENTER
&
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Hybrid CPU/GPU Implementation of the FE2 Multi-Scale Method for Composite Problems

by

Guido Giuntoli

Supervisors:
Dr. Mariano Vázquez
Dr. Sergio Oller

A thesis submitted for the degree of Doctor of Philosophy in the
Universitat Politècnica de Catalunya
Civil Engineering

October 2019

Abstract

This thesis aims to develop a High-Performance Computing implementation to solve large composite materials problems through the use of the FE2 multi-scale method. Previous works have not been able to scale the FE2 strategy to real size problems with mesh resolutions of more than 10K elements at the macro-scale and 100^3 elements at the micro-scale. The latter is due to the computational requirements needed to carry out these calculations. This works identifies the most computationally intensive parts of the FE2 algorithm and ports several parts of the micro-scale computations to GPUs. The cases considered assume small deformations and steady-state equilibrium conditions. The work provides a feasible parallel strategy that can be used in real engineering cases to optimize the design of composite material structures. For this, it presents a coupling scheme between the MPI multi-physics code Alya (macro-scale) and the CPU/GPU-accelerated code Micropp (micro-scale). The coupled system is designed to work on multi-GPU architectures and to exploit the GPU overloading. Also, a Multi-Zone coupling methodology combined with weighted partitioning is proposed to reduce the computational cost and to solve the load balance problem. The thesis demonstrates that the method proposed scales notably well for the target problems, especially in hybrid architectures with distributed CPU nodes and communicated with multiple GPUs. Moreover, it clarifies the advantages achieved with the CPU/GPU accelerated version respect to the pure CPU approach.

Keywords: **HPC, Composite Materials, FE2, Multi-Scale, CPU, GPU**

Contents

Abbreviations and Symbols

HPC	**H**igh **P**erformance **C**omputing
CPU	**C**omputational **P**rocessing **U**nit
GPU	**G**raphical **P**rocessing **U**nit
FE	**F**inite **E**lement
FEM	**F**inite **E**lement **M**ethod
FE2	**F**inite **E**lement Squared
RVE	**R**epresentative **V**olume **E**lement
MVP	**M**atrix **V**ector **P**roduct
CGDP	**C**onjugate **G**radients with **D**iagonal **P**reconditioner
SPD	**S**ymmetric **P**ositive **D**efinite

$\overline{\sigma}$	Macro-Scale Stress Tensor	N/m^2
$\bar{\epsilon}$	Macro-Scale Strain/Deformation Tensor	Dimensionless
$\overline{\mathbb{C}}$	Macro-Scale Tangent Constitutive Tensor	N/m^2
σ	Micro-Scale Stress Tensor	N/m^2
ϵ	Micro-Scale Strain/Deformation Tensor	Dimensionless
$\mathbb{C}$	Micro-Scale Tangent Constitutive Tensor	N/m^2
E	Young's Modulus	N/m^2
ν	Poisson Ratio	Dimensionless

Chapter 1

Introduction

1.1 Motivation and Objective

The Finite Element Squared (FE2) multi-scale method is a technique used to solve complex solid mechanics problems with considerable spatial scale differences ($L/l > 10^6$). This method is particularly well suited to study problems related to aeronautic composite material parts. These parts consist of a mixture of two or more materials with different mechanical properties, and when they are combined, produce a material with beneficial mechanical properties, for example, both high resistance and low mass density [1]. Fig. 1.1 outlines a 2D representation of a general composite material structure subjected to the forces and deformations in typical working conditions. This structure could, for example, represent a composite component of an aircraft like a wing. In an engineering and real 3D case, more than two materials can be used to build the composite structure. Furthermore, the constituents may display very complex and arbitrary shapes such as cylinders or spheres arranged in more intricate ways for maximizing the resulting mechanical properties.

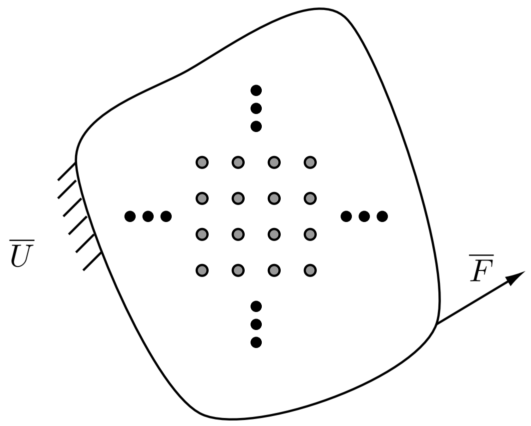


FIGURE 1.1: Basic representation of a composite material structure subjected to mechanical forces and deformations. The heterogeneous material is conformed with two or more materials with different mechanical properties. The resulting composite has beneficial mechanical properties that are not obtained by each of the constituents alone. In this representation, the difference between the length scales is less than 10^6 to give a better representation of a composite component.

The main interest in solving these kinds of problems numerically is directly related in reducing the cost (financial and time) of designing composite structures. The capacity to predict the behavior of a new composite model can increase the reliability and efficiency of the mechanical pieces that the real application is going to use. From the engineering point of view, different optimizations can be conducted to reduce the drag force or the weight of an aircraft. This design process can conceive a better mode of transport that demands less fuel consumption and releases less CO2. A typical numerical simulation of these physical systems generally consists of solving a solid mechanics problem, mathematically modeled with partial derivative equations (PDEs), that allows the engineers to know how the structures behave or evolve under different external forces and deformations (as in Fig. 1.1). Numerical methods provide a faster and cheaper solution to these problems in contrast to experimental studies. In any cases, a validation with physical experiments is required to verify the accuracy of the model. Generally, for large and complex problems like the ones which are frequently encounter in industry, High-Performance Computing (HPC) techniques are employed to perform the massive number of calculations demanded by the PDE system. The HPC component of the FE2 multi-scale method is the central topic covered in this thesis.

A wide variety of numerical techniques have been developed to solve the PDEs that represent a composite material structure. Most of these techniques use the Finite Element Method (FEM), the most common discretization in the solid mechanics area. Unfortunately, a naive approach applying one-scale FEM to solve heterogeneous composite material problems leads to a large numerical system which is challenging to deal with, even when using the most advanced computational technology available today. The massive computational cost of solving composite structure is a consequence of the high heterogeneity of these materials in a wide range of scales. For example a typical composite material panel may have about 40 layers with thousands of carbon fibers of approximately 50 µm of radius oriented in different directions. An FE mesh for this system should capture up to the smallest length scale (the fiber radius at least). The latter means that a simulation of an aircraft panel with an area of $1\,\mathrm{m}^2$ and a width of 1 cm for the 40 layers requires approximately 10^{18} finite elements to solve the problem accurately. The last fact demonstrates that the one-scale FEM is not a feasible solution for composite material problems. On the other hand, there exists an extra difficulty in creating or building a mesh able to capture both scales and to carrying out parameter analysis on it, for example, for optimizing the parts' shape or the micro-structure. For these cases, where there is a clear physical scales separation, the multi-scale methods are the most natural approach.

Despite the broad spectrum of multi-scale methods, it is possible to make a clear distinction between the usual phenomenological models and the one of central interest here, the FE2 scheme. Both approaches consider to decompose the original heterogeneous problem in two parts: a macro and a micro-scale problem. Fig. 1.2 outlines the multi-scale concept for a better understanding. The idea consists of using the FEM at the macro-scale and of applying a micro-scale model to calculate the average parameters of the mixture. The main difference between both approaches resides in the micro-scale model.

For phenomenological laws, the micro-scale consists basically in formulas that approximate the constitutive law of the mixture, making the method computationally cheap. The disadvantage is that most of the time, the methods rely on non-trivial assumptions and approximations and complex formulas derivations should be carried out. Furthermore, various parameters should be adjusted through the use of experimental data to gain enough accuracy.

On the other hand, the FE2 technique consists basically of modeling the micro-scale by a Representative Volume Element (RVE). This consists of a cubic domain containing all the details of the micro-structure. The RVE problem is solved by applying the FEM and the average macro-scale parameters can be obtained. In this way, the FE2 computes two nested and coupled FE problems solving the macro and micro-scale. The advantage of this method is that a small number of assumptions are needed to model the micro-scale mixture. Specifically, the strongest assumptions are the constitutive laws of each of the constituents in the RVE cell (the inclusions and the resin); these can be non-linear (non-elastic) in the worst case. It is much easier to assume and characterize the behavior of each constituent individually rather than the final mixture as in the phenomenological law approach. The FE2 method delegates the more complex effort to the FE process in the micro-scale to average automatically in comparison to the deduction of the phenomenological laws.

On the other side, the main disadvantage of the FE2 scheme resides in the massive computational cost needed to solve the large amount of micro-scale FE problems (one for each integration point of the macro-scale). To deal with this and to solve real engineering problems, the only computational solution available is to use fully parallelized algorithms that make use of robust computational architectures, i.e., Code_Aster [2], Code_Saturn [3] or Dealii [4] between others. Until now, the computational challenges for the FE2 multi-scale method have been the main obstacle. Fortunately, some work has been done to tackle the difficulties in the past. This thesis moves a step forward towards the applicability of this method in industry.

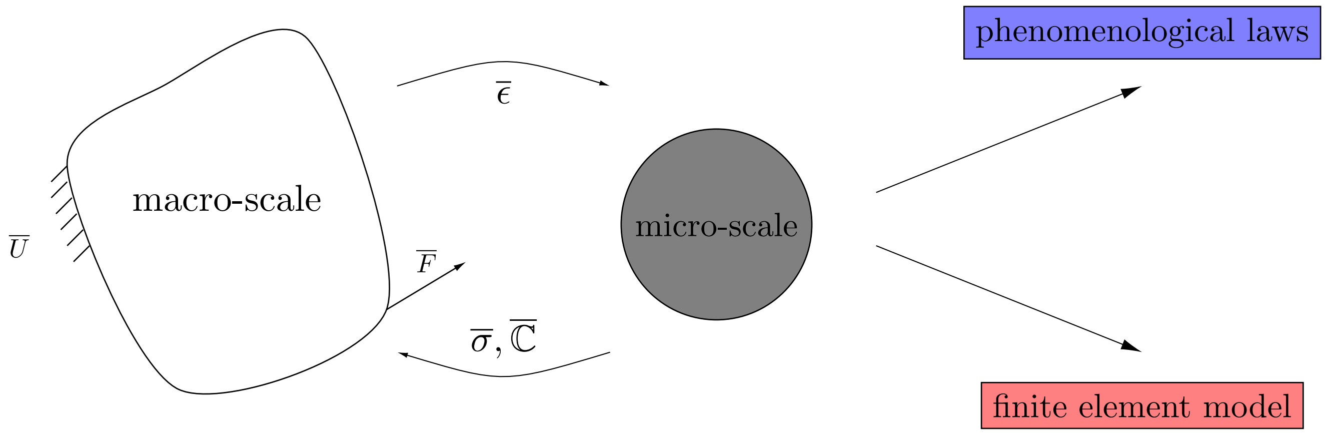


FIGURE 1.2: Basic scheme of the multi-scale approaches. A main distinction can be done separating those that use phenomenological models in the micro-scale and the FE2 method which uses the FEM to perform the averaging there. The first is computationally cheap but lot of assumptions and parameter calibrations should be done to achieve accuracy, the second is the opposite.

The objective of this thesis is to develop, implement, and test a robust FE2 multi-scale tool for the aircraft industry. The work proposes a parallel scheme that combines an MPI-based code for the macro-scale and a hybrid CPU/GPU accelerated version for the micro-scale. Also, it demonstrates that this technique allows the FE2 strategy to be valid for engineering purposes and that it is massively parallel and prepared for Exascale computing. The study makes a clear proposal of how to apply the methodology in practice and analyses the algorithm in detail in conjunction with the parallel performance.

The work is structured as follows. Chapter 2 presents the critical details of the FE2 algorithm. First describes the governing equations of the system, then the numerical FE discretization, and finally, the application of the Admissible Boundary Conditions. Chapter 3 clarifies the main computational obstacles that appear in the multi-scale scheme, focusing on the implications of non-linearities and the coupling scheme used to communicate between scales. It also illustrates a real and feasible example of how to apply the algorithm for engineering purposes. Chapter 4 reveals the technical and performance aspects of the micro-scale calculation; this is the most computationally intensive part of the whole algorithm. Also, it encourages the use of the solver and preconditioner type used and exposes the performance results for CPU/GPU and CPU. Chapter 5 is advocated to explain the parallel algorithm implemented and proposed here, it analyses the coupled performance with hybrid architectures (CPUs and GPUs) and points out a weighted partition strategy to solve the load unbalance problem. Chapter 6 is dedicated to solving engineering problems with the proposed scheme demonstrating the parallel performance achieved. Finally, Chapter 7 summarizes the main conclusions of this work and opens new research topics in the field to advance to the Exascale computing. An appendix is included at the end describing some additional technical details of the Micropp code.

1.2 Background

The technological advances in hardware and software during the last decades have allowed the use of the FE2 multi-scale method for solving real engineering scenarios. The most significant limitation of this multi-scale algorithm resides in the high demand for FE computations at the micro-scale level. Furthermore, the method requires to store a considerable amount of data for solving non-linear problems meaning that distributed computing is the only possible way to solve this problem.

The distributed computing applied to the scientific numerical simulations consists basically in decomposing a large problem into smaller ones and to solve them with a computational processor and its memory resources. The latter methodology allows a substantial computational time reduction and solves problems associated to memory requirements. The Message Passing Interface (MPI) is the most used protocol in the scientific community to perform parallel and distributed computations. All work reviewed in this chapter uses MPI to perform the multi-scale calculations. MPI enables users to focus on the algorithmic part of the problem rather than on the management of the network protocols. Other parallel models can be applied to perform parallel computations, i.e., using shared memory-based methods [5]. The most advanced works that have applied distributed computing to solve the FE2 multi-scale problem are [6, 7, 8, 9]. This section discusses their most important contributions.

The parallel and distributed strategy proposed by Matsui [6] in 2004 consists of solving the macro-scale FE problem sequentially and distributing the computations of all the micro-scale problems to the different processors. The hardware architecture they used consists of 8 PCs communicated by the MPI protocol. Fig. 1.3 represents the parallel distribution of the processes in the hardware. In the latter, a single processor in one PC allocates the macro-scale process, and the rest of the PCs allocate a group of micro-scale problems to be solved. Matsui [6] demonstrates that the MPI communication time is negligible for this implementation and attributes most of the computational cost to the micro-scale calculations. Also, it displays that the presence of non-linearities creates a load unbalance for this scheme and collapses the parallel performance. To solve this issue, the author proposes a load balance strategy that uses the first time step solution to calculate the work done by each micro-scale problem and then redistributes the work in approximately equally large sub-problems to each MPI process. The work achieves to solve problems of 600 elements in the macro-scale and 6^3 elements in the micro-scale [1]. This mesh resolution is small for the industry demands but Matsui [6] was one of the first in applying parallel computing methodologies to the FE2 method (2004).

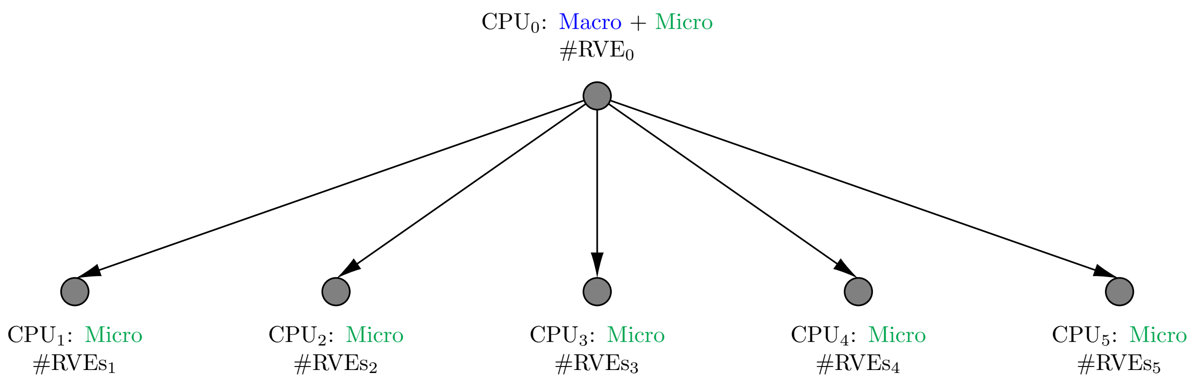


Figure 1.3: Parallel MPI-based implementation of the FE2 method used by Matsui [6] in 2004. The scheme solves the macro-scale sequentially in one process and distributes all the micro-scale problems (number of RVEs) in parallel to the rest of CPUs.

The work of Rahul [7] in 2010 moves a step forward and presents a hybrid MPI/OpenMP implementation of the FE2 multi-scale method. It proposes two MPI-based parallelization schemes. The first is quite similar to the work of [6] visualized in Fig. 1.3. The second consists of performing a domain decomposition of the macro-scale assigning a processor to allocate each sub-domain. Each processor is responsible for solving the micro-scale problems of the corresponding macro-scale sub-domain. The latter accelerates the execution with OpenMP as outlined in Fig. 1.4.

Unfortunately, Rahul [7] does not show any parallel performance of the MPI/OpenMP strategy because each computing node has only one physical core. The results outline that the second approach reduces the communications overhead of the FE2 multi-scale method increasing the parallel efficiency with respect to the first proposal. This work was able to deal with problems of around 1K elements at the macro-scale and 10^3 at the micro-scale.

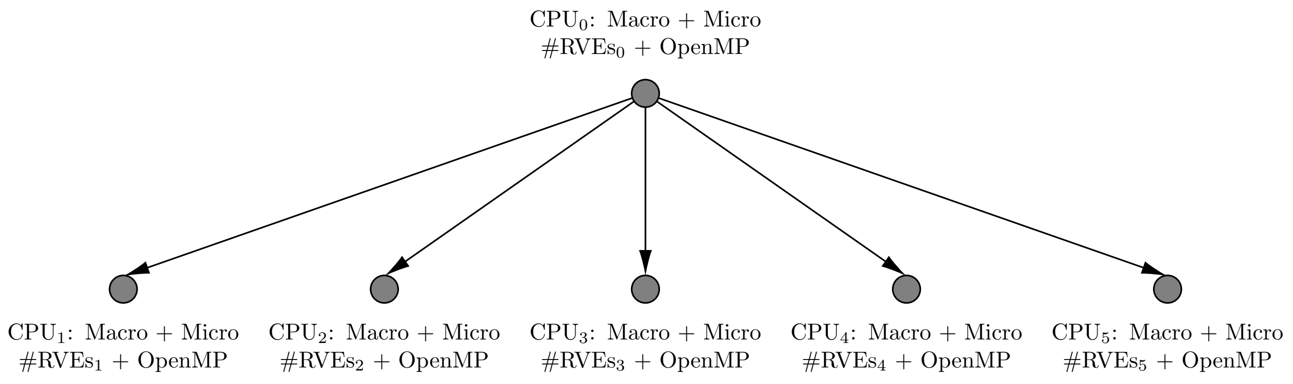


Figure 1.4: Parallel implementation of the FE2 method proposed by Ref. [7] in 2010. A domain decomposition method is applied at the macro-scale and all the micro-scale problems (number of RVEs) are distributed to the different processors where their sub-domains are allocated. The calculation of the micro-scale problems is accelerated with OpenMP.

More recent studies such as of Klawonn and Balzani [8, 9] propose a hybrid MPI/OpenMP implementation based on domain decomposition at the micro-scale. The strategy permits to solve very complex problems at the micro-level but limited size macro-scale ones. They achieve to solve problems around 0.2K elements at the macro-scale

[1]The exponent notation X^3 (elements) is used across this work to specify the number of elements for the micro-scale mesh resolution since the implementation developed here uses structured grids.

and 100^3 elements at the micro-scale using near 1 million CPU cores (40 time steps). They introduce the One-Way and Full coupling schemes to simplify the FE2 computational cost.

The four articles presented here are the most detailed works which have applied the FE2 multi-scale algorithm in parallel using distributed computing strategies (all of them with MPI). In all the cases, the micro-scale calculation step is the most computationally intensive part. No author has attempted to solve both scales by means of domain decomposition techniques (using MPI in both scales). This approach presents some prohibiting difficulties that are discussed in Chapter 5. This work implements an improved version based on the articles [6, 7] by accelerating the micro-scale part with multi-core CPUs (with OpenMP), and with GPUs (with OpenACC). This last technique, accelerating with GPU computing, can solve problems bigger than 10K elements at the macro-scale and 100^3 elements at the micro-scale constituting a breakthrough in this field.

1.3 Contributions

The main contributions of the present work are enumerated as follows:

- This works develops a 3D FE2 multi-scale algorithm that solves large non-linear composite material problems for industry. The implementation can run on modern computational architectures comprised of distributed multi-core CPUs processors and multiple GPUs. The implementation primarily consists in the coupling of the MPI-based and multi-physics code Alya and the micro-scale code Micropp. This last was designed and entirely implemented from scratch in C++ language and ported to multi-core CPUs with OpenMP and to GPUs with OpenACC. This decision boosts the portability and the reusability of both codes while at the same time, accelerates the code execution dramatically.

- The thesis presents realistic cases and examples for applying the FE2 multi-scale method and in a robust way to fit the demands of industry. The work focuses on the computational cost of each part of the algorithm making a precise analysis and recognition of which parts of the code should be optimized and accelerated. The work also suggests different coupling schemes that serve to communicate both scales, further reducing the computational cost while preserving the accuracy at the same time.

- The most important aspects of the Micropp code are presented. This is the code used to solve the micro-scale to be coupled with generic macro-scale codes to carry out the FE2 computations. Special consideration is paid on it because this is the computationally intensive part of the algorithm. Micropp offers a significant extra advantage in contrast to other codes in the literature since it is Free Software [10]. Moreover, it works efficiently on different architectures such as Intel and IBM machines, and on NVIDIA GPUs. It offers a reliable configuration and compilation process which uses CMake and, a test suite environment developed through CTest (an internal tool of CMake). The Micropp documentation [10] gives specific details of how the code works.

- As shown in Fig. 1.5 the FE2 multi-scale implementation accelerated with CPU/GPU has shown a suitable performance against other works such as [6, 7, 8, 9]. The Figure applies a comparison based on the mesh resolution at both scales, which is a suitable engineering parameter because it gives a notion of which problems can be solved and which not. Notice that the present scheme with Alya (MPI-based) and Micropp with CPU/GPU acceleration opens new possibilities and expands the resolution ranges to deal with larger composite problems for the aeronautic industry and to solve the usual composite problems in less time.

- The thesis provides references to many Micropp benchmarks that are accessible by the readers and Micropp users through [10] to run them in different architectures with CPU/GPU capabilities. The benchmarks permits to evaluate and compare the execution of the micro-scale computations analysing the performance in different hardware and software resource (libraries and compilers).

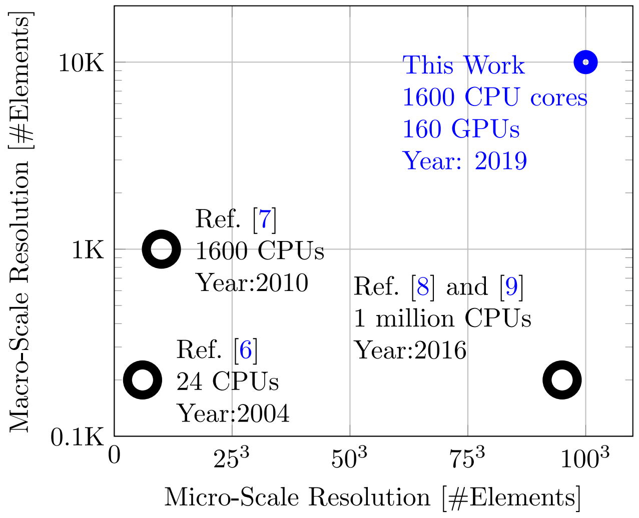


FIGURE 1.5: Micro-scale and macro-scale problem sizes of the current FE2 multi-scale implementation and other comparable works in literature [6, 7, 8, 9].

For this thesis an article titled "An FE2 multi-scale implementation for modeling composite materials on distributed architectures" [11] was published in the Coupled Systems Mechanics Journal for the CILAMCE 2018 conference at Paris, France. This publication consists of a preliminary implementation of the FE2 multi-scale method through the coupling of Alya (MPI-based code) and Micropp (used in sequential mode). The work shows a comparison of the mechanical results of this FE2 implementation against the one-scale FE model for non-linear composite material problems. It demonstrates that the results of the two methods match for very periodic media. The article also shows the results of parallel scalability experiments carried out in the MareNostrum4 supercomputer up to 10K cores with cases composed of 100K elements in the macro-scale and 10^3 elements in the micro-scale. The last results are complementary to the present work.

Finally, this work includes a presentation of a poster titled "Hybrid CPU/GPU FE2 multi-scale implementation coupling Alya and Micropp" [12] at the SuperComputing Conference, 2019. This last work presents simulations and comparisons of the CPU/GPU strategy with the pure CPU approach. The benchmark cases are also described in Chapter 6 of this thesis.

Chapter 2

The FE2 Multi-Scale Method

The objective of this chapter is to explain and describe the most fundamental aspects of the FE2 multi-scale method. The first section presents the governing equations of the FE2 algorithm. The second section exposes the FE discretization, and the Newton-Raphson procedure to solve the non-linear equilibrium problem. The third section displays the Admissible boundary conditions for the micro-scale problem, and some numerical results are provided to compare them.

2.1 Governing Equations

The FE2 multi-scale method is based on breaking the original single scale and highly heterogeneous problem in a two separate problems: one at the macro-scale and the other at the micro-scale. These problems are coupled together and both are solved with the FEM in their corresponding domain. The macro-scale FE mesh consists in a coarse mesh where each element can hold thousands or millions of micro-scale constituents. For a carbon fiber panel one macro-element can contain thousands or millions of fibers. The average constitutive law that it is used for the macro-elements is provided by the micro-scale FE model. The micro-scale model consists in a RVE, which is generally a cubic domain, where the FEM is applied to obtain the stress fluctuations and finally to compute the average of it which is passed back to the macro-scale.

In this work the steady state condition and the small deformation approach are assumed in the FE2 multi-scale formulation. For the steady state case the inertial force term is neglected. This assumption is valid for a wide variety of aeronautic problems specially for low speed impacts simulations, i.e, when the speed of the object is small in comparison with the speed of the sound in the solid. On the other hand, the small deformation approach considers a linear relation between the deformation of strain tensor ϵ and the displacements u:

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{2.1}$$

The small deformation assumption is valid for those cases where the deformation remains small ($|\epsilon| < 1 \times 10^{-3}$). The linear nature of this formula makes the problem to be geometrically linear but non-linearities can appear due to the presence of non-linear constitutive laws in the constituents of the micro-scale [1]. This problem will be threaded in the next chapters.

Applying these considerations, the equations at the macro-scale level can be written as:

$$\begin{cases} \nabla \cdot \overline{\sigma} = 0 & \text{in } \overline{\Omega}, \\ \overline{u} = \overline{u}_d & \text{in } \overline{x} \in \overline{\Gamma}_d, \\ \overline{\sigma} \cdot \hat{n} = \overline{\sigma}_n \cdot \hat{n} & \text{in } \overline{x} \in \overline{\Gamma}_n, \\ \overline{\sigma} = \langle \sigma \rangle (\overline{\epsilon}). \end{cases} \tag{2.2}$$

The variables $\overline{u}$, $\overline{\epsilon}$ and $\overline{\sigma}$ are the displacement, the strain and the stress fields respectively. On the other hand, $\overline{\Omega}$ is the macro-scale domain where these variables are defined [2]. The second and third equations are the boundary conditions at the macro-scale. In this case: Dirichlet and Neumann constrains are applied respectively. These can be combined or mixed in the same region of the boundary ($\overline{\Gamma}_n \cap \overline{\Gamma}_d \neq \emptyset$), it is often done in typical solid mechanics problems. The last equation is the average constitutive material law that relates the macro-stress $\overline{\sigma}$ with the macro-strain $\overline{\epsilon}$. This relation it is not known a priori and should be determined by solving the micro-scale FE problem.

For the micro-scale problem the governing equations are:

$$\begin{cases} \nabla \cdot \sigma = 0 & \text{in } \Omega, \\ \sigma = f(\epsilon, q), \\ \text{boundary conditions.} \end{cases} \tag{2.3}$$

The system represents an integration point of the macro-scale where the average constitutive law is evaluated. The variables u, ϵ and σ are the displacement, the strain and the stress fields respectively, while Ω is the micro-scale domain. The first equation is the equilibrium of forces similar to the macro-scale. The second is the constitutive law of each of the material of the micro-structure. As a difference with the macro-scale these laws are well know here.

The boundary conditions of the micro-scale reflect the deformation of the RVE that is theoretically placed in the integration point of the macro-scale. It is not possible to know the exact deformation of the RVE without solving the one-scale FE problem. Logically this is not going to be done when the FE2 multi-scale method is used. For this multi-scale technique it is important to remark that the boundary conditions at the micro-scale are approximated as a function of the macro-scale strain tensor $\overline{\epsilon}$. This process is illustrated in Fig. 2.1.

The process of imposing the boundary conditions and then solving the FE problem is called Localization of the macro-strain $\overline{\epsilon}$. Physically the desired condition in the micro-scale when the equilibrium is achieved is that:

$$\overline{\epsilon} = \frac{1}{\text{vol}(\Omega)} \int_{\Omega} \epsilon \, dV. \tag{2.4}$$

[1]It is said hat the problem is Geometrically Linear but Materially Non-Linear.

[2]The line $\bar{\ }$ means that they are defined for the macro-scale, in the other case they are defined in the micro-scale.

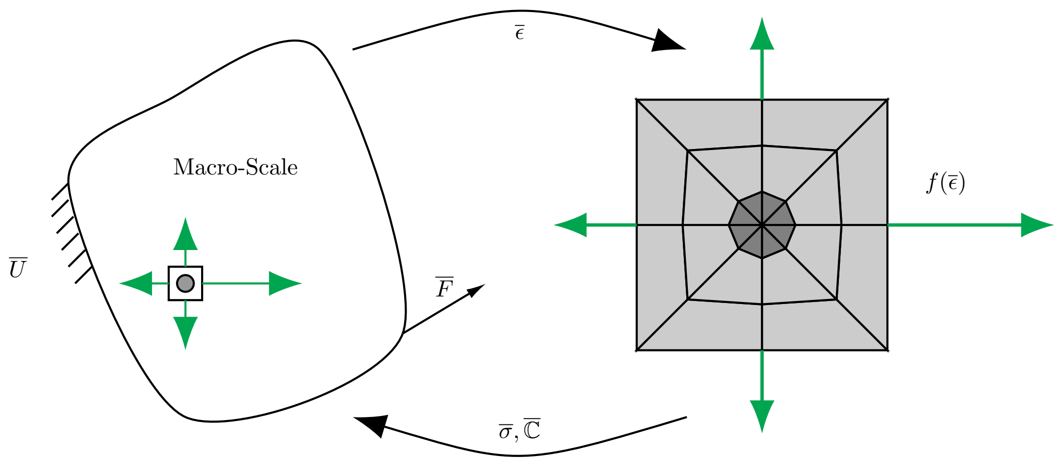


FIGURE 2.1: The boundary conditions applied in the micro-scale problem are a function of the macro-strain $\overline{\epsilon}$. The deformation of the RVE should be similar to what the cell would have in situ in order to obtain accurate average stress values.

This relation defines the coupling in the macro-to-micro direction. The evaluation of the Eq. 2.4 is not strictly necessary necessary but it should be satisfied when the boundary conditions are applied and the equilibrium is achieved in the RVE. Eq. 2.4 can be used to check that the boundary conditions have been implemented correctly. Not all the boundary conditions work and satisfy Eq. 2.4, the ones that do are called Admissible Boundary Conditions and are the ones that should be implemented in the micro-scale.

The coupling in the micro-to-macro direction is done by calculating the macro-scale stress $\overline{\sigma}$ in the RVE, this procedure is called Homogenization and it is performed after the macro-strain $\overline{\sigma}$ has been localized, i.e. after solving the FE problem and reaching the equilibrium. The homogenization is done simply by:

$$\overline{\sigma} = \frac{1}{\mathrm{vol}(\Omega)} \int_{\Omega} \sigma \, dV. \tag{2.5}$$

Eqs. 2.4 and 2.5 define the coupling between both scales for the FE2 scheme. The crucial approximation are the boundary conditions applied at the micro-scale, these are a function of the macro-strain tensor $\overline{\epsilon}$. Three kind of Admissible Boundary Conditions are known: the Uniform Strain, the Uniform Stress and the Periodic boundary conditions.

The Uniform Strain consists in a pure Dirichlet boundary condition that prescribes the displacements at the boundary as:

$$u(x) = \overline{\epsilon} \cdot x \quad \text{for} \quad x \in \Gamma. \tag{2.6}$$

The Uniform Stress boundary condition is defined in terms of the force at the boundary as:

$$\sigma(x) \cdot \hat{n} = \overline{\sigma} \cdot \hat{n} \quad \text{for} \quad x \in \Gamma. \tag{2.7}$$

Finally the Periodic boundary condition is a combination of forces and displacements in the boundary:

$$u(x^+) - u(x^-) = \bar{\epsilon} \cdot (x^+ - x^-) \ \text{ and } \ \sigma(x^+) \cdot \hat{n} + \sigma(x^-) \cdot \hat{n} = 0. \tag{2.8}$$

It is possible to demonstrate that the three kind of boundary conditions comply Eq. 2.4 [13]. The selection of one of these conditions determines the accuracy of this method. The Periodic boundary condition generally gives the most accurate results of the three kinds due to the periodic nature of the composite materials.

2.2 Finite Element Discretization

The goal of this section is to discretize Eqs. 2.2 and 2.3 by applying the FEM. Both equilibrium equations are equivalent so the derivation of the discrete system of equation will be done for only one. The first step for applying the FEM is to obtain the Weak Form of the partial derivative equation $\nabla \cdot \sigma$ [14]). This can be derive directly from the Virtual Work Principle of the solid mechanics problem [15]:

$$r(u) = \int_\Omega \sigma(u) : \delta\epsilon \, dV - \int_\Gamma \sigma \cdot \hat{n} \cdot \delta v \, dS = 0. \tag{2.9}$$

Here, r is the Residue of the system. The test function δv is added into the Week Form of the equations and the virtual strain is calculated as in Eq. 2.1 by $\delta\epsilon = \delta\epsilon(\delta u)$. The problem is translated into finding the displacement u which makes the Residue to vanish for every test function δv.

The next step in the FEM is to propose an approximation for the displacement field as:

$$\tilde{u}(x) = \sum_i^N u_i \psi_i(x), \tag{2.10}$$

where ψ_i are the shape functions defined in the finite element mesh composed of N nodes. Then, substituting this approximation into Eq. 2.9 it is possible to derive a discrete system of equations:

$$r_j(\tilde{u}) = \int_\Omega \sigma(\tilde{u}) : \delta\epsilon_j \, dV - \int_\Gamma \sigma \cdot \hat{n} \cdot \delta v_j \, dS = 0 \quad \text{for} \quad j = 1 \dots N. \tag{2.11}$$

Considering that the problem is non-linear, the multi-dimensional Newton-Raphson iterative procedure is used to obtain the equilibrium. Taking the Taylor expansion of the Residue of Eq. 2.11:

$$r_j(\tilde{u}) = r_j(\tilde{u}_0) + \left[\frac{dr}{du}\right]_{\tilde{u}_0} du = 0, \tag{2.12}$$

it is possible to derive an approximation for the increment du. The iterative procedure continues by finding successive values of the displacement increment du and calculating new values of the Residue vector $r(u)$ and the unknown displacement u as:

$$\begin{aligned} &\left[\tfrac{dr}{du}\right]_{\tilde{u}_0} du = r(u_{n+1}^k), \\ &u_{n+1}^{k+1} = u_{n+1}^k + du. \end{aligned} \tag{2.13}$$

The matrix $\mathbb{A} = \left[\frac{dr}{du}\right]_{\tilde{u}_0}$ is called Jacobian. The size of the Jacobian is $N \cdot \mathrm{DIM} \times N \cdot \mathrm{DIM}$ where DIM is the spatial dimension of the problem; for this work 3D problems are covered (DIM = 3). The shape functions of Eq. 2.11 are defined zero in most of the parts of the domain, this makes the Jacobian to be a sparse matrix, i.e. a matrix with a large number of zero coefficients. This property allows to represent computationally the Jacobian matrix with a compact format such as CSR or ELLPACK in order to reduce the memory required.

The linear system $\mathbb{A}\, du = r(u_{n+1}^k)$ can be solved in different ways but most of the time, for the FEM, the matrix is not inverted and direct or iterative solvers are used. For massively huge problems (meshes with more than 1 million elements) iterative solvers are preferred due to the low demand of data memory storage and the efficient parallel algorithms that exist for implementing them [5].

The Jacobian matrix A and the Residue vector r are built in a process which is called assembly. Basically Eq. 2.11 is an integral in the whole volume Ω so the assembly processes consists in building, for each mesh element, an elemental Jacobian matrix $\mathbb{A}_e$ and an elemental Residue vector r_e. These are added to the corresponding positions of $\mathbb{A}$ and r:

$$\mathbb{A} = \sum_e^{n_e} \mathbb{A}_e \ \text{ and } \ r = \sum_e^{n_e} r_e. \tag{2.14}$$

The elemental Jacobian matrix $\mathbb{A}_e$ and the elemental Residue vector r_e are calculated by the numerical integration quadrature rules as:

$$\mathbb{A}_e = \sum_g^{n_g} \mathbb{B}^T \mathbb{C}\, \mathbb{B}\, w_g \ \text{ and } \ r_e = \sum_g^{n_g} \mathbb{B}^T \sigma\, w_g. \tag{2.15}$$

where $\mathbb{B}$ is defined in terms of the shape functions derivatives as:

$$\mathbb{B} = \begin{pmatrix} \frac{\partial\psi}{\partial x} & 0 & 0 \\ 0 & \frac{\partial\psi}{\partial x} & 0 \\ 0 & 0 & \frac{\partial\psi}{\partial x} \\ 0 & \frac{\partial\psi}{\partial z} & \frac{\partial\psi}{\partial y} \\ \frac{\partial\psi}{\partial z} & 0 & \frac{\partial\psi}{\partial x} \\ \frac{\partial\psi}{\partial y} & \frac{\partial\psi}{\partial x} & 0 \end{pmatrix}. \tag{2.16}$$

The $\mathbb{B}$ matrix relates the strain ϵ with the displacement u in the interior of an element as $\epsilon = \mathbb{B}\, u$. On the other hand, $\mathbb{C}$ is the tangent constitutive tensor that relates the variation of the stress $\overline{\sigma}$ respect to the strain ϵ:

$$\mathbb{C} = \frac{d\sigma}{d\epsilon}. \tag{2.17}$$

When the constitutive material law $\sigma = f(\epsilon, q)$ has a simple expression, it is possible to find an analytic expression for $\mathbb{C}$. Unfortunately most of the time this does not happen and other numerical procedures are generally used. A simple technique consists of applying small perturbations $\delta\epsilon$ around the strain $\overline{\epsilon}$ for each of the six components $j = 1 \ldots 6$ (three components in two dimension problems) like:

$$\overline{\epsilon}_i^j = (1 - \delta_{ij})\,\overline{\epsilon}_i + \delta_{ij}\,\delta\overline{\epsilon} \;\text{ with }\; j = 1 \ldots 6. \tag{2.18}$$

Then, the constitutive law $\sigma^j = f(\epsilon^j)$ is evaluated for each j value and also for ϵ to obtain σ. Finally, each column of $\mathbb{C}$ can be approximated as:

$$\mathbb{C}^j \approx \frac{\sigma^j - \sigma}{\delta\epsilon}. \tag{2.19}$$

For the macro-scale problem, rather than to evaluate a constitutive law, the strain $\overline{\epsilon}^j$ is localized in the micro-scale's RVE and the stress $\overline{\sigma}^j$ is obtained by the homogenization procedure. So, basically seven homogenizations are required to calculate this tensor:

$$\overline{\mathbb{C}}^j \approx \frac{\overline{\sigma}^j(\text{FE computation}) - \overline{\sigma}(\text{FE computation})}{\delta\overline{\epsilon}}. \tag{2.20}$$

Being needed six FE computations, this process constitutes one of the most computationally intensive parts of the micro-scale calculation. In Chapter 3 a Multi-Zone strategy, which avoids the computation of this tensor along the entire volume, is explained.

2.3 Admissible Boundary Conditions

The compliance of Eq. 2.4 in the micro-scale problem is achieved strictly by the imposition of Admissible boundary conditions on the RVE. For the FE2 implementation developed in this work the Uniform Strain boundary condition was applied. Despite of this, a study of the Uniform Stress and the Periodic boundary conditions is performed in this section in order to analyse a future implementation for the HPC-application. The implementation of these boundary conditions is based on the works of [16, 17].

In the three cases f_a represent the components of the nodal forces in the interior nodes of the FE mesh and f_b are the forces in the boundary nodes. The same notation is used for the displacement: u_a are the displacements in the internal nodes and u_b in the boundary nodes.

Uniform Strain

This boundary conditions prescribes the displacements at the boundary with the form of $u_b = \overline{\epsilon} \cdot x$. By this way the Residue vector of the micro-scale can be written as:

$$r(u) = \begin{pmatrix} f_a(u) \\ f_b(u) - \delta \\ u_b - \bar{\epsilon} \cdot x \end{pmatrix}. \tag{2.21}$$

Where δ is a Lagrange multiplier needed to ensure the net force in the boundary to be different from zero.

To solve this non-linear system, the Newton-Raphson procedure is used. For this, it is necessary to calculate the Jacobian matrix defined as $\mathbb{A} = \frac{\partial r}{\partial x}$, where x represents the unknowns in Eq. 2.21:

$$x = (u_a,\, u_b,\, \delta)^T. \tag{2.22}$$

Applying the derivative to the Residue in Eq. 2.21, the Jacobian is expressed as:

$$\mathbb{A}(u) = \begin{pmatrix} \frac{\partial f_a}{\partial u_a} & \frac{\partial f_a}{\partial u_b} & 0 \\ \frac{\partial f_b}{\partial u_a} & \frac{\partial f_b}{\partial u_b} & \frac{\partial \delta}{\partial \delta} \\ 0 & \frac{\partial u_b}{\partial u_b} & 0 \end{pmatrix} = \begin{pmatrix} \mathbb{K}_{aa} & \mathbb{K}_{ab} & 0 \\ \mathbb{K}_{ba} & \mathbb{K}_{bb} & \mathbb{I} \\ 0 & \mathbb{I} & 0 \end{pmatrix}. \tag{2.23}$$

The displacement increment is then calculated as:

$$du = -\mathbb{A}^{-1}(u)\, r. \tag{2.24}$$

A variant of this implementation can be used; this consists of simplifying the Residue vector expression as:

$$r = \begin{pmatrix} f_a \\ 0 \end{pmatrix} \Rightarrow \mathbb{A}(u) = \begin{pmatrix} \mathbb{K}_{aa} & 0 \\ 0 & \mathbb{I} \end{pmatrix}, \tag{2.25}$$

while at the same time the initial displacement is set: $u_b = \bar{\epsilon} \cdot x$.

Uniform Stress

The formulation to apply this boundary condition considers an equivalent expression of Eq. 2.7 that can be written and discretized as [17] :

$$\frac{1}{V} \int_\Gamma sym[u_q \otimes \hat{n}] dS = \bar{\epsilon} \quad \Rightarrow \quad \frac{1}{V} \sum_q^M \mathbb{S}_q u_q = \bar{\epsilon}. \tag{2.26}$$

The index q goes around all the boundary element of the domain and the matrix S_q relates the area of this element in order to comply with Eq. 2.26. In this way the expression for the Residue vector is written as:

$$r(u) = \begin{pmatrix} f_a(u) \\ f_b(u) - \mathbb{S}^T \delta \\ \mathbb{S}\, u_b - \overline{\epsilon} \end{pmatrix}. \tag{2.27}$$

Where δ is a Lagrange parameter that coincides with the average macro-scale stress tensor $\overline{\sigma}$. Consequently, after applying the derivatives, the Jacobian matrix is expressed as:

$$\mathbb{A}(u) = \begin{pmatrix} \mathbb{K}_{aa} & \mathbb{K}_{ab} & 0 \\ \mathbb{K}_{ba} & \mathbb{K}_{bb} & -\mathbb{S}^T \\ 0 & \mathbb{S} & 0 \end{pmatrix}. \tag{2.28}$$

Notice that this method incorporates 3 degrees of freedom for the 2D case and 6 for the 3D case.

Periodic

The Periodic boundary condition can be applied in two possible ways: by using the Lagrange Multipliers method or by the Master-Slave technique. For both cases it is necessary to assume that each node in the boundary has a symmetric node in the opposite side of it, as it is indicated in Fig. 2.2 for a 2D example with x_i^- and x_i^+.

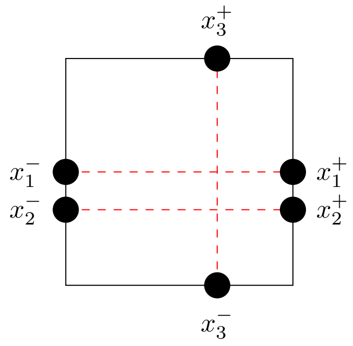


FIGURE 2.2: To apply the Periodic boundary condition it is necessary that each node in the boundary has a symmetric node in the opposite side of the boundary, i.e., (x_i^-, x_i^+).

The boundary conditions of Eq. 2.8 can be included inside the Residue vector expression adding the Lagrange Multiplier δ as:

$$\begin{cases} f_a(u) = 0, \\ f_b(u) - \mathbb{P}^T \delta = 0, \\ \mathbb{P}\, u_b - \mathbb{Q}\, \overline{\epsilon} = 0. \end{cases} \tag{2.29}$$

Being $\mathbb{P}$ a matrix with ones and zeros that produce the antisymmetry effect of the force. Notice that this matrix is the same that it is needed to produce the symmetry in the displacement constrain of the final expression in Eq. 2.29. On the other hand, the matrix $\mathbb{Q}$ is the one that contains the information about the distances of the symmetric pairs of nodes $(x_i^+ - x_i^-)$.

Applying the derivatives to Eq. 2.29 the resultant Jacobian matrix is:

$$\mathbb{A}(u) = \begin{pmatrix} \mathbb{K}_{aa} & \mathbb{K}_{ab} & 0 \\ \mathbb{K}_{ba} & \mathbb{K}_{bb} & -\mathbb{P}^T \\ 0 & -\mathbb{P} & \mathbb{Q}^T \end{pmatrix}. \tag{2.30}$$

The Master-Slave intends to eliminate the use of extra Lagrange Multiplier variables basically taking into account that:

$$u_i^+ - u_i^+ = \bar{\epsilon} \cdot (x^+ - x^-) \Rightarrow du^+ - du^- = 0 \Rightarrow du^+ = du^-. \tag{2.31}$$

Considering the anti-symmetry for the forces $f_b^+ = f_b^-$ the system represented in Eq. 2.29 can be rewritten as:

$$\begin{cases} f_a(u) = 0, \\ f_b^-(u) + f_b^+(u) = 0. \end{cases} \tag{2.32}$$

For the set of unknowns:

$$x = [u_a,\, du^+]^T \tag{2.33}$$

the Jacobian matrix is in this case:

$$\mathbb{A}(u) = \begin{pmatrix} \mathbb{K}_{aa} & \mathbb{K}_{a+} + \mathbb{K}_{a-} \\ \mathbb{K}_{+a} + \mathbb{K}_{-a} & \mathbb{K}_{++} + \mathbb{K}_{-+} + \mathbb{K}_{+-} + \mathbb{K}_{--} \end{pmatrix}. \tag{2.34}$$

Numerical Comparison in 2D

The three kind of Admissible boundary conditions were implemented in a simple 2D FE code [18]. The code simulates an square cell with a empty circular inclusion, as it is shown in Fig. 2.3. A structured mesh of 1002 elements has been used for the FE discretization. The non-empty volume of the cell has a Young's Modulus of $E_f = 1.0 \times 10^7\,\mathrm{N/m^2}$ and a Poisson Ratio of $\nu_m = 0.25$. On the other hand, the empty circle is modeled with a Young's Modulus $E_f = 3.0 \times 10^3\,\mathrm{N/m^2}$ and a Poisson Ratio of $\nu_m = 0.25$ for the fiber. Fig. 2.3 shows the deformed cells and the internal stress field for the three kind of boundary conditions. These results correspond to the localization of a pure share strain $\bar{\epsilon}_{xy} \neq 0$, $\bar{\epsilon}_{xx} = \bar{\epsilon}_{yy} = 0$.

As it is explained by [19], for a fictitious RVE placed in the macro-structure, the energy $E \approx \bar{\sigma} : \bar{\epsilon}$ achieved would be the minimum. Any other deformation would increase the internal energy stored in the RVE ($E \uparrow$). For the Uniform Strain boundary condition the displacement is prescribed at the boundary ($\bar{\epsilon}$ is fixed), this produce an increase in the average macro-stress because $E \uparrow \Rightarrow \bar{\sigma} \uparrow$. For the Uniform Stress boundary condition the macro-stress is fixed $\bar{\sigma}$. This is translated in a decrement of the macro-scale strain because $E \uparrow \Rightarrow \bar{\epsilon} \uparrow$. In conclusion a harder average material is modeled when the Uniform Strain boundary condition is used and the opposite happen for the Uniform Stress boundary condition. The Periodic boundary condition is an intermediate term between the other two.

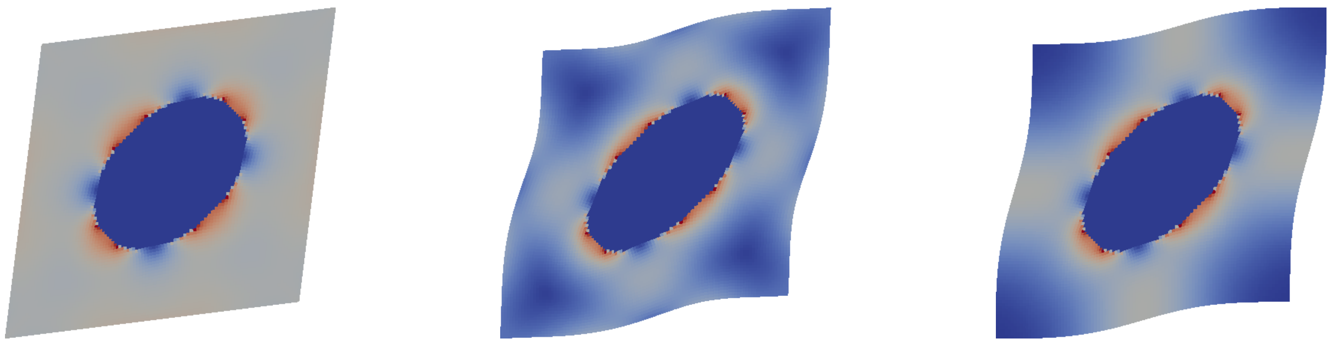

(a) Uniform Strain (b) Uniform Stress (c) Periodic

FIGURE 2.3: Deformed cell for the three kind of Admissible boundary conditions applying a pure shear strain $\bar{\epsilon}_{xy} \neq 0$, $\bar{\epsilon}_{xx} = \bar{\epsilon}_{yy} = 0$.

The quantitative results, obtained with [18], for a pure traction case ($\bar{\epsilon}_{xx} \neq 5 \times 10^{-3}$, $\bar{\epsilon}_{xy} = \bar{\epsilon}_{yy} = 0$, demonstrate that the homogenized stress:

$$\underbrace{\overline{\sigma}_{xx} = 3.2 \times 10^4}_{\text{uniform stress}} \leq \underbrace{\overline{\sigma}_{xx} = 3.8 \times 10^4}_{\text{periodic}} \leq \underbrace{\overline{\sigma}_{xx} = 3.9 \times 10^4}_{\text{uniform strain}}, \tag{2.35}$$

comply with the previous hypothesis made by [19].

2.4 Conclusions

This chapter has presented the governing equations of the FE2 multi-scale method. It has been shown that the boundary conditions applied at the micro-scale determine the accuracy of the method. These should be a good approximation of the deformation of a cell in situ (in the macro-scale). Three types of boundary conditions can be applied: Uniform Strain, Uniform Stress and Periodic.

The FE discretization and the Newton-Raphson iterative procedure have been presented for solving the equilibrium in non-linear scenarios. The implementation of the three kinds of Admissible boundary conditions was discussed based on the works of [16, 17].

Finally, the chapter provides micro-scale results applying the three kinds of boundary conditions on a simple 2D example and compares the quantitative results with the theory.

Chapter 3

Feasibility of the FE2 Method

The FE2 multi-scale approach enables solving complex composite problems that cannot to be tackled with the one-scale FE method. Moreover, it provides a substantial increase in accuracy for the solution compared with other multi-scale theories based on phenomenological laws. The method however comes with massive computational costs and demands optimizations in its implementation to make it feasible.

The first section aims to identify the most computationally intensive parts of the algorithm, basing the analysis on the internal procedures carried out at both scales. The second section makes a crucial distinction (from the computational perspective) between linear and non-linear cases and their implications. The third section presents the three coupling approaches applied in the FE2 scheme; these are a trade-off between accuracy and computational cost. Finally, the fourth section proposes a Multi-Zone coupling methodology to establish the link between the macro-scale and the micro-scale to reduce the computational cost.

3.1 Computational Analysis

As it was described in the Chapter 2 the computational cost of the FE2 multi-scale method can be estimated quite accurately by taking into account the size of the problems, i.e. number of degrees of freedom, at both scales. This section gives in depth description of all the computational steps applied to each part of the algorithm. The explanation starts from the resolution of the macro-scale equations and then goes into the micro-scale kernel.

The equilibrium at both scales is achieved applying the Newton-Raphson iterative procedure summarized in Eq. 2.13. This involves two main steps: first, to build the Jacobian matrix and the Residue vector. Second, to solve the linear system of equations corresponding to that iteration. Beginning with the macro-scale problem each iteration of the Newton-Raphson can be expressed as:

$$\begin{cases} \overline{\mathbb{A}} = \sum_{\overline{\Omega}_e} \overline{\mathbb{A}}_e, \quad \overline{r} = \sum_{\overline{\Omega}_e} \overline{r}_e & \rightarrow \text{Jacobian \& Residue Assembly,} \\ \overline{du} = -\overline{\mathbb{A}}^{-1}\overline{r} & \rightarrow \text{Solver,} \\ \overline{u} = \overline{u} + \overline{du}. & \end{cases} \tag{3.1}$$

The algorithm to assembly the Jacobian matrix and the Residue vector consists in building an elemental Jacobian $\overline{\mathbb{A}}_e$ and an elemental Residue $\overline{r}_e$ for each element in the macro-scale discretization. Both of these variables require the use of the micro-scale model to compute the average stress tensor $\overline{\sigma}_{gp}$ and a tangent constitutive tensor $\overline{\mathbb{C}}_{gp}$ in every integration point of the macroscopic element. Using these variables the construction of the elemental Jacobian and the elemental Residue can be state as:

$$\overline{\mathbb{A}}_e = \sum_{gp} \overline{\mathbb{B}}^T \overline{\mathbb{C}}_{gp} \overline{\mathbb{B}}\, \overline{w}_{gp} \quad \text{and} \quad \overline{r}_e = \sum_{gp} \overline{\mathbb{B}}^T \overline{\sigma}_{gp}\, \overline{w}_{gp}. \tag{3.2}$$

As can it can be seen, to perform an iteration in the macro-scale a large amount of micro-scale problems equal to the number of macro integration points should be solved. This fact explains the dominance of the micro-scale calculation in the total computational time of the FE2 multi-scale algorithm. In reality, for a given macro-scale size, the computational cost is dominated by the micro-scale computations when the micro-scale resolution is large enough as is represented in Fig. 3.1. For very small micro-scale problems the cost is dominated by the macro-scale calculations and for large micro-scale resolution the FE2 calculation is dominated by the microscopic computations.

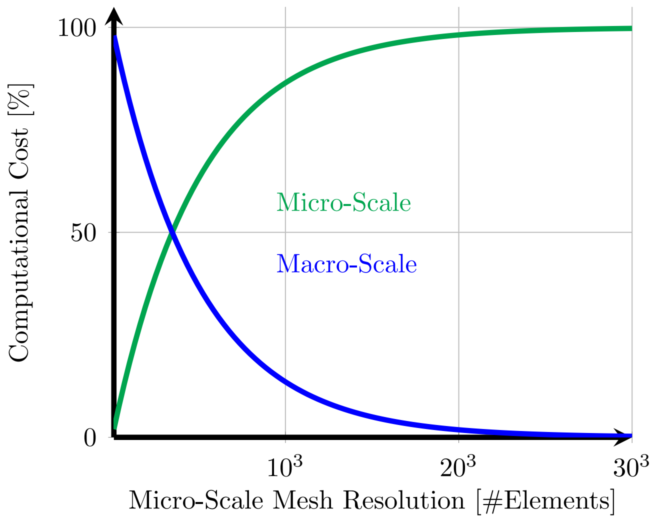


FIGURE 3.1: Theoretical and approximate contribution to the total computational cost of each of the problem when the size of the macro-scale remains fixed and the micro-scale resolution increases.

The micro-scale, being the most computational expensive part for this algorithm, it is where most of the optimizations can be done. This procedure consists in solving a large number of FE problems which are independent from each other; in consequence, this makes the parallelization more easy to be performed in the code.

In the same way, the Newton-Raphson for the micro-scale is expressed as:

$$\begin{cases} \mathbb{A} = \sum_{\Omega_e} \mathbb{A}_e, \quad r = \sum_{\Omega_e} r_e & \rightarrow \text{Jacobian \& Residue Assembly,} \\ du = -A^{-1} r & \rightarrow \text{Solver,} \\ u = u + du. & \end{cases} \tag{3.3}$$

The Jacobian $\mathbb{A}$ and the Residue vector r are assembled as:

$$\mathbb{A}_e = \sum_{gp} \mathbb{B}^T \mathbb{C}_{gp} \mathbb{B}\, w_{gp} \quad \text{and} \quad r_e = \sum_{gp} \mathbb{B}^T \sigma_{gp}\, w_{gp}. \tag{3.4}$$

This is done by using the mechanical properties of the constituent materials of the micro-structure: the stress tensor σ_{gp} and the tangent constitutive tensor $\mathbb{C}_{gp}$. These are calculated using the constitutive laws as a function of micro-scale strain ϵ. The main difference between the macro-scale and the micro-scale is the solution of the constitutive laws; in the micro-scale the material model consists of the usual constitutive laws, while in the macro-scale the micro-scale the one which acts as a constitutive law.

For a better understanding in the order of magnitude of the computing time for the Jacobian and Residue assembly against the solver Fig. 3.2 shows the computing time of both procedures performed by the Micropp code and different micro-scale mesh resolutions. The computations were done using the `benchmark-sol-ass` whose execution is detailed in the Documentation [10] and one core of an Intel 8160 CPU of one computing node of MareNostrum4 with GCC V7.1 compiler. Notice that for a micro-scale resolution of 10^3 elements the assembly represents 64% and the solver 36%. For a resolution of 50^3 this relation changes to 32% for the assembly and 68% for the solver. Finally for a resolution of 100^3 the assembly represents 21% and the solver 79%. Notice that as the micro-scale mesh resolution grows the computing time ends to be dominated mainly by the solver algorithm that in this case it is a Conjugate Gradient (CG) method.

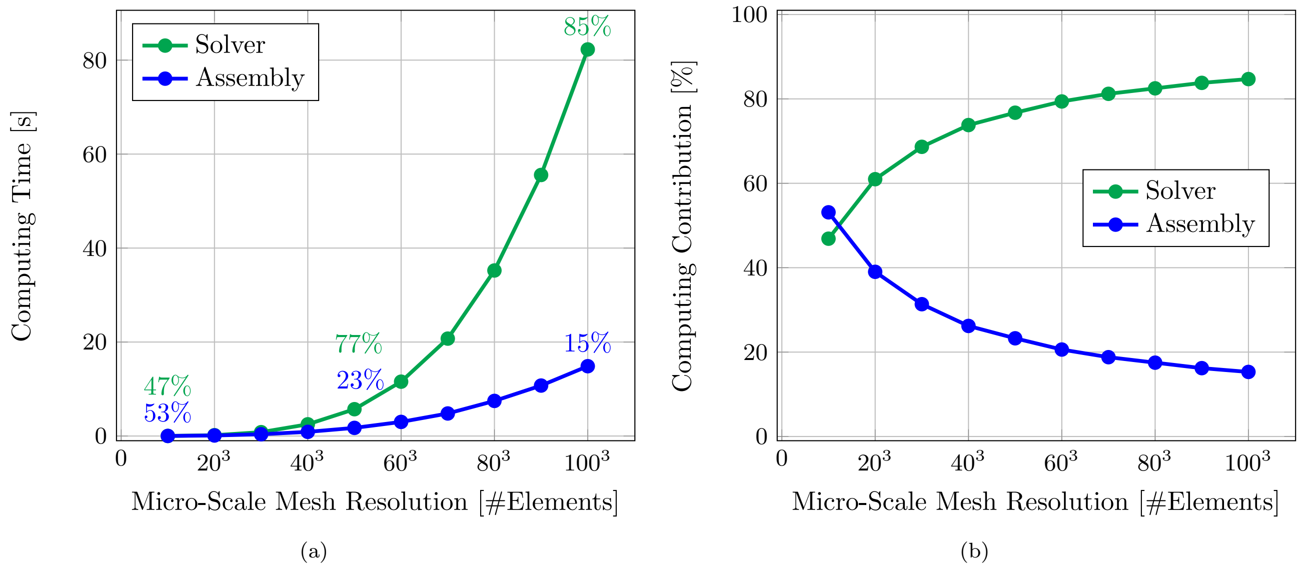


Figure 3.2: Computing time for the Jacobian matrix and the Residue vector assembly, and for the CGDP solver algorithm of the Micropp code to perform the micro-scale FE calculation in an Intel Core i7-3520M CPU 2.90 GHz.

3.2 Linear and Non-Linear Problems

Most of the engineering problems are considered non-linear and constitute the real challenge from a computational point of view. The non-linear source terms can become mainly from three sources: the geometrical contribution which is due to the change in the integration rule due to the mesh shape as the body deforms; the material contribution which basically establishes a non-linear relation between the deformations and the stresses; and the contact mechanics contribution where the deformations and stresses can exhibit abrupt changes when two surfaces enter in contact.

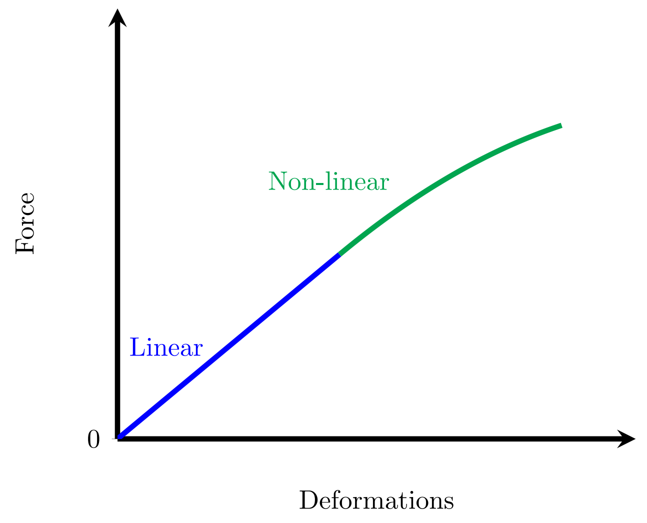


FIGURE 3.3: Typical force-deformation curve for a non-linear solid mechanics problem. During the low range deformation regime the material remains in the elastic zone but after the deformation passes a deformation limit the material enters in the non-elastic region exhibiting a non-linear response.

From an engineering point of view most of the simulations are assumed linear in the beginning and the non-linearities are modeled when certain limits in the variables' values are overpassed. For example, to study if a composite material panel breaks under deformation, it only would be necessary to do a linear simulation and check if the stress limit exceeds the critical value in the body. In contrast, if it is important to go beyond the non-linear limit and study what happens even after the material start to damage then the simulation should consider how to deal with it and generally this adds an additional computational cost, this case is displayed in Fig. 3.3. Most of the time the aircraft parts are built with some imperfections that can evolve over time and it is important to predict how they will behave. A good implementation of the algorithm should be able to handle both kind of phenomena.

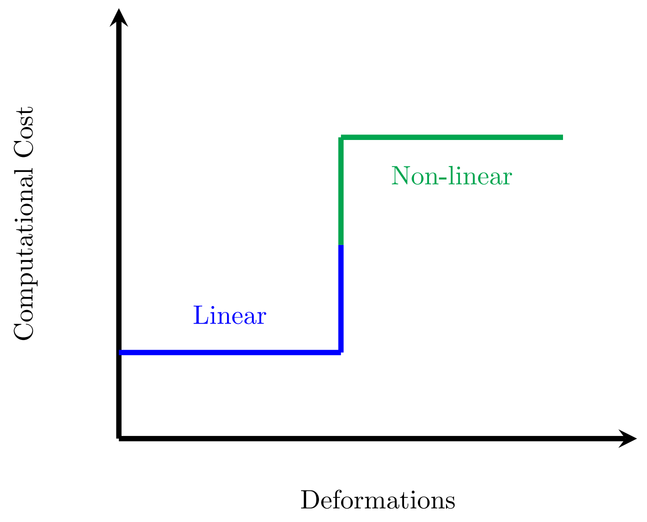


FIGURE 3.4: Increase of the computational cost of a micro-scale FE problem when it enters in the non-linear region.

From the algorithmic point of view the transition from the linear to the non-linear region in the micro-scale can be determined by the number of Newton-Raphson required to compute the solution. In this way, only one Newton-Raphson iteration is demanded for linear problems and more than one are required to be computed in the non-linear cases. For this reason the computational cost is effectively higher in non-linear problems and special considerations should be taken into account for the parallel scheme due to the load unbalance effects that can appear. The increased in the computational cost for the transition from linear to non-linear can be visualized in Fig. 3.4.

3.3 Coupling Schemes

The FE2 multi-scale algorithm presented is based in the exchange of the macro-scale variables $\overline{\epsilon}$, $\overline{\sigma}$ and $\overline{\mathbb{C}}$ between both scales to obtain a global solution. Firstly, the macro-scale problem computes the deformation tensors $\overline{\epsilon}$ at each integration point of the mesh and sends them to the micro-scale to perform the localization and solve the FE problem. Then, the average stress tensors $\overline{\sigma}$ and the tangent constitutive tensors $\overline{\mathbb{C}}$ are obtained by applying the homogenization procedure. The process is repeated for each time step and for each internal Newton-Raphson iteration of the macro-scale. The stress and the tangent constitutive tensor that come from the micro-scale are then used to calculate a new guess of the solution to achieve the equilibrium of the non-linear equilibrium.

The process of sending the average data such as the stress and tangent constitutive tensors from the micro-scale to the macro-scale is required to calculate the deformation $\overline{\sigma}(\overline{x})$ and the stress $\overline{\epsilon}(\overline{x})$ fields at the macro-scale. Some composite problems, for example those which exhibit small deformations at the macro-scale or those which do not have a complicated non-linear response, can be simplified by performing different approximations in the micro-scale. The proper use of these approximations in the micro-scale, which is the most intensive part, can lead to a substantial reduction in the computational cost without compromising on the accuracy of the algorithm. This can be considered a first level of optimization for this FE2 multi-scale algorithm that only consists in performing a right selection of the coupling between both scales. There are three possible coupling schemes that can be applied in the micro-scale: the Linear, the One-Way and the Full coupling [8, 9].

Linear Coupling

This is the most computationally affordable but the less accurate of the three coupling schemes. The approximation is based on sending back a linear response from the micro-scale to the macro-scale and avoiding computing FE problems for the integrations points. The FE computations at the micro-scale are only performed during the initialization of the code in order to obtain the linear tangent constitutive tensor $\overline{\mathbb{C}}_0$ applying Eq. 2.20. For this, 6 FE computations with the Localization and Homogenization are performed at the beginning. In this way for every iteration and for every integration point of the macro-scale the micro-scale computes the average variables as:

$$\begin{cases} \overline{\sigma} = \overline{\mathbb{C}}_0 \cdot \overline{\epsilon} & \rightarrow 0 \text{ FE computation,} \\ \overline{\mathbb{C}} = \overline{\mathbb{C}}_0 & \rightarrow 0 \text{ FE computation.} \end{cases} \tag{3.5}$$

This procedure was commonly used in the last decades when the FE2 multi-scale method was invented and the computational technology was much less advanced than what is available today. It was used to approximate the behavior of composite structures in the linear range. Logically the applicability of it is very limited and do not allow to solve more realistic and interesting problems for industrial purposes [13]. As it will be explained this procedure can continue be useful if it is combined with the others to be used in a multi-coupling scheme to solve the composite parts of the macrostructure that remain in the elastic range.

One-way Coupling

This coupling scheme is able to capture the non-linear response of the composites by incorporating the computation of an FE problem in the micro-scale. By this way the procedure can predict the behavior of the micro-structure when the macro-scale deforms beyond the elastic limits. The average macro-scale stress tensor is computed with the integral of Eq. 2.5. For this at least one non-linear FE computation with the Newton-Raphson method is demanded. On the other hand, the tangent constitutive tensor is approximated with the linear tensor that it is only computed at the beginning of the code such as in the Linear coupling scheme. At every integration point of every iteration of the macro-scale the micro-scale code computes:

$$\begin{cases} \overline{\sigma} = \int_\Omega \sigma \, dV & \rightarrow 1 \text{ FE computation,} \\ \overline{\mathbb{C}} = \overline{\mathbb{C}}_0 & \rightarrow 0 \text{ FE computation.} \end{cases} \tag{3.6}$$

This scheme is usually used for problems where the requirement is to find the linear or elastic limit of a structure. It is relatively more computationally expensive than the Linear scheme since it demands an FE computation for each integration point of the macro-scale but it is substantially more accurate and allows to predict the non-linear response of a wide variety of material mixtures. From empirical evidence during the experiments carried out in this thesis usually the converge of the macro-scale is not achieved due to the linear approximation of the tangent constitutive tensor ($\overline{\mathbb{C}} = \overline{\mathbb{C}}_0$). It was observed that this last depends on the micro-structure and the materials models used to represent the constituents.

Full Coupling

The Full coupling scheme is the more accurate of the three and also the most computationally intensive. It solves an FE problem in the macro-scale to compute the average stress using Eq. 2.5 and also applies the perturbation procedure of Eq. 2.20 solving six FE computation with Localization and Homogenization to compute the average tangent constitutive tensor $\overline{\mathbb{C}}$. In the current work this is done only if the micro-scale has passed the linear region to update the value of the tangent constitutive tensor. This optimization is done in order to save computing time for the linear problems where six Newton-Raphson operations can be avoided. Logically in the linear region the constitutive tensor of that integration point corresponds to the linear tensor ($\overline{\mathbb{C}} = \overline{\mathbb{C}}_0$). In this case the micro-scale computes:

$$\begin{cases} \overline{\sigma} = \int_\Omega \sigma \, dV & \rightarrow 1 \text{ FE computation,} \\ \overline{\mathbb{C}} = \frac{\overline{\sigma}^j - \overline{\sigma}}{\delta \overline{\epsilon}} & \rightarrow 6 \text{ FE computation (Non-Linear and 0 if it is Linear).} \end{cases} \tag{3.7}$$

This process is much more computational intensive in non-linear problems (7 FE computations for non-linear integration points) than the Linear coupling scheme (1 FE in non-linear cases). The advantage is that it is more accurate and can solve the non-converge issue of the macro-scale equations that the One-Way coupling has. This improvement happens because the tangent constitutive tensor calculated with more accuracy for the non-linear range cases than in the Linear coupling case (for linear and non-linear problems $\overline{\mathbb{C}} = \overline{\mathbb{C}}_0$) and the Newton-Raphson scheme has more possibilities to converge.

Comparison of the Coupling Schemes

To compare the results and the computational cost of the three coupling schemes, an experiment with a beam subjected to flexion is considered. Fig 3.5 shows a scheme of it. The macro-scale mesh is composed of only three elements in order to evaluate the micro-scale influence and to produce a non-uniform strain and stress fields for each of the integration points. The micro-scale mesh resolution has 25^3 elements and the micro-structure consist in a cubic cell with and spherical void in the center as it is visualized in Fig 3.5. The material of the cell is a non-linear plastic model which is described in the Appendix A, the spherical void sphere is modelled with an elastic material with a very small Young's Module. The simulation consists of 20 time steps that allows the beam to deform in a cantilever like deformation with a free and fixed ends. The calculation was perform using the coupling algorithm with Alya and Micropp which is going to be presented with more details in the Chapter 5.

Fig. 3.5 shows the plastic progression in the micro-structures corresponding to three integration points. These micro-structures are coupled with one integration point of each of the elements. Every element has a different strain this produce a different the plastic pattern inside the RVEs (represented with green color).

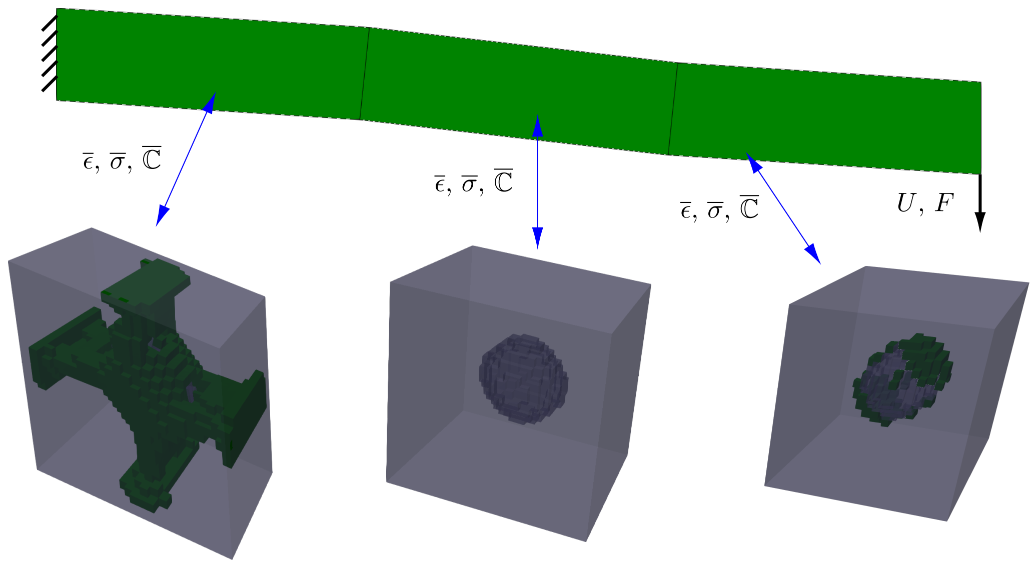
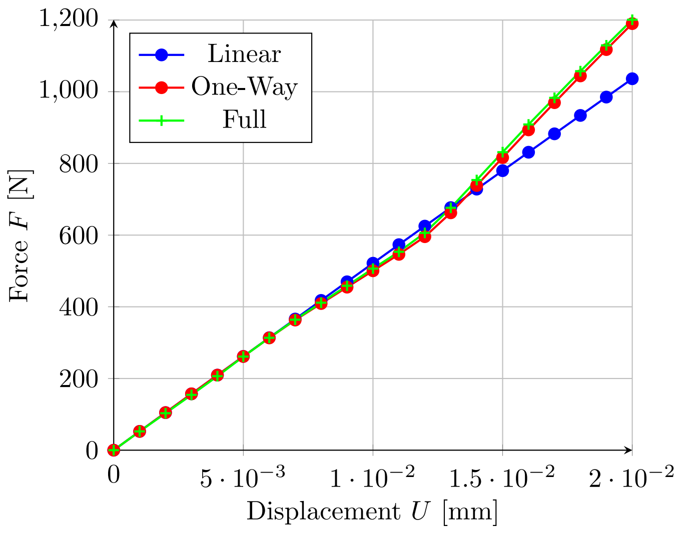

$\overline{\epsilon}$, $\overline{\sigma}$, $\overline{\mathbb{C}}$
$\overline{\epsilon}$, $\overline{\sigma}$, $\overline{\mathbb{C}}$
$\overline{\epsilon}$, $\overline{\sigma}$, $\overline{\mathbb{C}}$
U, F

FIGURE 3.5: Coupling example of a macro-scale composed of a beam with a mesh of three elements which is subjected to flexion in one of the extremes. The internal micro-structure has a spherical void inclusion and the main bulk material models the plasticity. The deformation and plasticity progression (represented in green) is shown for three integration points of the macro-structure using the Full coupling scheme.

Fig. 3.6(a) shows the Force vs. Deformation curve for the three coupling mechanisms. Notice that the force in the Full coupling scheme follows a similar behavior than the Linear and One-Way coupling schemes in the linear range. The curve of the Full scheme and the One-Way coupling start pulling apart from the Linear after a displacement of 8×10^{-3} mm when the non-linear plasticity starts appearing in the micro-scale. The Full and the One-Way curves are not strictly the same in the non-linear range because the approximations of the tangent constitutive tensor is different for both cases. In the table represented in Fig. 3.6(b) the computational cost each coupling case is outlined to clarify that the computational cost goes as Linear < One-Way < Full as it was predicted. Since the difference between the amount of FE calculations in the Linear and the Full coupling scheme is 1 to 7; it is expected that the computing time would follow a similar relation: the Full coupling would be six time more expensive than the Linear coupling. In this case the cost between these coupling schemes is approximately 1 to 1.33 as it indicated in Fig. 3.6(b). This happens because the non-linear contributions appears only in the last part of the simulation and no during all the time steps.



(a)

Coupling scheme	Linear	One-Way	Full
Computing Time [sec]	3	660	880

(b)

FIGURE 3.6: Numerical results of the non-linear beam problem shown in Fig. 3.5 and obtained with the FE2 multi-scale implementation coupling Alya and Micropp comparing the three coupling methodologies. a) Force vs. Displacement relation. b) Computing time difference.

3.4 Multi-Zone Coupling

Observations from the aforementioned sections suggest that the most accurate way of applying the FE2 multi-scale procedure is by applying the Full Coupling Scheme for all the integrations points of the macro-scale were a composite is present. This procedure gives a good approximation to the behavior of non-linear composite elements for both scales and it can be used for solving a wide variety of problems. Unfortunately this approach demands a huge computational cost since 7 FE computations should be performed in each non-linear integration point and in every iteration of the macro-scale. Moreover, some components of the macro-scale, which are made with composite are not important to be analysed with the multi-scale strategy because they remain in the linear zone during the whole simulation; then, it is not reasonable to solve a FE problem in those integration points. To solve this drawback in the computational issue, and adapt the current algorithm to a broader variety of industrial problems, a Multi-Zone Coupling and a Multi-Mesh Resolutions schemes are proposed here in order to reduce the computational cost and to preserve the accuracy.

The Multi-Zone Coupling strategy consists in pulling apart the entire domain of the macro-scale into different regions where each of them is coupled differently with the micro-scale. For example, the Full coupling scheme can be used in a region whose elements can exhibit non-linearities easily. On the other hand, the Linear coupling can be used in regions that would remain in the elastic zone during the whole simulation. With this approach the number of Full coupled integration points diminishes; and what it is crucial, the computational cost. In Fig.3.7 an example of the Multi-Zone algorithm using the three coupling schemes: Linear, One-way and Full in different regions of the macro-scale is visualized. The three regions are in contact with elements that do not represent composite parts and can be simulated with simple constitutive laws like an elastic model.

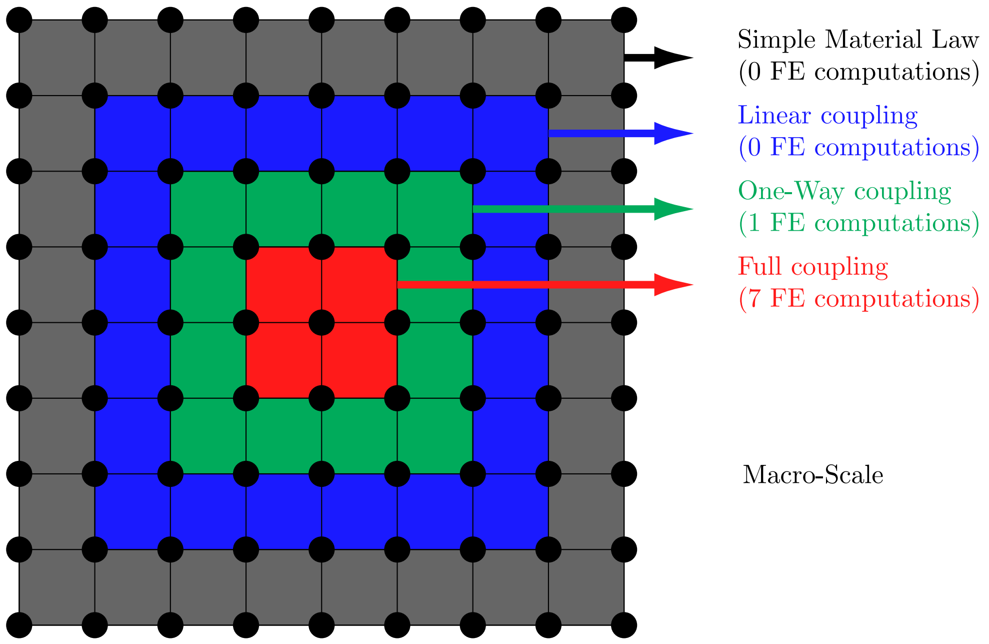


Figure 3.7: Multi-Zone Coupling strategy. The domain is separated into non-overlapping volumetric regions whose elements are coupled with the micro-scale using the different approaches: Linear, One-Way or Full coupling depending on where the non-linearities are going to appear during the simulation.

The main idea of the Multi-Zone Coupling approach is used to reduce the most possible number of elements coupled with the Full and One-Way schemes to decrease the computational cost. The approach has the drawback that it depends on the user knowledge since the areas with non-linearities should known a priori. Unfortunately, in some cases it is difficult to know the behavior of the composite by beforehand and to determine in which zones the structure enters in the non-linear region. The Multi-Zone scheme is used to solve some composite examples in Chapter 6 and the zones are explicitly determined during the initialization. More practical approaches can be developed in order to automatize this process, such as changing dynamically the coupling scheme of the integration points during different time steps of the simulation, and reducing the computational cost.

A similar approach can be applied using a Multi-Mesh Resolution for the domain. The idea is to decompose the domain in different regions and to use different micro-scale mesh resolutions for each. In this way, those integration points in the macro-scale which are exposed to large deformations and large stresses that enter in the non-linear region will use a large micro-scale mesh resolution, the other which remain in the elastic zone will used a coarse micro-scale mesh to reduce the computational cost maintaining the accuracy of the method. The algorithm can be improved further if this process is made automatically by changing the micro-scale mesh resolution dynamically. This means that a region can be modelled with a micro-scale mesh of a certain resolution, for example 25^3 elements during the first time steps of simulation; and then, when an integration point overpasses the elastic zone, the computation of that micro-scale problem changes to a more refined mesh of 50^3 elements. This approach can reduced considerably the whole computation cost exploiting the computational power for those integration points which are most computationally intensive. The strategy can be generalized to more than two mesh resolutions, for example, changing the discretization from $25^3 \rightarrow 50^3 \rightarrow 75^3 \rightarrow 100^3$ elements. This Multi-Mesh Resolution technique was not implemented in the current work and it is only limited to this proposal. A proper implementation demands a substantial amount of code development to be done.

3.5 Conclusions

The FE2 algorithm applies the Newton-Raphson procedure at both scales to achieve the equilibrium of the non-linear problems. The most time-consuming operation is the solving of all the micro-scale problems coupled with the integration points of the macro-scale. The micro-scale problems compute the average stress and tangent constitutive tensors required to solve the macro-scale governing equations. The latter means that most of the computational optimization efforts should be located on the micro-scale to reduce the computing time of the whole algorithm.

Each of the micro-scale calculation has two dominant processes to carry out the localization of the macro-strain. The first is the assembly of the Jacobian matrix and the Residue vector, and the second the solver of the linear system of equations.

Some of the most challenging engineer problems exhibit non-linear behavior caused due to the high deformations in some areas of the macrostructure. The macro-scale translates the deformations to the micro-scale making this last to overpass the elastic zone (maybe by the stress limits). This last phenomenon produces a non-linear response between the strain and the stress and increases the computational cost of the algorithm because of the additional Newton-Raphson iterations required to converge.

The coupling scheme of the integrations points of the macro-scale with the micro-scale calculations influences the computational cost of the algorithm substantially. The Full coupling scheme provides the highest level accuracy but demands the higher computational cost. On the other hand, the Linear coupling is computationally cheap but is not accurate in non-linear cases. An intermediate solution is the One-Way coupling, which is a compromise in accuracy and computational intensity between the others.

The Multi-Zone coupling approach proposed consists of discretizing the domain into several volumetric regions and to couple each of them differently according to the expected deformation during the simulation. In this way, the regions of the macrostructure subjected to large deformations are coupled with the micro-scale with the Full scheme, and those with small deformations are coupled with the Linear one. The One-Way scheme can be used for intermediate regions.

Chapter 4

The Micro-Scale Kernel

The micro-scale computations are the most computationally intensive tasks of the FE2 multi-scale method due to the massive quantity of FE problems that are required to be solved in each macro-scale iteration (one micro-scale problem per integration point of the macro-scale). This chapter discusses the main computational details of the micro-scale code Micropp developed primarily for this work. The first section explains some of the design bases of Micropp code and the main functionalities. The second section is dedicated to an analysis of the solver of linear equations at the micro-scale; a comparison of different solvers and preconditioners implemented with the PETSc library [20] are compared with the intrinsic solver of Micropp. The third section presents a sub-iterations method for dealing with abrupt macro-strain jumps at the integration points which affect the convergence of the micro-scale problems. The fourth section explains the multi-task CPU parallelization of Micropp with OpenMP. And finally, the fifth section is focused on the accelerated CPU/GPU scheme with OpenACC.

4.1 Calculation Procedures and Basic Design

Micropp is an Object-Oriented library developed in C++ and designed to be coupled with different macro-scale codes (written in C, C++ or Fortran) to create in conjunction a FE2 multi-scale application. The fundamental structure of Micropp is an object of the main Class responsible of computing, for each integration point of the macro-scale, the average constitutive laws by solving the micro-scale FE problems. It is important consider that Micropp exposes several functions that are called by the macro-scale code and Micropp plays the role of a black-box, computing average constitutive laws receiving strains $\overline{\epsilon}$ and returning stresses and tangent constitutive tensors ($\overline{\sigma}$, $\overline{\mathbb{C}}$), as it is visualized in Fig. 4.1. The most important functions that allows Micropp to be coupled with other macro-scale codes are summarized next:

- **Receiving the strains from the macro-scale**: This allows to pass the strain tensors $\overline{\epsilon}$ corresponding to the integration points of the macro-scale domain from the macro-scale to the micro-scale. This function is called in each macro-scale iteration after the displacements $\overline{u}$ is calculated (when it is possible to compute the strains $\overline{\epsilon}$ at the integration points).

- **Localize and Homogenize**: This is the computationally intensive part of the micro-scale algorithm and the complete FE2 method. This function solves one by one all the micro-scale problems applying the FEM for the problems which have being coupled with the One-Way or the Full coupling scheme. This function performs all the computations described in Chapter 2 localizing the strains $\bar{\epsilon}$ (applying the Admissible boundary conditions) and solving the FE problem in the RVE. This function is called after the strains have been passed from the macro-scale to the micro-scale.

- **Sending the average tensors to the macro-scale**: This function sends the average stress $\overline{\sigma}$ and the tangent constitutive tensor $\overline{\mathbb{C}}$, obtained during the FE computations, from the micro-scale to the macro-scale.

More details about the implementation and the precise name of the functions in the Micropp code are described in the documentation [10], and also, in the Appendix A of the present thesis.

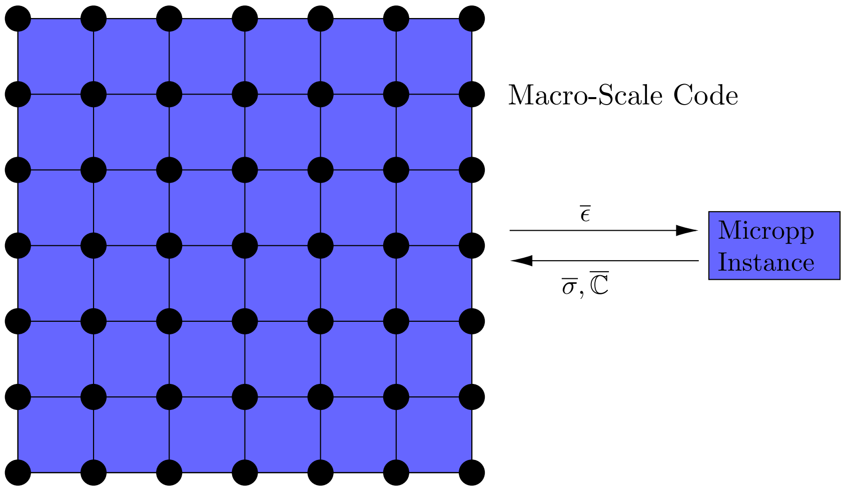


FIGURE 4.1: Coupling scheme between the Micropp code (micro-scale) and a macro-scale code.

Micropp was designed to solve the FE problems on structured grids. The latter offers some advantages in the computational optimization sense over unstructured FE codes. The internal data structures of Micropp minimize the memory data storage since the design is based on the well-known connectivity of the nodes and elements. In this way, it is possible to optimize some of the most important algorithms like: the solver of the linear system of equations, and also, the assembly of the Jacobian matrix and the Residue vector.

Structured meshes also avoid the use of data reading from the machine file systems; for unstructured meshes codes generally the conectivity of nodes and elements is generated previously with another specialized code; then, the meshes are read it by the FEM code to perform the computations. Using structured meshes, Micropp can generate different micro-structure patterns that are intrinsically programmed. This internal micro-structures can be parametrized easily in order to do parametric analyses, i.e., study the strength of a certain micro-structure when the radius of the fibers changes. Fig. 4.2 shows some of the micro-structures that Micropp can represent internally.

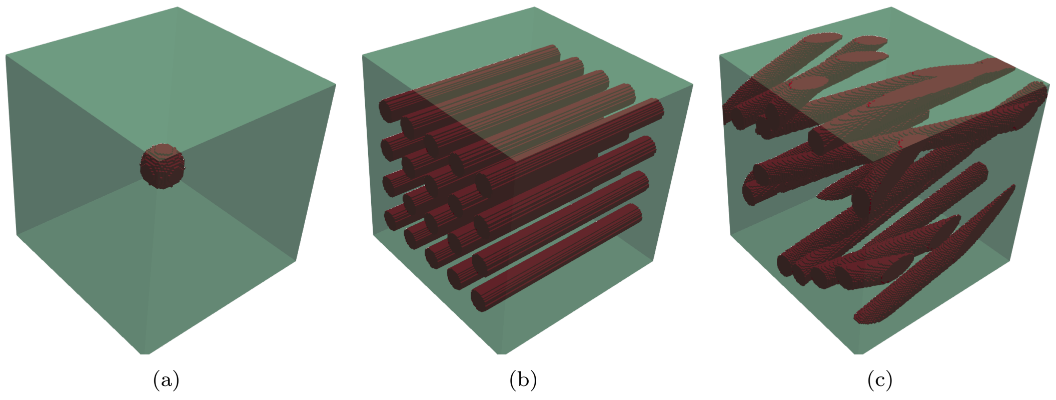


FIGURE 4.2: Micropp intrinsic micro-structures.

4.2 Solver

The solver of the linear system of equations constitutes one of the most important time consuming parts in the solution of the micro-scale problem. This is used in every Newton-Raphson iteration to obtain the approximations to the solution converging to the equilibrium. As it is shown in Fig. 3.2, a comparison of the computing time of the solver over the assembly of the Jacobian matrix and the Residue vector, is outlined. As the micro-scale mesh resolution grows the solver contribution starts dominating the computing time of the entire algorithm. The results on Fig. 3.2 correspond to a simulation performed with the Micropp code in one CPU core using the Conjugate Gradient algorithm with a Diagonal Preconditioner (CGDP). The latter is the only intrinsic solver implemented in Micropp.

In this kind of FE problems, iterative algorithms are preferred over direct methods since they demand less data memory storage for sparse matrices (like the Jacobian of the solid mechanics problem) [5]). Direct methods often generate new matrix coefficients (fill-in) that increase the data memory storage requirements, as well as the code complexity, i.e., the LU factorization algorithm.

The CGDP algorithm was implemented in Micropp since the governing equations of the this system generally lead to Symmetric Positive Definite (SPD) Jacobian matrices; this guaranties the convergence of this algorithm, making it well-suited for this solid mechanics problem.

The convergence of different iterative algorithms was measured for two solid mechanics problem: a homogeneous and a heterogeneous micro-structure. For this, the Microc code [21] was used. The latter was coupled with the PETSc library [20] for studying different algorithms to solve the linear system of equations [1]. Figs. 4.3(a) and 4.3(a) show the comparison of the convergence speed (the norm of the Residue $||r - A\,du||$) of different solver algorithms: the Conjugate Gradients (CG), Bi-Conjugate Gradients (BiCG), Bi-Conjugate Gradient with Stabilization (BCGS) and the Generalized Minimal Residue method (GMRES); all of them without preconditioning. Notice that, for the two cases, the CG algorithm presents the fastest convergence over the rest of the methods. One additional advantage of the CG algorithm, over the other iterative algorithm, is its ease to be implemented [5].

[1]Microc is a Free Software code written in C which was not completed and was mainly designed to study the performance of the PETSc algorithms in solving the linear system of equations of the micro-scale solid mechanics problems.

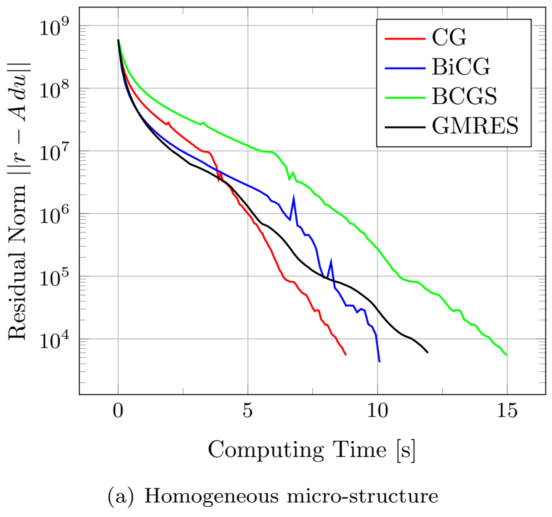
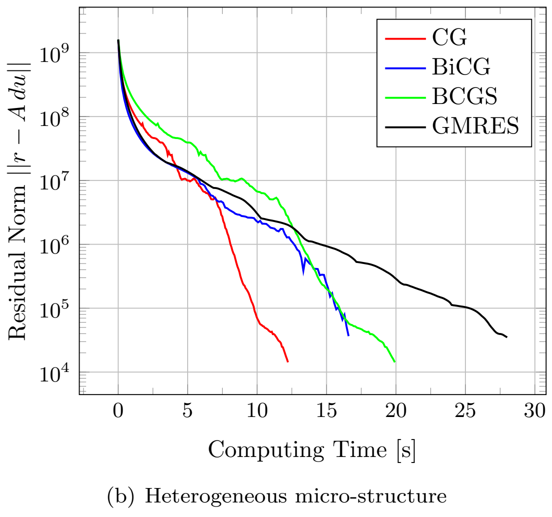
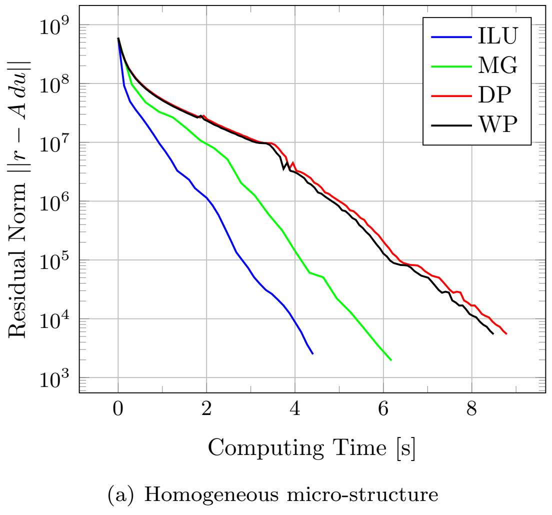


Figure 4.3: Convergence of the Residue vector norm ($||r - A\,du||$) in Microc code using different algorithms of the PETSc library.



Iterative solvers are generally used with preconditioners to increase the convergence speed by improving the condition number of the matrix. Figs. 4.4(a) and 4.4(b) show the convergence result with Microc for the same two cases of the previous example: a homogeneous and a heterogeneous micro-structure. The comparison was done without preconditioning (WP) and using the Diagonal Preconditioner (DP), the Incomplete LU factorization (ILU) and the Multi-Grid (MG); all of them implemented in the PETSc library. Notice that in all the cases the fastest convergence was achieved using the ILU, followed by the MG preconditioner. Despite the DP is not the fastest algorithm, this is used in Micropp since it makes ease its implementation and parallelization for GPU computing (with OpenACC). The speedup achieved using the DP is around $\times 2$ than without preconditioning (WP), justifying the advantages of implementing it.



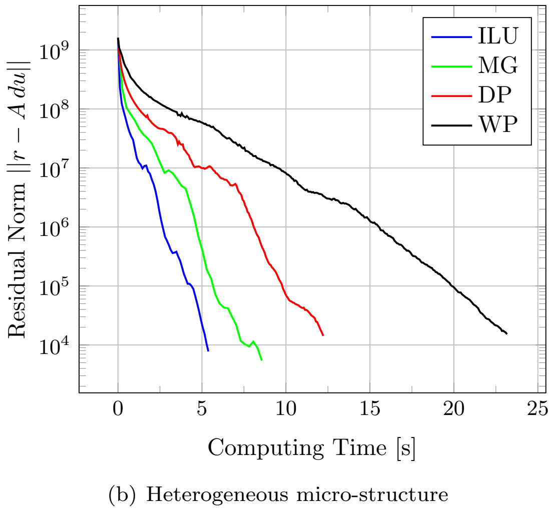


Figure 4.4: Convergence of the Residue vector norm ($||r - A\,du||$) in Microc code using the CG solver algorithm and different preconditioner of the PETSc library.

From the results shown in Figs.4.3 and 4.4 it can be concluded that as long as a problem becomes more heterogeneous, i.e., by having more difference between the material properties of the constituents, the convergence speed of the iterative solvers and preconditioners decreases. This is due to the detriment of the condition number of the matrix, requiring more amount of iterations to converge.

The implementation of the intrinsic CGPD solver in Micropp avoids the need of external libraries and enables to port the code easily to different architectures such as multi-core CPUs and GPUs. Also more optimizations can be performed since the solver is designed for this particular solid mechanics problem. The use of structured meshes allows to represent the matrix, using the ELLPACK matrix format. The latter can minimize the data memory storage that is needed with other sparse matrix formats such as the Compressed Sparse Row (CSR).

A final comparison was performed between the solver implemented in Micropp (CGPD) and the equivalent algorithm of the PETSc library (with Microc). The convergence speed is indicated in Fig. 4.5. Notice that the solver of Micropp is slightly better than the CGPD of PETSc. This is understood since PETSc uses the CSR format for the matrix representation. This format allows to solve more generic problems, i.e., problems related with unstructured grids. The data structure of the CSR is not as optimal as the ELLPACK when performing the Matrix-Vector Product (MVP). This is the fundamental and most computationally intensive component of the CG algorithm.

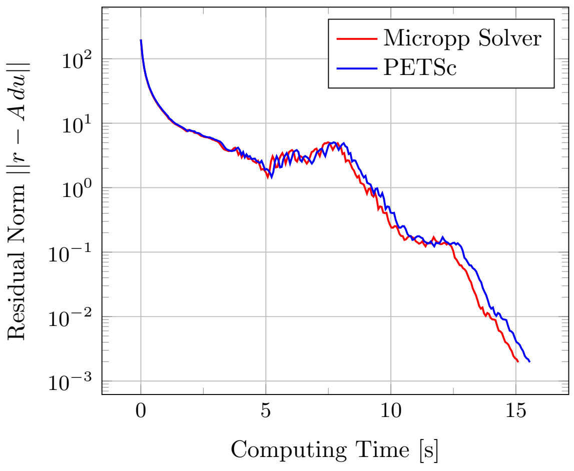


FIGURE 4.5: Comparison of the convergence speed between the CGPD solvers of Micropp and PETSc (with Microc).

4.3 Newton-Raphson Sub-Iterations

To achieve the convergence of the FE2 problem, Eqs. 2.2 : 2.3 should be solved, for this a Newton-Raphson iterative procedure is applied at both scales. Complications appear when the strain jumps $\Delta\bar{\epsilon}$ at the integration points of the macro-scale are too large. In that case, the micro-structure which is coupled with one of those integration points receives a large deformation increment and the Newton-Raphson of the micro-scale diverges. This happens because the initial guess of the solution u^{0}_{n+1} is far enough from the final solution u_{n+1} and possibly near enough to a local minimum of the minimization function (in this case the norm of the Residue vector $||r||$), as it visualized on Fig. 4.6. Also, the initial guess may be located in an unstable region of the minimization function that makes the Newton-Raphson procedure not converge.

To fix this problem a sub-iteration procedure is proposed. This consists in dividing the macro-strain jump $\Delta\bar{\epsilon}$ into smaller increments $\Delta\bar{\epsilon}^{\text{sub}}$ and applying them iteratively. The procedure is applied after detecting that the Newton-Raphson is diverging when the original strain increment $\Delta\bar{\epsilon}$ is tried to be localized (i.e., checking the value of the residue norm $||r(u)||$ or the number of iterations performed). Alg. 1 indicates this procedure.

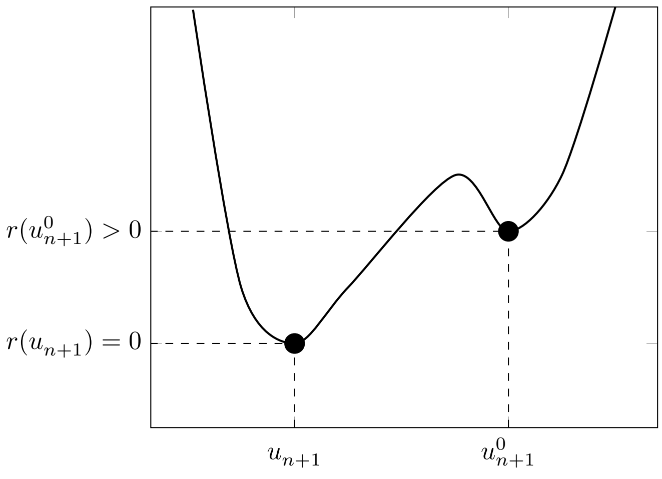


FIGURE 4.6: Representation of the problem that appear in the Newton-Raphson procedure when the guess u^0_{n+1} is far enough from the final solution u_{n+1}. This is only an illustrative example corresponding to a one-degree of freedom case.

Algorithm 1: Newton-Raphson sub-iteration procedure.

```
Newton-Raphson($\bar{\epsilon}^k$)
if micro-scale has not converged then
    $\Delta\bar{\epsilon}^{\text{sub}} = \left(\bar{\epsilon}^k - \bar{\epsilon}^{k-1}\right) / N^{\text{sub}}$
    $\bar{\epsilon}^k = \bar{\epsilon}^{k-1}$
    for $i = 1$ to $N^{\text{sub}}$ do
        $\bar{\epsilon}^k = \bar{\epsilon}^k + \Delta\bar{\epsilon}^{\text{sub}}$
        Newton-Raphson($\bar{\epsilon}^k$)
    end
end
```

```c
#pragma omp parallel for schedule(dynamic ,1)
for (int igp = 0; igp < ngp; ++igp) {

        // Newton-Raphson applied to the integration point <igp>.

}
```

ALGORITHM 4.1: Use of OpenMP in Micropp to taskify each FE problem.

4.4 CPU Acceleration

Micropp uses OpenMP to accelerate the computations of the micro-scale problems in multi-core CPUs. The strategy consists in making that each OpenMP thread solves one micro-scale problem with it is own data memory resources. In other words each threads computes the Newton-Raphson procedure with its own Jacobian matrix and Residue vector and the rest of the memory needed. This is the strategy applied since the micro-scale problems are completely independent from each other and no atomic operations are required, making the parallelization procedure straightforward to be implemented. Alg. 4.1 shows the parallelization in Micropp with the OpenMP for-loop directive to solve the list of micro-scale problems.

The parallel performance of this strategy was measured in an Intel Xeon Platinum 8160 CPU (MareNostrum4 supercomputer [22]) with 48 physical cores. The simulation consists in the computation of 480 Micropp problems with a micro-scale mesh resolution of 30^3 elements (two time steps). The results are visualized in Fig. 4.7. Notice that parallel performance is still acceptable under the 48 threads execution. With more of 48 OpenMP threads there is no improvements. This happens when the number of threads equalizes the number of physical cores in the node.

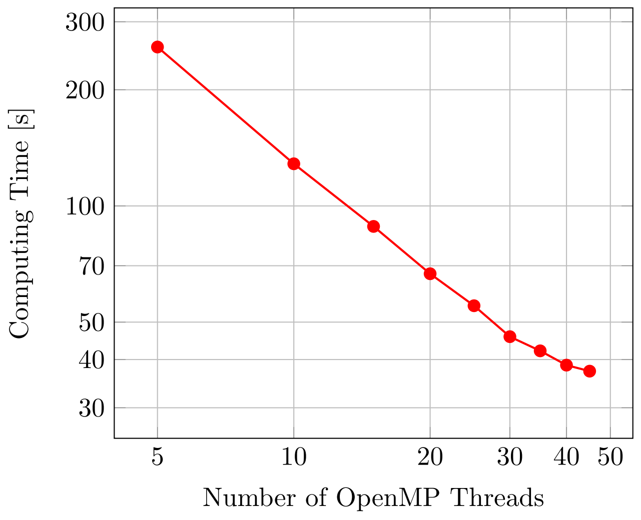


FIGURE 4.7: Parallel performance using OpenMP for solving 480 micro-scale problems. Architecture: Intel Xeon Platinum 8160 CPU belonging to MareNostrum4 supercomputer [22] (48 physical cores).

4.5 CPU/GPU Acceleration

The main distinction of Micropp is the GPU parallelization for the micro-structure calculations. GPUs can accelerate substantially the execution of those programs that have a considerable amount of parallel operations. A perfect example is the MVP and the Jacobian and/or Residue assembly that are very known and applied in the FEM. Unfortunately, the whole execution of the code cannot be done on the GPU; generally the programs are executed in an hybrid way that combines a part on the CPU and another part on the GPU depending on the operations that are performed. Generally some parts are better to be done in the CPU rather than the GPU and an optimal of performance can be observed with this hybrid execution.

For Micropp, the CGDP solver and part of the assembly of the Jacobian matrix and the Residue vector, were ported to GPUs. Alg. 4.2 shows for example the parallelization with OpenACC of the MVP operation in the Micropp solver. These steps together demand almost 98% of the total time of the micro-scale calculations for every macro-scale integration point. The porting the code to GPU was done with OpenACC. This allows to convert a normal code written originally for a CPU into an accelerated code for GPU. The procedure with OpenACC consists in adding pragma directives inside the the code, like the for-loops, that can be parallelized.

Currently the OpenACC standard is being supported mainly by the PGI compilers (C, C++ and Fortran) but it is being extended to other compilers such as GCC. OpenACC has the main advantage of not being strongly invasive for the code, i.e., it almost does not modify the structure of the program. On the other hand, the approach of using OpenACC allows to maintain the portability of the code for different GPU devices without having to modify the code depending the machine where the code is going runs. To do this, the compiler checks the GPU device in

```
void ell_mvp_acc(const ell_matrix *m, const double *x, double *y)
{
#pragma acc parallel loop present(m[:1])
        for (int i = 0; i < m->nrow; i++) {
                double tmp = 0;
                const int ix = i * m->nnz;
#pragma acc loop vector
                for (int j = 0; j < m->nnz; j++){
                        tmp += m->vals[ix + j] * x[m->cols[ix + j]];
                }
                y[i] = tmp;
        }
}
```

ALGORITHM 4.2: Example of OpenACC programming model in the MVP operation. This is a simplification of the real function of the official repository [10].

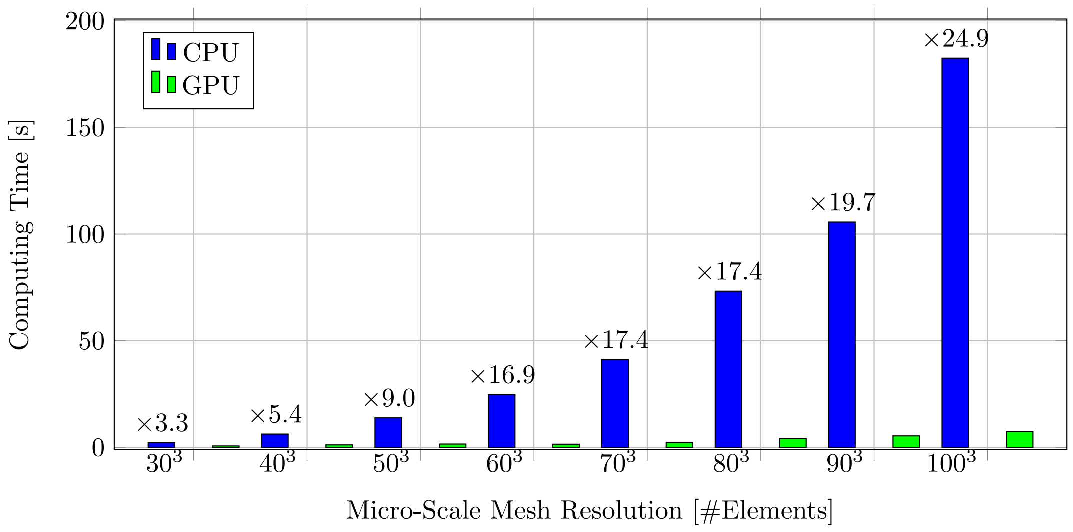


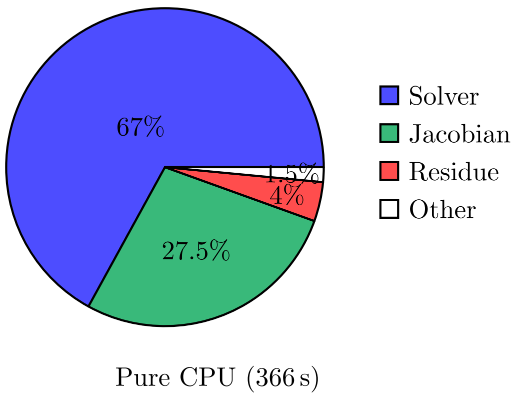
FIGURE 4.8: Computing time of the micro-scale for different mesh resolutions (`benchmark-cpu-gpu` [10]). Comparison between one CPU IBM Power9 CPU core with an Nvidia V100 GPU (CTE-POWER cluster).

which the code is going to run and performs machine-specific optimizations. Probably this is the main advantage of OpenACC respect to CUDA code. CUDA code generally is more efficiency than an OpenACC code. OpenACC can be used to do a proof of concept to see how efficient the code can run on GPUs without refactoring entirely a code. This can be a previous step before deciding if the code can be ported to CUDA.

The performance of the CPU/GPU accelerated version is compared over the pure CPU execution. Fig. 4.8 shows the computing time of one Newton-Raphson iteration for the micro-scale problem evaluating different mesh resolutions. The calculations were performed between in an IBM Power9 CPU core and an NVidia V100 Tesla GPU belonging to the CTE-POWER cluster [24]. For this a benchmark (`benchmark-cpu-gpu`) of Micropp was used. The execution description of it can be found in the documentation of the code [10]. The Newton-Raphson iteration includes the assembly of the Jacobian matrix, two times the assembly of the Residue vector (the second assembly is done for checking the that the final norm is near zero after the solver) and the CGDP solver part. The experiments have demonstrated the accelerated CPU/GPU version of the code is much more efficient than the pure CPU one. Notice that the speedup increases as the micro-scale mesh resolution grows.

In Fig. 4.9, the percentages of the computing time of the Micropp intrinsic functions are indicated for the pure CPU version of the code and for the CPU/GPU acelerated one. The results correspond to the executions of a

Micropp benchmark (`benchmark-cpu-gpu` [10]) in the CTE-POWER cluster [24]. Notice that, for the pure CPU case, most of the computing time is spent in the solver part followed by the assembly of the Jacobian matrix. On the hand, in the accelerated CPU/GPU version, most of the time is concentrated in the CPU part of the Jacobian matrix assembly. Also, 53% of the computing time is still spent in the CPU part of the code.



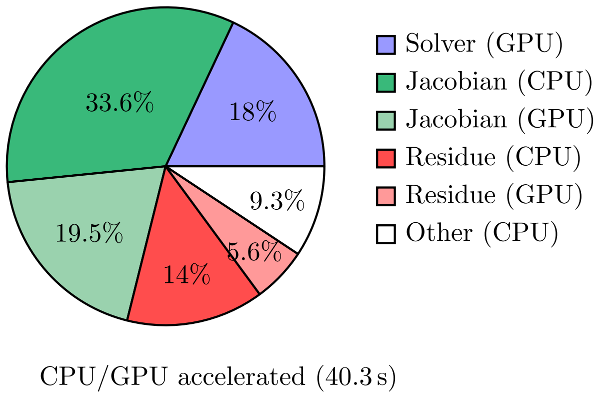


FIGURE 4.9: Percentage of the computing time spent in the Micropp intrisic functions: the solver, the assembly of the Jacobian matrix and the Residue vector.

4.6 Conclusions

This chapter has explained the most important procedures carried out by the micro-scale code Micropp. In the first part, the coupling strategy of Micropp with the macro-scale code was presented. The most important functions called from the macro-scale to control the whole FE2 procedure were described.

Different linear solvers and preconditioners implemented in the PETSc library have been compared with the intrinsic CGDP algorithm implemented in Micropp. The CGPD proved to be a well-suite combination of solver and preconditioner for this solid mechanics in comparison to the BiCG, the BCGS, and the GMRES solvers, as well as for the ILU and the MG preconditioner. An important advantage of the CGDP algorithm is its ease to be implemented and parallelized for GPUs with OpenACC.

A sub-iteration method for dealing with abrupt strain-jumps at the macro-scale integration points was presented. This method basically decomposes the abrupt jump into smaller increments, and performs several standard Newton-Raphson iterations for computing all the small increments.

The multi-core CPU parallelization (with OpenMP) and the GPU acceleration (with OpenACC) of Micropp were explained. In the first, each OpenMP task operates over an entire micro-scale FE problem; while in the second the main computational intensive operations are ported to GPU in order to accelerate the computation of each micro-scale problem.

Chapter 5

Implementation for Heterogeneous Architectures

This chapter explains the specific details of the parallel strategy used for the FE2 application and how the code runs in distributed memory architectures with GPU capabilities. The first section explains the most crucial details of the coupling between the MPI code Alya (macro-scale) and the Micropp code (micro-scale). Then, the second section describes the GPU overloading technique and the improvements in the parallel scalability achieved. Finally, the third section exposes a load balance strategy to increase the parallel performance of the algorithm when the Multi-Zone coupling is used based on weighted partition methods.

5.1 Macro-MPI and Micro-X Strategy

Parallel computing is one of the most used techniques for dealing with large computational problems. The reason is that it allows to decompose very large problems into smaller ones that require less memory and are solved faster. The scope of this FE2 application is in solving problems larger than 10K elements at the macro-scale and 100^3 elements at the micro-scale. A simple estimation of the order of magnitude of this kind of problems gives a total of $10\text{K} \times 8 \times 100^3 \approx 10^{11}$ degrees of freedom (the "8" is since there are 8 integration points per macro-scale element); clearly, distributed computing methodologies are mandatory.

The strategy proposed for solving the FE2 problem is based on coupling the MPI-based code Alya [23] (macro-scale) with the accelerated code Micropp (micro-scale). Each MPI process of Alya (responsible of computing one sub-domain) is coupled with Micropp, this is visualized in Fig. 5.1. The micro-scale computations in the RVEs are accelerated using the techniques presented in Chapter 4: the multi-core CPU acceleration (OpenMP) or the hybrid CPU/GPU scheme (OpenACC) [1].

Applying domain partition at the micro-scale with MPI, and keep the coupling scheme, would require to handle the MPI communicators for both codes, this would increase the complexity of the application. Also more standards

[1] In the name "Micro-X", "X" is the acceleration strategy used.

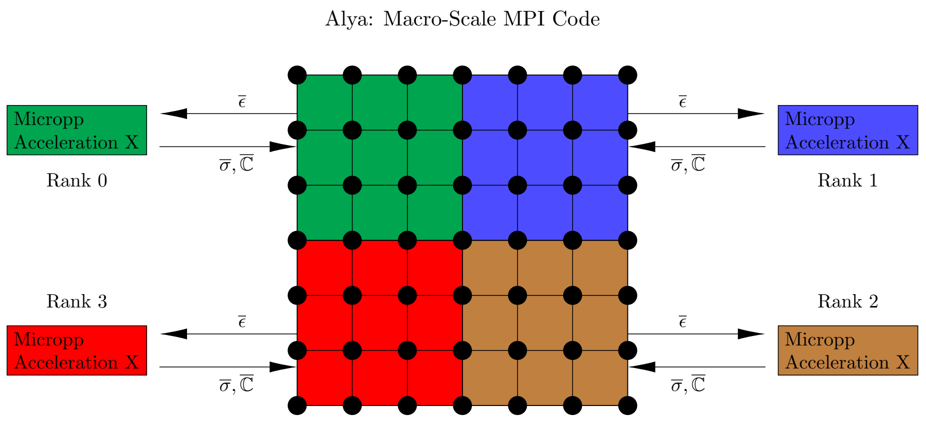


FIGURE 5.1: Parallel strategy used coupling the MPI-based code Alya (macro-scale) and Micropp (micro-scale).

be fulfill by both codes in order to make them MPI compatible, this can affect negatively the maintainability of the code.

The direct disadvantage of not using domain partition methods at the micro-scale is that it is not possible to solve large problems at that scale. In this work Micropp was designed to solve micro-scale problems up to 100^3 elements (this is enough for a broad variety of composite problems). This constrain in the Micropp code design, has motivated to gain acceleration and efficiency through the strategies presented, in particular, the CPU/GPU acceleration.

5.2 Multi-GPU and Overloading

Some architectures are built today with several multi-core CPU nodes connected with one or more GPUs in order to accelerate the applications. This means that each thread, which is running in a different core of the CPU, can execute part of the program in the GPUs connected to the node. This FE2 application exploits this property since Micropp uses the MPI-task identification number (rank), provided by Alya, to select the GPU device to accelerate the program. In Micropp the GPU device is selected using the OpenACC directives as shown in Alg. 5.1.

```
#ifdef _OPENACC
        int acc_num_gpus = acc_get_num_devices(acc_device_nvidia);
        gpu_id = mpi_rank % acc_num_gpus;
        acc_set_device_num(gpu_id, acc_device_nvidia);
#endif
```

ALGORITHM 5.1: Selection of the GPU device in Micropp.

For example, if the MPI ranks 0, 1, 2 and 3 are being executed in a node containing 4 GPUs, each GPU receives one different instance of Micropp. If the number of MPI tasks executed in the node is more than the number GPU devices then the GPU Overloading takes place. In this case, two threads use the same GPU for doing their computations (Alg. 5.1). If the MPI ranks 0, 1, 2, 3, 4, 5, 6 and 7 are executed on a node, ranks 0 and 4 use GPU device number 0, ranks 1 and 5 use the device 1, etc., as it is visualized in Fig. 5.2.

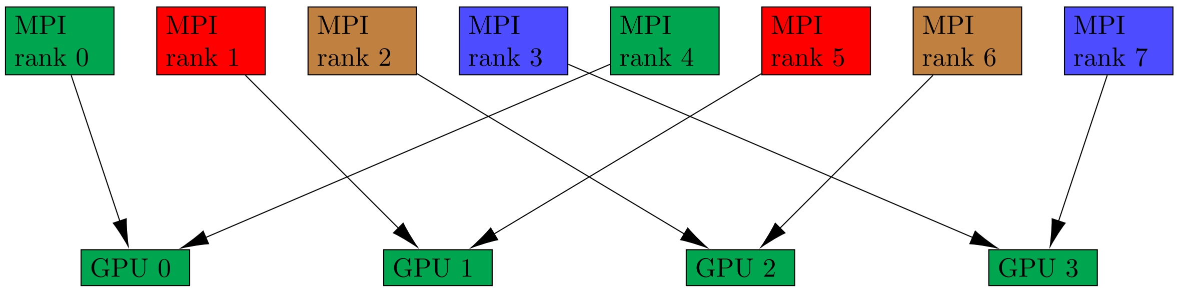


FIGURE 5.2: Representation of the GPU overloading in a multi-GPU machine.

There is a maximum number of tasks that can be executed simulaneously on a GPU. This limitation is due to the memory resources of the device. As the micro-scale resolution grows, fewer problems can be executed simultaneosly on the same GPU. For instance, Tab. 5.1 summarizes the number of Micropp instances that can be executed at the same time in one V100 Nvidia GPU of the CTE-POWER cluster [24] for different micro-scale mesh resolutions. These numbers were calculated by trial and error checking increasing the number of micro-scale problems until the program have crashed.

Size [# Elements]	CPU [MPI Tasks/Node]	GPU [MPI Tasks/Node]
25^3	40	40
50^3	40	36
75^3	40	12
100^3	20	8

TABLE 5.1: Maximum number of Micropp tasks that can be executed simultaneously on a node of the CTE-POWER cluster [24] (4 V100 Nvidia GPUs per node).

To measure the scaling of the GPU overloading an experiment running up to 40 instances of Micropp was performed in one node of the CTE-POWER cluster [24] (4 V100 Nvidia GPUs per node). The program solves 100 micro-scale FE problems with a mesh resolution of 50^3 elements (one time step). As the number of MPI tasks increases the program divides and distributes the total amount of problems across the tasks equally. Fig. 5.3 shows the speedup achieved with the GPU overloading technique. Notice that for the simulation of 4 MPI tasks each GPU is solving one micro-scale problem at the time, for 8 MPI tasks each GPU solves two problems, and so on. The scaling is still good up to 20 MPI tasks (5 problems of 50^3 solve at the same time in each GPU).

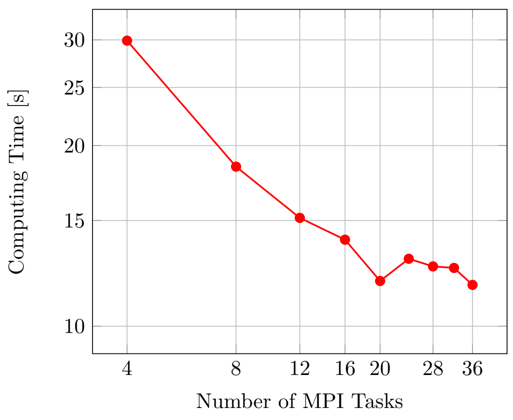


FIGURE 5.3: GPU overloading with Micropp code solving 100 micro-scale FE problems of 50^3 elements (one time step) in a computing node of the CTE-POWER cluster [24] (4 V100 Nvidia GPUs per node).

5.3 Load Balance with Weighted Partitioning

In order to reduce the computational cost of the entire FE2 calculation, the Multi-Zone coupling approach (explained in Chapter 3) is proposed. The method consists in dividing the macro-scale domain in regions depending on the deformation field $\bar{\epsilon}$ (i.e., regions with larger deformations would be coupled using the Full approach while the regions with smaller deformations would use the Linear approach). Each region sends a signal to Micropp specifying the calculation type that should be performed in the micro-scale (Linear, One-Way or Full). The strategy is used to reduce as low as possible the number of integration points coupled with the Full approach which are the most computationally demanding (maintaining at the same time the accuracy of the algorithm).

The Multi-Zone strategy reduces the total computational cost of the FE2 algorithm but the parallel performance can be deteriorated if no weighing technique is used. Alya applies a domain partition method that decomposes the domain into non-overlapping sub-domains during the initialization. Then, each MPI task performs the FE computations over one of these sub-domains (static approach). If a domain partition does not take into account the computational cost difference in the elements of the mesh (due to the coupling with the micro-scale), a high load unbalance can appear and this would be translated in a poor parallel scalability. Fig. 5.4 outlines an example of two different domain partitions of a macro-structure that implements the Multi-Zone approach with Linear, One-Way and Full coupling approaches. Fig. 5.4(a) shows a balanced partition (which is a desirable) and Fig. 5.4(b) indicates an unbalance partition (not desirable).

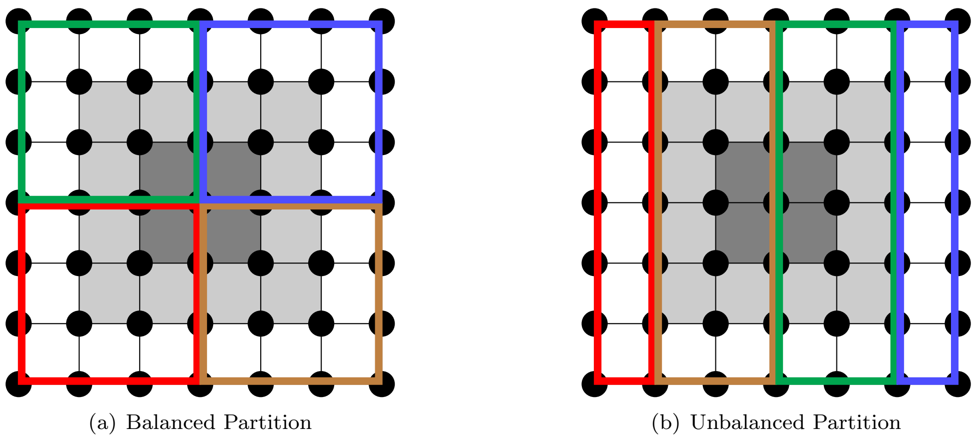

(a) Balanced Partition (b) Unbalanced Partition

FIGURE 5.4: Example of domain partitions of a macro-scale using the Multi-Zone coupling approach. The decomposition of the domain consists of three regions that apply the Linear (white), the One-Way (light grey) and the Full (dark grey) coupling.

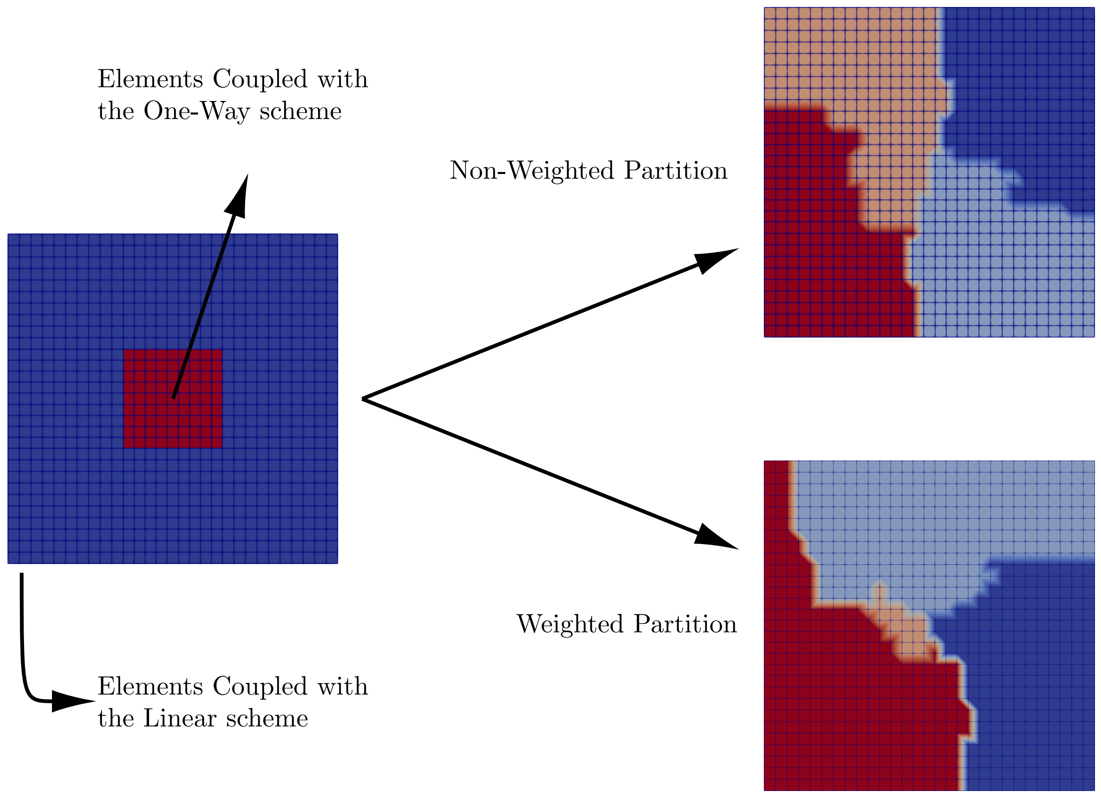
Alya uses Metis library [25] to perform the domain partition. Metis implements weighting partition algorithms that can be used to increase the load balance of the FE2 application. The method requires to set the weights for each element in the FE mesh, in this case, Alya would set the same weight for all the elements in a certain region i. A possible formula for calculating the weight for the partition of a region i is:

$$w_i = \frac{\#N_i C_i}{\sum_j \#N_j C_j}. \tag{5.1}$$

where C_i is the computational cost of one micro-scale problem of a region i and N_i is the number of elements there. Unfortunately the cost of all the integrations points is not the same (even they use the same coupling scheme:

Linear, One-Way or Full) since some of them can enter in the non-linear zone and others not. This is always a problem with static load balance algorithm which takes a domain partition decision only at the beginning of the code (not dynamically).

To quantify the improvements using the weighted partition with Alya and Micropp an experiment was carried out with a macro-scale mesh of 1.7K elements and a micro-scale resolution of 3^3 elements. For the simulation 4 MPI processes were used to decompose the domain of the macro-scale as it is shown in Fig. 5.5. The domain is divided in two regions, the central square has 12% of the total elements and it is coupled with the One-Way approach. The other 88% of the elements are coupled with Linear approach. Fig. 5.5 the domain partitions with the weighted approach and with a normal partition (without weighting). The result shows that for the weighting approach the computing time of this problem took 105 s and 177 s with the normal partition, this is a speedup of $\times 1.7$ with the weighted partition.



	Non-Weighted	Weighted	Speedup
Calculation Time	177 s	105 s	$\times 1.7$

Figure 5.5: Example with 4 MPI tasks, 1.7K elements at the macro-scale (12% One-Way coupled and 88% Linearly coupled) and a micro-scale resolution of 3^3 elements. Comparison of the computational time using the Weighted and the Non-Weighted domain partition of Metis which is called from Alya to solve the FE2 multi-scale problem.

5.4 Conclusions

This chapter has presented the parallel strategy used for the FE2 multi-scale application. The strategy consists in coupling a macro-scale code capable of doing domain partition at the macro-scale (MPI-based) with an accelerated micro-scale code. In this case Alya code is used for the macro-scale and Micropp is used for the micro-scale. This implements CPU acceleration through OpenMP and CPU/GPU acceleration with OpenACC.

The chapter has also explained the Multi-GPU and the GPU overloading of the FE2 application code. This technique uses MPI in combination to OpenACC directives for the GPU device selection for executing the micro-scale instance. Results using this technique have been outlined showing a substantial speedup can be achieved.

Finally, a load balance based on a weighted partition strategy has been proposed to solve the computational work unbalance introduced by the Multi-Zone coupling approach. The method consists in performing domain weighting partitions in order to distribute in a more uniform way the computational load across the MPI processes.

Chapter 6

Industrial Applications Solved in Hybrid CPU/GPU Machines

The objective of this Chapter is to demonstrate that the FE2 multi-scale implementation developed in this work is well suited for industrial applications. Three different aeronautical problems are addressed here. The problems are solved with pure CPU execution and with the CPU/GPU scheme to make comparisons between both strategies. Moreover, strong scaling experiments with the CPU/GPU scheme are done to demonstrate the parallel efficiency of the code.

The first section addresses a compression of an aircraft fuselage panel while the second and third sections show the simulations of the impact of a hailstone on different components of the aircraft. The examples apply the Multi-Zone scheme coupling different regions with the Linear, One-Way, and Full approaches; the weighted partitioning is also applied for increasing the load balance in the parallel executions.

The simulations and the results presented were computed using the CTE-POWER cluster [24]. This cluster holds 50 computing nodes with 2 IBM Power9 8335-GTH CPUs (40 CPU cores) and 4 Nvidia V100 GPUs, with 16GB of memory. For this work up to 40 computing nodes were used (a total of 1600 CPU cores and 160 GPUs).

The FE meshes of the aircraft structures were obtained thanks to the SHERLOC project [26]. These were built primarily for testing phenomenological models and cohesive laws [27]. In this Chapter, these are used to test the efficiency of the FE2 application in a HPC environment, which is the main scope of the thesis.

6.1 Fuselage Panel Subjected to Compression

The example intends to predict the behavior of a composite material panel (belonging to an aircraft fuselage) which is subjected to a compression load. Fig. 6.1 helps to describe this problem. On the top of it, a picture of the manufacturing process of an A350 XWB fuselage [28] is shown. Also, next to it, an augmented image of the micro-structure (carbon fiber in layers directed at 0°, 90°, 0°, etc.) of the composite [29] is presented. It is known that this kind of microstructural pattern provides the demanded strength to the structure, allowing to resist the inward pressure and the external forces (of the air) that are exerted during normal working conditions [1]. This micro-structure pattern does not correspond to the real Airbus fuselage; in this case it is used to validate the capabilities of the FE2 implementation. On the bottom of Fig. 6.1 the FE models (meshes) of both scales are outlined; the macro-scale solved with Alya code (MPI) and the micro-scale modeled with Micropp code (accelerated with CPU/GPU).

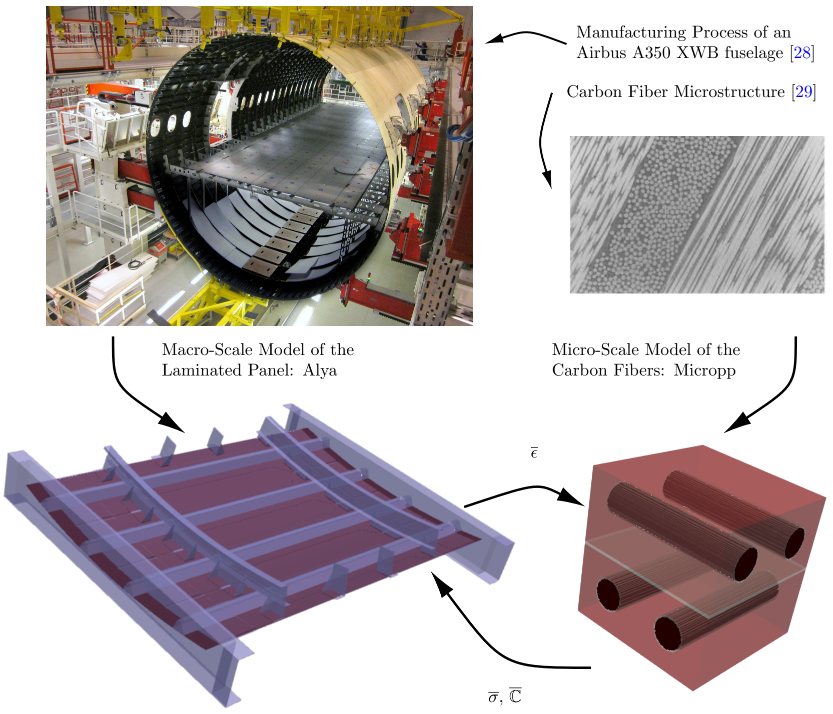

Manufacturing Process of an Airbus A350 XWB fuselage [28]
Carbon Fiber Microstructure [29]
Macro-Scale Model of the Laminated Panel: Alya
Micro-Scale Model of the Carbon Fibers: Micropp
$\bar{\epsilon}$
$\bar{\sigma}, \bar{\mathbb{C}}$

Figure 6.1: On the top, the manufacturing process of an A350 XWB fuselage [28] and an augmented image of a carbon fiber micro-structure [29] (this does not correspond to the real fuselage). On the bottom, the macro-scale modelled with Alya and the micro-scale solved with Micropp.

The macro-scale discretization consists of a hybrid FE mesh with a resolution of 140K elements that combines hexahedral elements (eight nodes and six faces) with prismatic elements (six nodes and five faces). The domain is decomposed into three main regions. At the bottom of Fig. 6.1, the red part (composed of a total of 131K elements) has 122K elements which are coupled with Micropp using the Linear scheme, the other 9K elements are coupled using the One-Way scheme. The latter group (those with the One-Way coupling) represents the more important computational issue of the algorithm since they are the ones that activate the FE computations at the micro-scale. On the other hand, the blue region in Fig. 6.1 consists of 9K elements that model the aluminum skeleton of the fuselage. The material law is intrinsic of the Alya code and corresponds to an elastic isotropic material.

Tab. 6.1 summarizes the details of the meshes used in both scales. For the micro-scale, several discretizations were used in the simulation (5^3, 25^3, 50^3, 75^3 and 100^3 elements) primarily to analyze the influence of this parameter in the computational performance. Changing the micro-scale mesh resolution is straight forward in this FE2 application since the Micropp code works using structured grids and it can create the data structures for storing the mesh during its initialization. The macro-scale Alya code is responsible for sending the micro-scale resolution to Micropp during its initialization (it has to send the number of elements across each axis of the RVE cell: n_x, n_y and n_z).

	Macro-Scale	Micro-Scale
Size [# Elements]	140K (Total) = 122K, weight = 1.0 (Linear Coupling → Micropp) 9K, weight = 1000.0 (One-Way Coupling (FE) → Micropp) 9K, weight = 1.0 (Aluminium Model → Alya)	5^3 25^3 50^3 75^3 100^3

TABLE 6.1: Details of the domain discretization used at both scales.

Fig. 6.2 displays the boundary condition settings for this problem. They consist of a constant vertical velocity V_y at the top surface of the panel and zero displacement condition at the bottom surface.

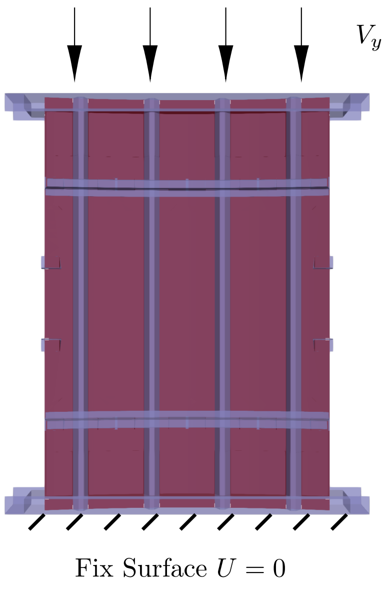


FIGURE 6.2: Boundary conditions applied at the macro-scale.

The material models used for the constituents of the micro-scale consist of an isotropic elastic type [30], for the epoxy resin, and a damage with hardening [33] for the fibers (see the bottom of Fig. 6.1). The elastic has a Young's Modulus of $E = 3.0 \times 10^7\,\mathrm{N/m^2}$ and a Poisson Ratio of $\nu = 0.25$. The carbon fibers have a Young's Modulus of $E = 3.0 \times 10^8\,\mathrm{N/m^2}$, a Poisson Ratio of $\nu = 0.25$ and a Yield Stress of $\sigma_Y = 1.0 \times 10^4\,\mathrm{N/m^2}$. Micropp includes a benchmark (`benchmark-mic-1`) as part of the Source Code [10] that runs in standalone mode (without coupling) for different micro-scale mesh resolutions and allows to evaluate the behavior of this RVE cell [10]. In addition Appendix A includes a simplified formulation of the constitutives laws that are used in this example.

Mechanical Results

Fig. 6.3 shows the stress field after one time step in the macro-scale and in two micro-scale cells that are One-Way coupled with two different integration points of Alya code. The micro-structure resolution used for this simulation has 100^3 elements.

Displaying the RVE cells (such as in Fig. 6.3) is a straightforward task since Alya contains a tool called Witness Points. For this feature, the user provides a coordinate in the macro-scale (x, y, z) and Alya searches the elements where the point belongs. Finally, Alya identifies one of the integration points inside the elements, finding its global ID, and calls a Micropp function for writing the VTU format file with the deformations and stresses of that particular RVE cell.

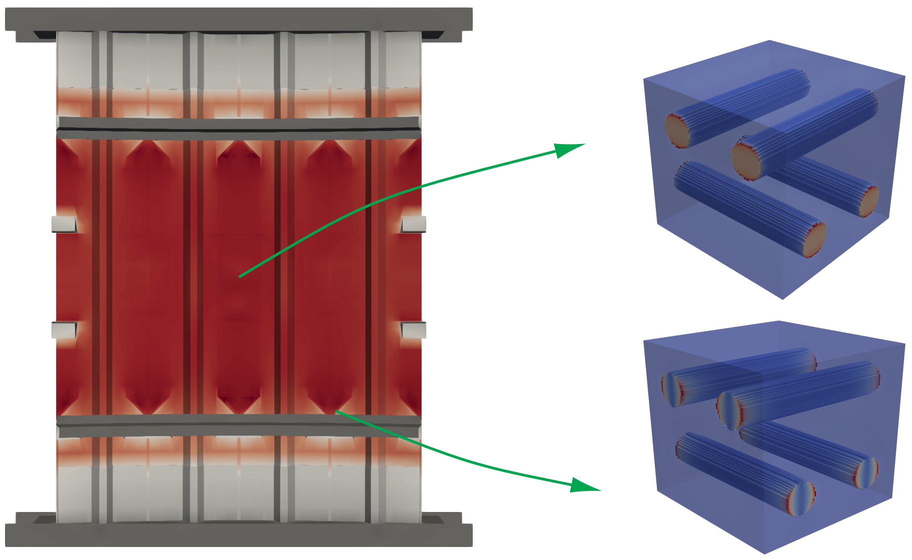


FIGURE 6.3: Stress field after one time step in the macro-scale and in two deformed RVE cells that correspond to two different integrations points of the panel.

Fig. 6.4 shows the results of a simulation after 26 time steps using a micro-scale resolution of 5^3 elements. In Fig. 6.4(a) the macro-scale elements which entered in the non-linear region are visualized. Fig. 6.4(b) displays the computational cost distribution in the macro-scale. The latter is computed as the total number of CGDP solver iterations of the Newton-Raphson process for solving the FE problem. Notice that the cost is maximum for those elements located where the non-linearities are present.

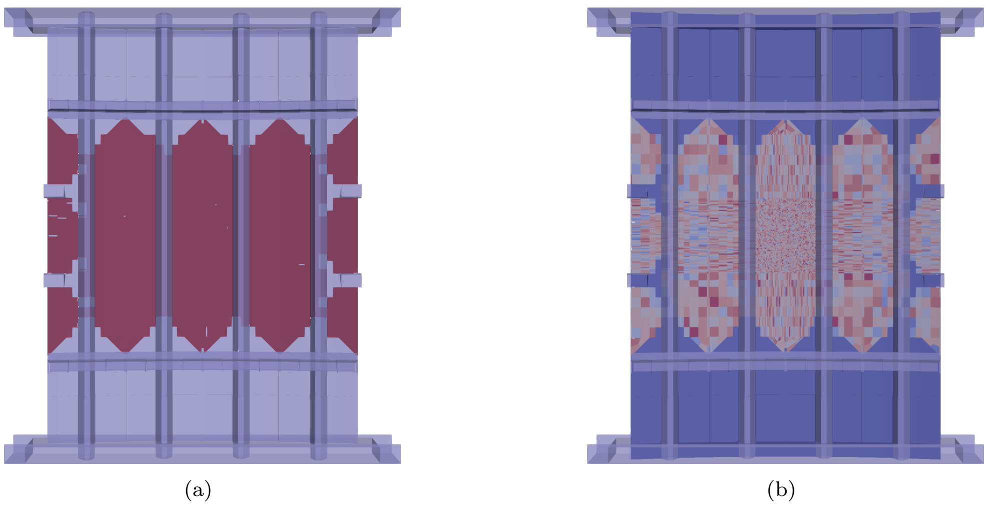


FIGURE 6.4: (a) Non-linearities in the macro-scale (in red) and the elements that remained in the linear region (in blue). (b) Computational cost distribution at the macro-scale.

Performance Results

The first performance experiment consists of a comparison between the accelerated CPU/GPU and the pure CPU approach. These results are outlined in Fig. 6.5. The simulations were carried out in 8 computing nodes (320 CPU cores and 32 GPUs) and micro-scale mesh resolutions of 25^3, 50^3, 75^3 and 100^3 elements were considered. The results show that for micro-scale mesh resolutions larger than 25^3 elements, the CPU/GPU acceleration reduces the calculation time required in the typical CPU scheme. For instance, for the micro-scale resolution of 100^3 elements, the speedup achieved with the CPU/GPU approach is around ×3.2; showing the substantial benefit of porting the application to CPU/GPU. The configuration of MPI tasks per computational node used is presented in Tab. 5.1.

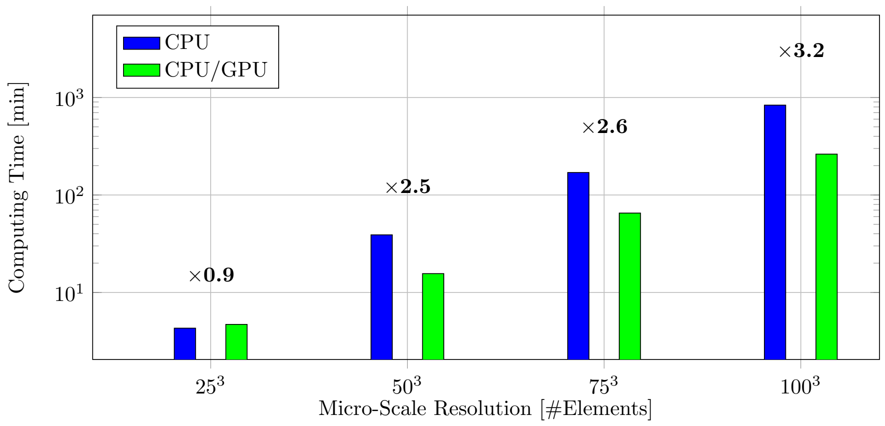


FIGURE 6.5: Calculation time for different micro-scale mesh resolutions using 8 nodes of the CTE-POWER cluster (320 CPU cores and 32 GPUs). Comparison using the pure CPU and the accelerated CPU/GPU approach.

The last result consists of a strong scaling experiment with a micro-scale resolution of 50^3 elements with the accelerated CPU/GPU approach using 8, 16, 24 and 32 computing nodes (1024 CPU cores and 128 GPUs). The simulations were carried out for 20 time steps, letting the simulation enter in the non-linear region. Fig. 6.6(a) shows the speedup curve and Fig. 6.6(b) displays the efficiency of the four experiments. Notice that the efficiency is still about 80% for the simulation with 32 nodes, demonstrating the suitable parallel performance of the application.

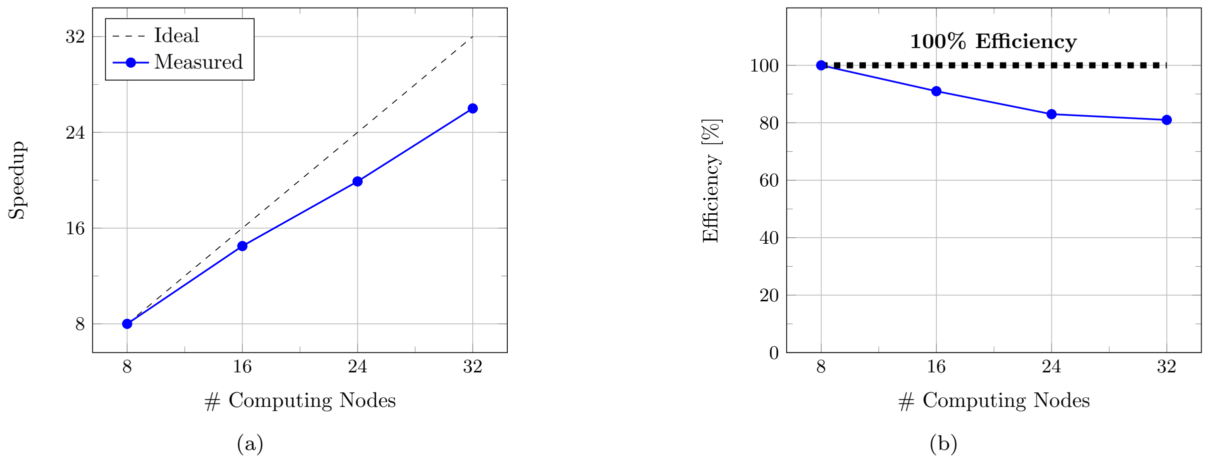


FIGURE 6.6: Strong scaling experiment using the accelerated CPU/GPU scheme. Each point consists of a non-linear simulation of 20 time steps with a micro-scale resolution of 50^3 elements. (a) Speedup. (b) Efficiency.

6.2 Impact of an Indenter in a Composite Panel

This example models the contact of an indenter on a composite panel of an aircraft fuselage, as it is shown in Fig. 6.7. In this case, Alya manages the contact mechanics through an additional coupling between the MPI instances and the PLE library that belongs to the Code_Saturne [3] CFD code. PLE library is responsible to handle the interchange of forces and displacements between the solid bodies for modeling the contact mechanics equations [31].

The panel, made of a composite material, is coupled purely with Micropp, while the indenter consists of an isolinear elastic material intrinsic of Alya. Fig. 6.7 shows the micro-structure for this example, this is a porous material with spheres with vacuum distributed randomly across the volume of the RVE. This micro-structure does not correspond to the real one used in the aircraft fuselage; in this case, it is used to evaluate the robustness of the application.

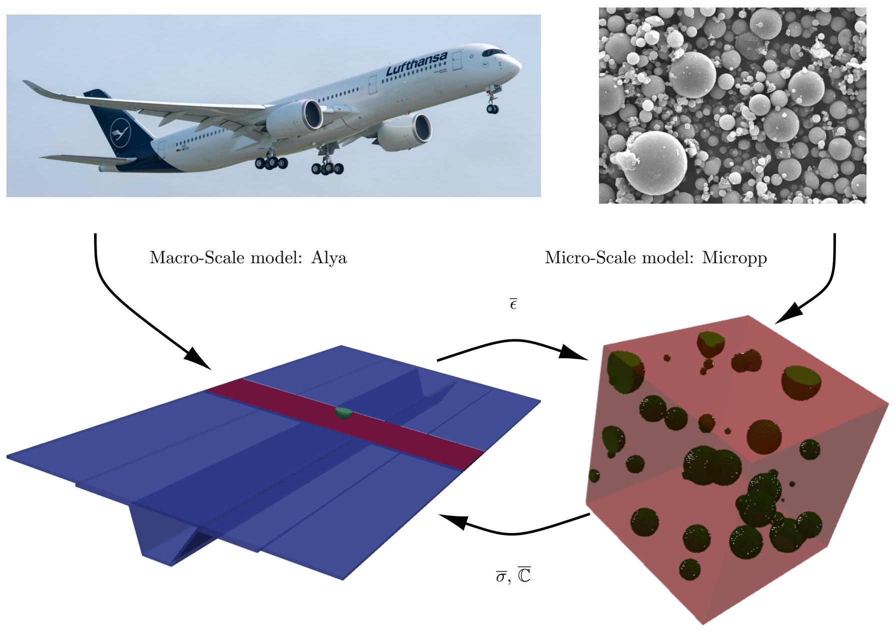

Macro-Scale model: Alya
Micro-Scale model: Micropp
$\overline{\epsilon}$
$\overline{\sigma}$, $\overline{\mathbb{C}}$

FIGURE 6.7: At the top, a picture of an A350 XWB aircraft of Airbus and a porous micro-structure (this does not correspond to the real fuselage). On the bottom, the macro-scale modelled with Alya and the micro-scale solved with Micropp.

Tab. 6.2 indicates the details for the domain discretization used. The macro-scale, modeled with Alya, consists of a mesh composed of 307K hexahedral elements, where 287K elements (colored with blue in Fig. 6.7) are coupled with the Linear scheme and 20K elements (colored with red in Fig. 6.7) are coupled with the One-Way approach. The indenter is composed of around 1K elements; this does not introduce additional computations to the algorithm since it is modeled with an intrinsic Alya material (not coupled with Micropp). For the micro-scale, mesh resolutions of 5^3, 25^3, 50^3, 75^3 and 100^3 elements were used to study the mechanical results and to evaluate the parallel performance.

The boundary conditions consist of a zero displacement condition at the bottom surface of the panel and a constant vertical velocity V_y at the top surface of the indenter, as it is outlined in Fig. 6.8.

The material models for the micro-structure consist in a plastic material with linear hardening [32] for the volume and an elastic materials for the spheres with vacuum (see the micro-structure in Fig. 6.7). The volume has a

	Macro-Scale	Micro-scale
Size [# elements]	Panel: 307K (Total) = 287K (Linear Coupling $\rightarrow$ MicroPP) 20K (One-Way Coupling $\rightarrow$ MicroPP) Indenter: 1K (Aluminium Model $\rightarrow$ Alya)	5^3 25^3 50^3 75^3 100^3

Table 6.2: Details of the domain discretization used at both scales.

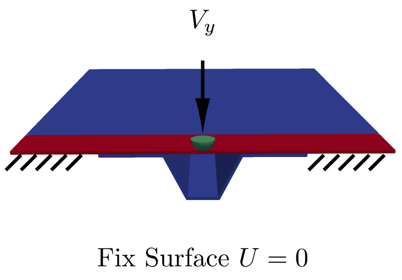


Figure 6.8: Boundary condition applied at the macro-scale. The bottom surface of the panel remains fix (zero displacement) while the top surface of the indenter moves vertically.

Young's Modulus of $E = 3.0 \times 10^7\,\mathrm{N/m^2}$, a Poisson Ratio of $\nu = 0.25$, a Yield Stress of $S_Y = 1.0 \times 10^7\,\mathrm{N/m^2}$ and a Hardening Modulus of $H = 1.0 \times 10^7\,\mathrm{N/m^2}$. The spheres with vacuum are modeled with an isotropic elastic material with a Young's Modulus of $E = 3.0 \times 10^3\,\mathrm{N/m^2}$ and a Poisson Ratio of $\nu = 0.25$. This micro-mechanical model can be tested in standalone mode with Micropp (without coupling) by running the benchmark: `benchmark-mic-2`. The execution of this last is explained in details in the documentation [10]. Additionally, the formulation of these material models are described in the Appendix A and some numerical examples with Micropp are provided.

Mechanical Results

Fig. 6.9 shows the displacement field over the macro-scale after one time step. The stress field of two RVE cells, corresponding to two different integration points over the macro-scale (indicated with arrows), are also visualized. Notice that the displacement is more significant at the contact region with the indenter. The same happens with the deformation field that is sent to the Micropp code to calculate the stresses. For this simulation the micro-scale mesh resolution indicated in Tab. 6.2 was used.

Fig. 6.10(a) displays the elements in the macro-scale that have entered in the non-linear zone after 10 time steps. For this simulation, a micro-scale composed of 5^3 elements is used to accelerate the computations. Notice that the non-linearities are placed around the central zone where the contact takes place. On the other hand, Fig. 6.10(b) shows the computational cost of the macroscopic elements; notice that this is also maximized in the central region because of the non-linearities caused by the contact.

Performance Results

The first performance simulation compares the accelerated CPU/GPU scheme with the pure CPU execution. Fig. 6.11 displays the results for different micro-scale resolutions (25^3, 50^3, 75^3 and 100^3 elements). Notice that the simulation with pure CPUs and with the micro-scale of 100^3 elements was not completed due to the Wall-Time limit of the CTE-POWER machine (setted in 48 hs). The latter is attributed to an implementation problem of

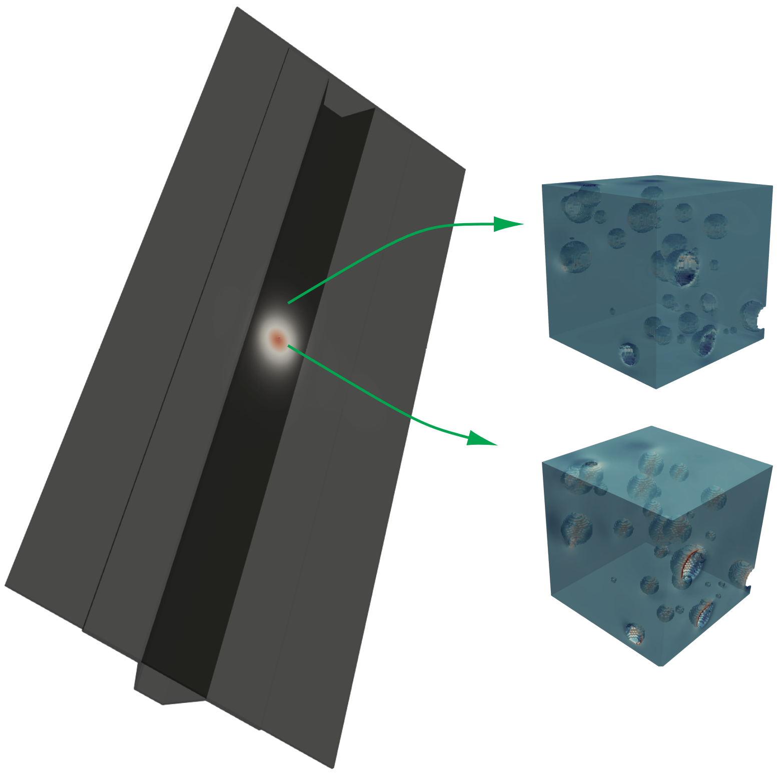


Figure 6.9: Displacement field over the macro-scale after one time step with two deformed micro-structures that correspond to two different integrations points of the panel.

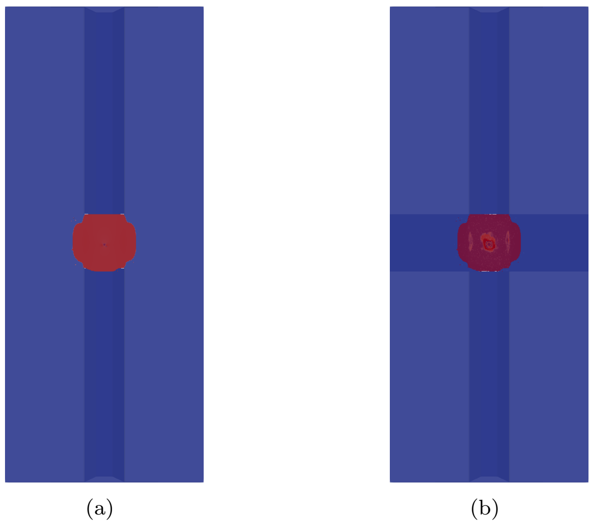


Figure 6.10: Simulation corresponding to 10 time steps using a micro-scale resolution of 5^3 elements. (a) In red, non-linearities concentrated in the macro-scale (b) Computational cost distribution at the macro-scale.

the MPI library inside the Alya code for this particular case (this has not happened for the CPU/GPU execution). Despite this inconvenience, it is clear that the speedup between pure the CPU/GPU and the pure CPU strategy increases with the micro-scale mesh resolution. For instance, a speedup of $\times 3.1$ is achieved for the micro-scale mesh of 75^3 elements.

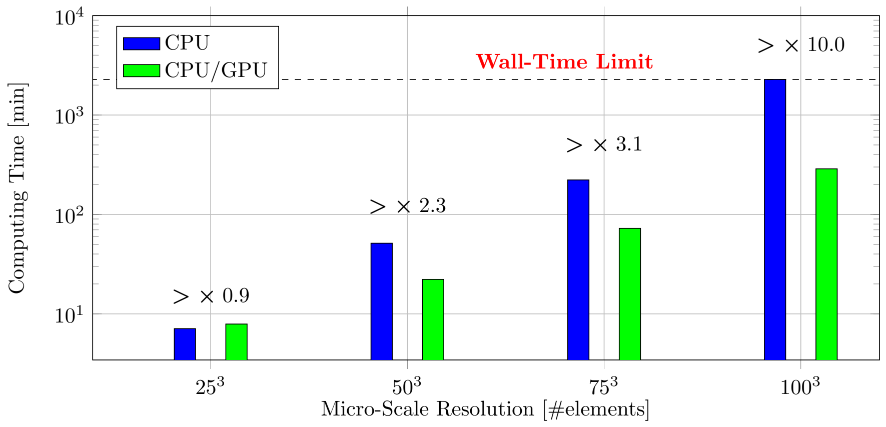


FIGURE 6.11: Comparison of the computing time for different micro-scale mesh resolutions between the CPU/GPU scheme and with pure CPU using 8 nodes of the CTE-POWER cluster (320 CPU cores and 32 GPUs).

The final experiment consists of a strong scaling test using the CPU/GPU acceleration approach up to 40 computing nodes of the CTE-POWER machine (1600 CPU cores and 160 GPUs). The micro-structure used for these experiments is composed of 50^3 elements and the simulation was performed for one time step. Fig. 6.12(a) shows the speedup curve for 8, 16, 24, 32 and 40 computing nodes, while Fig. 6.12(b) displays the corresponding efficiency for each case. Notice that the efficiency is still suitable (around 85%) up to 40 computing nodes showing the acceptable parallel scaling of the application.

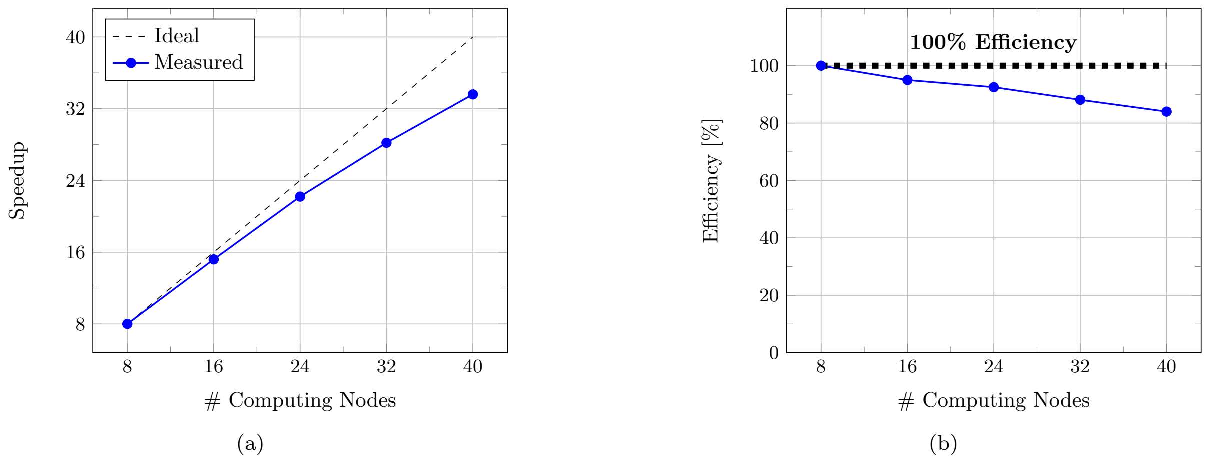


FIGURE 6.12: Strong scaling test using the accelerated CPU/GPU scheme with a micro-scale mesh resolution of 50^3 elements and one time step. (a) Speedup. (b) Efficiency.

6.3 Impact over an Aircraft Wing

This example simulates the impact of a hailstone over a composite aircraft wing. At the top of Fig. 6.13, a description of the real problem is outlined. This shows a wing of the A350 XWB mainly built with composite materials and a generic micro-structure with carbon fibers arranged in random directions. The latter does not correspond to any component of the real aircraft, this micro-structure is used only for showing the capabilities of the code. At the bottom of Fig. 6.13, the FE element meshes for the macro-scale and the micro-scale are visualized. For simplicity, the hailstone for this case is modeled with the same indenter used in Section 2.

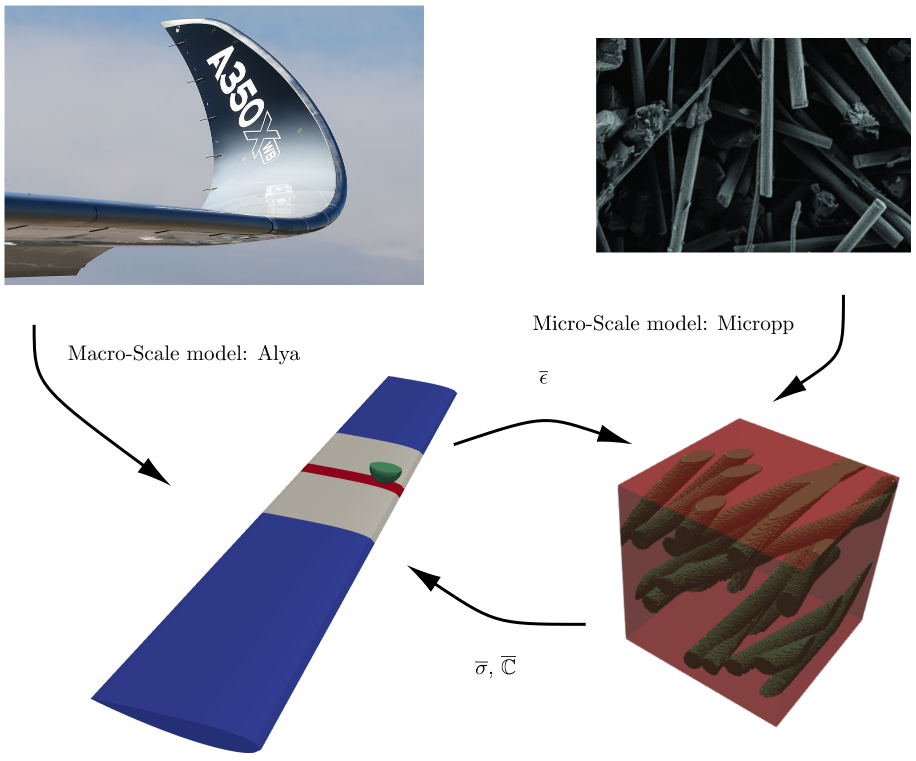

A350 XWB
Macro-Scale model: Alya
Micro-Scale model: Micropp
$\overline{\epsilon}$
$\overline{\sigma}, \overline{\mathbb{C}}$

FIGURE 6.13: On the top, a wing that belongs to the A350 XWB aircraft and an augmented image of the composite material micro-structure (this does not correspond to the real fuselage). On the bottom, the macro-scale modelled with Alya and the micro-scale solved with Micropp.

Tab. 6.3 indicates the details about the FE discretization for both scales. The macro-scale is composed of 9K tetrahedral elements. Among these, 1K elements (in the red zone, see Fig. 6.13) are coupled with the Full scheme to model, 4K (in the white region) with the One-Way scheme and 4K with the Linear scheme. The weights for the domain partitions at the macro-scale are also indicated in Tab. 6.3.

	Macro-Scale	Micro-Scale
Size [# elements]	9K (Total) = 1K, weight = 1000.0 (Full Coupling (FE) with Micropp) 4K, weight = 400.0 (One-Way Coupling (FE) with Micropp) 4K, weight = 1.0 (Linear Coupling with Micropp)	25^3 50^3 75^3 100^3

TABLE 6.3: Details of the domain discretization used at both scales.

The boundary conditions, visualized in Fig. 6.14, consist of a zero displacement condition at the bottom surface of the wing and a constant vertical velocity V_y at the top surface of the indenter.

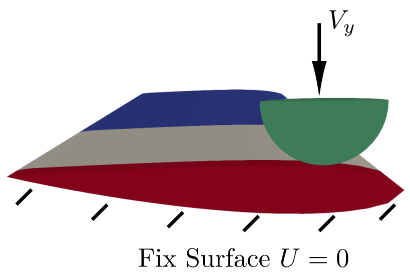


FIGURE 6.14: Boundary conditions applied at the macro-scale.

The material models for the RVE cell consist of a damage material with linear hardening [33] for the fibers and an isotropic elastic law for the epoxy resin. The fibers have a Young's Modulus of $E = 3.0 \times 10^8\,\mathrm{N/m^2}$, a Poisson Ratio of $\nu = 0.25$, a Yield Stress of $S_Y = 1.0 \times 10^7\,\mathrm{N/m^2}$ and a Hardening Modulus of $S_Y = 1.0 \times 10^7\,\mathrm{N/m^2}$. The elastic material of the resin has a Young's Modulus of $E = 3.0 \times 10^7\,\mathrm{N/m^2}$ and a Poisson Ratio of $\nu = 0.25$. Micropp includes a benchmark (`benchmark-mic-3`) for simulating this RVE cell in standalone mode (without coupling with Alya) which is explained in detail in the documentation [10]. Additionally, the formulation of these materials models is explained with detail in the Appendix A, including some examples with Micropp.

Mechanical Results

In Fig. 6.15 the deformation field over the wing is visualised after the impact. This corresponds to a single time step simulation. On the other hand two deformed RVE cells with the stress field are outlined; these correspond to two different integrations points of the macro-scale.

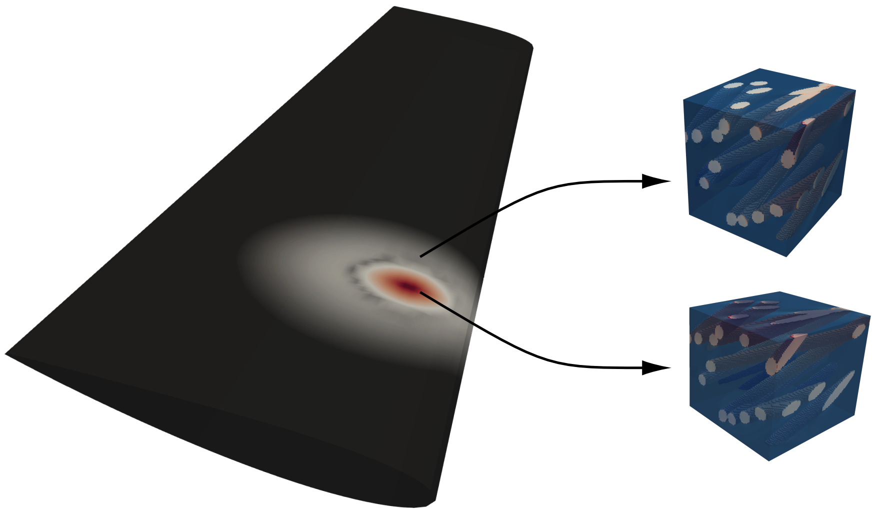


FIGURE 6.15: Displacement field over the macro-scale after one time step and the stress field in two deformed RVE cells both corresponding to two different integrations points of the panel.

The next simulation, shown in Fig. 6.16, consists of 20 time steps applying a micro-scale mesh resolution of 5^3 elements. In Fig. 6.16(a) the localization of the non-linearities in the macro-scale is visualized. On the other hand, Fig. 6.16(b) shows the computational cost distribution over the wing. Notice that the maximum is reached around the contact zone region.

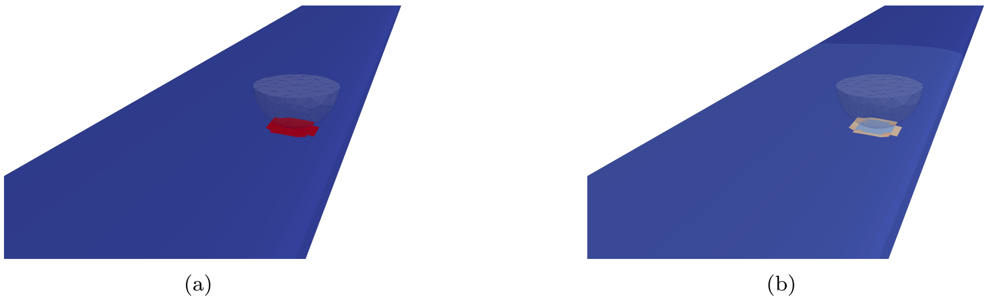


FIGURE 6.16: (a) Non-linearities in the macro-scale (in red) and the elements that remained in the linear region (in blue). (b) Computational cost distribution at the macro-scale.

Performance Results

The first experiment compares the performance of the FE2 implementation using the pure CPU approach with that of the accelerated CPU/GPU scheme. The simulation is done for a single time step and using different micro-scale mesh resolutions (25^3, 50^3, 75^3 and 100^3 elements). The results are visualized in Fig. 6.17. Notice that the speedup with the accelerated CPU/GPU increases as the micro-scale mesh resolution increases. This achieves a value of $\times 2.8$ for the micro-scale resolution of 100^3 elements. The reason why the speedup decreases from the mesh resolution of 75^3 to 100^3 elements has not been explained yet. It is considered that this is due to the MPI machine configuration and the way the MPI processes are allocated in the node.

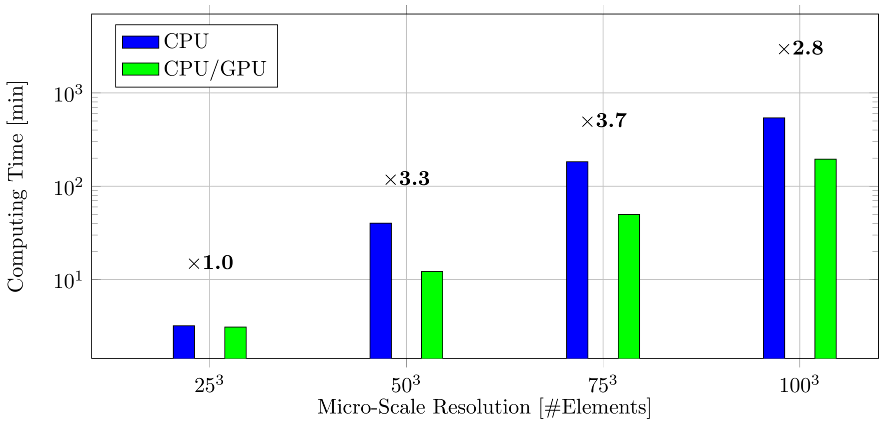
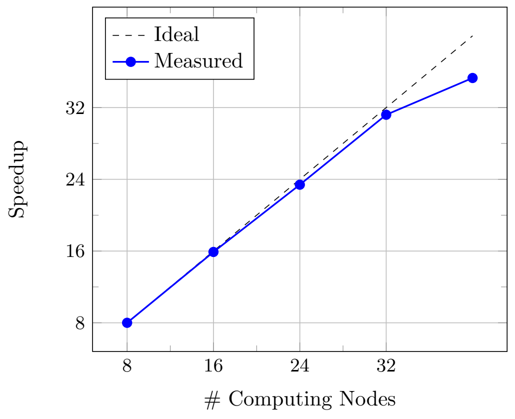
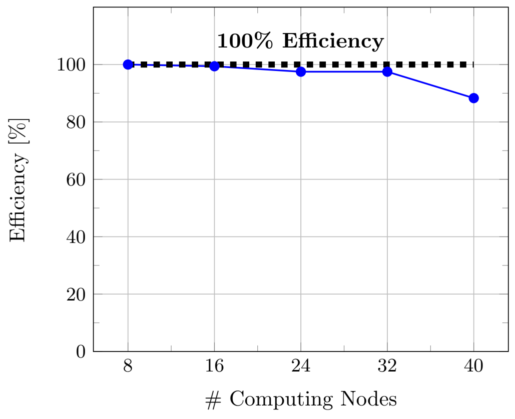


FIGURE 6.17: Calculation time for different micro-scale mesh resolutions using 8 nodes of the CTE-POWER cluster (320 CPU cores and 32 GPUs). Comparison using the pure CPU scheme and the accelerated CPU/GPU approach.

The final result is from a strong scaling experiment with the accelerated CPU/GPU approach. The simulations are done performing one time step and using a micro-scale mesh of 75^3 elements. Fig. 6.18(a) shows the speedup for five experiments running on 8, 16, 24, 32 and 40 computing nodes (160 GPUs maximum). On the other hand, Fig. 6.18(b) shows the efficiency for each of these executions. Notice that the parallel efficiency keeps staying high (89%) up to 40 nodes simulation.



(a)



(b)

FIGURE 6.18: Strong scaling test using the accelerated CPU/GPU scheme. Each case consists of a simulation of one time step and micro-scale resolution of 75^3 elements. (a) Speedup. (b) Efficiency.

6.4 Conclusions

The chapter expose three different engineering examples that evaluate the parallel performance of the FE2 multi-scale application coupling Alya and Micropp. The first two examples contain around 10K macro-scale elements coupled with Micropp using the One-Way scheme and a micro-scale mesh resolution of 100^3. The third example contains 1K macro-scale elements coupled with Micropp activating the Full scheme and 4K elements with the One-Way scheme. For this case, the micro-scale mesh resolution is also 100^3 elements.

The three examples demonstrate a crucial improvement using the accelerated CPU/GPU approach over the pure CPU execution: 2-4 times faster execution using the accelerated CPU/GPU scheme for all the cases. In addition to that, for up to 32 computing nodes (1280 CPU cores and 128 GPUs), an efficiency of 80% or above was achieved applying the accelerated CPU/GPU scheme. This confirms that the current implementation is robust to be used in HPC calculations and suitable for industrial applications. The most recent works in the FE2 multi-scale method [8, 9] have solved problems consisting of around 0.2K elements in the macro-scale and 100^3 elements in the micro-scale. The present work, applying the proposed hybrid CPU/GPU acceleration strategy, has solved problems consisting of more than 10K elements in the macro-scale and 100^3 elements in the micro-scale.

Chapter 7

Conclusions

The numerical resolution of composite material problems has been a challenge during the last decades due to the high heterogeneity and the length scales involved. The multi-scale methods were shown to be a suitable alternative to deal with these kinds of problems. In particular, the phenomenological laws were the most used in the industry during the last years. Unfortunately, the applicability of them is limited to simple micro-structures where the equivalent constitutive laws can be well approximated; they also have difficulties in predicting the behavior of non-linear mixtures. The improvements in the technology have allowed the FE2 multi-scale methods to gain reputation and to become an alternative for solving composite problems. The essential advantage of the FE2 method is that the work of computing the constitutive law is done automatically by solving an FE problem in a micro-scale RVE cell. The FE2 method is a tool capable of dealing with non-linear composites problems decreasing dramatically the number of assumptions and approximations needed. Its main disadvantage is the huge computational cost that it demands to solve the problems at both scales.

7.1 Discussion

The present thesis has exposed a new and full parallel GPU-accelerated implementation of the FE2 multi-scale algorithm for simulating complex composite material structures. The method consists primarily of the coupling of the multi-physics MPI-based code Alya for solving the macro-scale and the solid mechanics code Micropp for performing the Localization-Homogenization at the micro-scale. The current implementation has achieved to port and accelerate the micro-scale code Micropp with GPUs using the OpenACC parallel programming model. The accelerated application with CPU/GPU has shown speedups of around $\times 2 - 4$ for the coupling case (for a micro-scale resolution of 100^3 elements) with respect to the execution on pure CPUs. The standalone executions of Micropp in CPU/GPU mode has exhibited a speedup of about $\times 25$. These results have demonstrated that the CPU/GPU acceleration strategy is a well suited idea to improve the performance of the FE2 applications making it a competitive tool for solving real industrial problems.

Fig. 7.1 displays the micro-scale and macro-scale problem sizes that were solved with the current FE2 implementation and comparable works in literature [6, 7, 8, 9]. The future perspectives of this work aim to boost the capabilities of this implementation for the Exascale computing, expanding the complexity of the problems that can be solved and reducing the computing time.

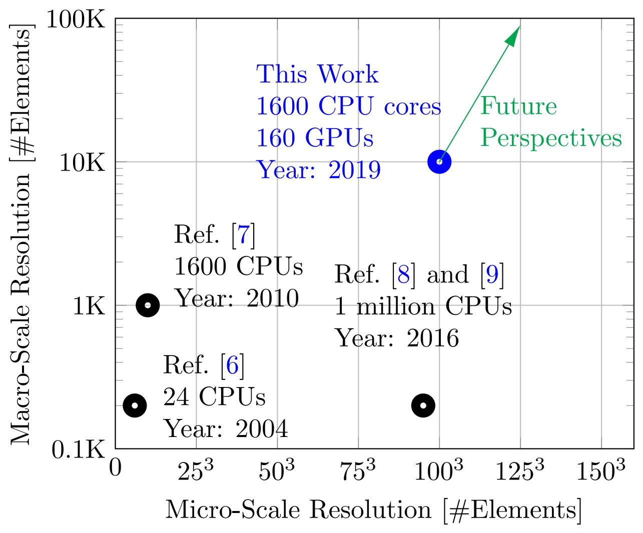


FIGURE 7.1: The FE2 multi-scale implementation presented in this work competing with other codes developed.

One of the essential parts of this work which has allowed the FE2 application to compete between other works in the literature was the implementation of the Micropp library. This was written from scratch in C++ language and it can be coupled with C, C++ and Fortran codes (for modeling the macro-scale). Micropp configuration is made with CMake, this allows to compile the code in a broad variety of machines in a straightforward manner. During this thesis, the code has been compiled successfully for Intel and IBM CPU architectures, and as well as for NVidia GPUs. Micropp also uses CTest, a CMake feature, to run the unity tests and to check the validity of the results when some modifications are done or when different configurations are required. Micropp is Free Software, meaning that future collaborations are desired to maintain, test and validate the code.

Several limitations in the FE2 application have been found during this thesis. From the mechanical point of view, the use of the Uniform Strain boundary condition applied at the micro-scale RVE is not accurate for aircraft composite materials [13]. The results obtained with Micropp using Uniform Strains have shown that, for non-linear cases, damage and plasticity are localized in the boundaries. In practice, these variables appear in different parts of the RVE (not only around the boundary). Periodic and Uniform Stress boundary conditions solve this issue and lead to better results. These boundary conditions have not been implemented in Micropp due to computational implementation difficulties. Micropp code was optimized to operate entirely with the Uniform Strain boundary condition. The Matrix-Vector product for the Conjugate Gradient algorithm, and also the elemental assembly on the Jacobian are optimized for this boundary condition. Periodic and Uniform Stress boundary conditions demand more complex algorithms (and data structures) to be implemented.

The micro-structure represented in Micropp are intrinsic to it. This means that the representation of the fibers, void spheres, and other inclusions shapes are written in the code. This design can affect the scalability of the code for solving more complex micro-structure patterns. Currently the strategy permits only to vary the parameters that characterize these patterns, i.e., the fiber radii. The disadvantage is that new patterns would demand to be programmed in Micropp and this requires to recompile the code.

Scarce constitutive laws are implemented on Micropp to represent a real composite micro-structure. These laws are limited to the isotropic elastic, the plastic and the damage model with hardening (see Appendix A). More complex material laws should be implemented to model industrial composites. For the future, Micropp should incorporate new materials laws for achieving and validate the code with experimental results.

The steady-state case and the small deformations assumptions are not enough for modeling some of the solid mechanics problems, i.e., a high-speed impact of a hailstone in a wing. More refined models, including the inertial force terms and large deformation theory, would be required to capture the physics of these phenomena. Large deformation theory would demand constitutive material laws formulated under that assumptions taking into account the strain measures which depends on the system of reference (the referential or the deformed system) [34]. Moreover, the current application maintains the fiber orientation for all the integration points of the macro-structure. In a curved geometry this assumption results in a wrong model since the bad orientation of the micro-scale model.

The parallel performance of the implementation presents several limitations. A load balance problem appears in the transition of the micro-scale from the linear range to the non-linear range [11]. For the coupling of Alya and Micropp, every MPI task of Alya is coupled with an instance of Micropp which solves, for every macro-scale iteration, a certain group of micro-scale problems related to the integration point of the Alya sub-domain. In a typical simulation the non-linearities are generally concentrated in different regions of the macro-structure which is translated in a high load unbalance between the MPI processes affecting dramatically the parallel performance. Despite the use of the weighted partitioning algorithm increases the parallel performance when the Multi-Zone coupling strategy is used (to reduce the computational cost), the scheme is static and does not takes into account the non-linearities that appear during the simulation which can increase again the load balance.

7.2 Future Perspectives

The weaknesses and achievements reached for the computational aspects of the FE2 code developed in this thesis have inspired several work lines:

- Dynamic load balance algorithms are necessary for dealing with non-linear problems. A little invasive solution for the code coupling consists in applying a load balance algorithm at the macro-scale, intrinsically or by adding an external library. The algorithm should schedule the execution of the micro-scale problems in the processing units of the distributed architecture taking into account their computational cost. Unlike the current method, this does not attach the Micropp instances to a fixed group of integration points of the macro-scale. It is important that for non-linear cases, the micro-scale time-dependent variables are demanded for computing the new time steps; this means that the dynamic scheduler would need to move these variables across the processes (and the memory of the distributed architecture).
- The Periodic and the Uniform Stress boundary conditions are required to model properly the behavior of non-linear phenomena at the micro-scale. The implementation of them in Micropp should be compatible to the high efficiency design of the code and the GPU portability. The most critical data structure for

implementing these conditions is the Jacobian matrix since the pattern of the coefficient is more complicate than for the Uniform Strain boundary condition (see Chapter 2).

- Micropp demands several improvements for being able to model more realistic composite designs for aircraft. From the mechanical point of view, new material models, large deformation theory and inertial force terms can be included to increase the validity of the method. Also, more coupling parameters, i.e. the temperature, can be included to study thermal effects on materials such as the metal alloys. Also, new solvers and preconditioners can be implemented to solve the linear system of equations efficiently.
- The use of OpenACC for the assembly and the solver stages has demonstrated good acceleration in the CPU/GPU scheme. This last motivates the use of CUDA programming model to accelerate the computations of these algorithmic steps in GPUs. Other strategies such as performing a massive data transfer of more than one micro-scale problem at the time can be studied in order to offload more work to the GPU.

Appendix A

The Micropp Code

Micropp V3.11 [10] is a C++ solid mechanics code designed to perform FE element calculations over cells. These represent mixtures of more than one materials with different mechanical properties, the main goal it is to obtain the average mechanical behavior of these mixtures. The main application case of Micropp is to model composite material cells like the carbon fiber. The code was designed mainly to be coupled with a wide variety of general solid mechanics macro-scale codes in order to carry out FE2 multi-scale calculations [35].

In a coupled simulation, Micropp receives the macro-scale strains $\bar{\epsilon}$ that correspond to the integration points of the elements in the macro-scale. For each of them it performs the localization inside the cells through the imposition of boundary conditions and solving a FE problem for the equilibrium [13]. After that homogenizes the stress fields that were calculated in the FE computations by performing average integrals. These average stress tensors $\overline{\sigma}$ are returned back to the macro-scale for each of the integration points that demand them.

Micropp V3.11 is implemented as a library in order to be coupled with different macro-scale codes which can be written in C, C++ and Fortran. The most important code with it was coupled is the parallel multi-physics code Alya [23] that was used to perform FE2 multi-scale calculations for aeronautic structures. The main advantage of Micropp against other codes is the CPU/GPU acceleration which has proven to accelerate substantially, up to a factor of $\times$ 20, the computation of the micro-scale problems [10].

In the first Section the code structure of Micropp is explained in detail, starting from the main parts of the code and following with the main public functions that are used. In the second Section the performance results of Micropp for different numerical experiments are exposed, the CPU/GPU acceleration strategy is compared with CPU only showing the advantages of the first for high micro-scale resolutions. In the third Section the material models implemented in Micropp are tested. Micropp is Free Software and is freely distributed [10]. All the results of this Appendix can be reproduced by simply download Micropp V3.11 from [10]. For the performance results the hardware and compilers are also specified in order to perform future technological comparisons.

A.1 Code Layout

Micropp was designed in C++ library form applying Object Oriented Programming (OOP). The code is used to solve non-linear FE problems on heterogeneous RVEs applying implicit algorithms, i.e., linear system of equations should be solved iteratively. This makes that the most computational intensive parts of Micropp are those needed for the assembly of the matrices and the vectors of the FEM discretization together with the iterative solver for the linear system of equations.

Micropp is written in library form, the basic way of how it works consist in first instantiating an object of the basic Micropp class, call the public functions to perform the localization and the homogenization processes, and destroy the main class. The `benchmark-mic-1` [10] is a simple example where these steps are performed. Notice that Micropp does not read any input file, all the parameters like the micro-structure configuration and the materials models of the constituents, between others are passed to the constructor during the instantiation.

In Fig. A.1 shows the basic Micropp code layout. The direction of the arrows represent the direction of the information. For example, for the instantiation of Micropp main class the macro-scale code sends the initialization parameters to the library while in the request of the stress and tangent constitutive tensors ($\overline{\sigma}$ and $\overline{\mathbb{C}}$) the macro-scale code calls a Micropp function for getting the tensors' values.

The most computational intensive functions of Micropp, the assembly of the matrices and the vectors and the linear solver of equations are CPU/GPU accelerated. Notice that this is displayed in Fig. A.1. The code applies OpenACC to generate the code for being run on CPU/GPU, this capability is activated using a preprocessor option. By default the code is compiled to run on CPU only.

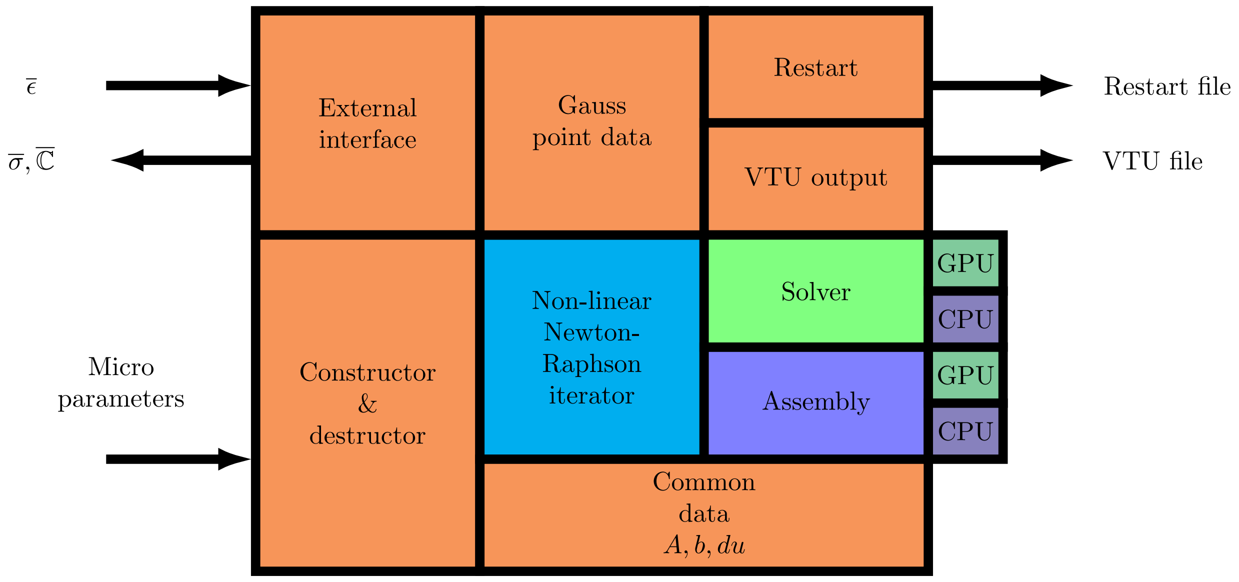


FIGURE A.1: Basic structure of the Micropp code.

The most important public functions of Micropp are:

- Constructor `micropp(const micropp_params_t &params)`: This function creates the micropp class initializing the data needed for the calculation of all the micro-scale problems. The structure `params` stores the size of mesh, the number of macro-scale integration points to solved and the coupling scheme for each (Linear, One-Way or Full). This is generally called by each MPI process from the macro-scale during the initialization and with the corresponding data of the sub-domain at which it represents.

- Destructor `~micropp()`: frees all the memory allocated by the micropp instance. Also is called by every MPI processes who has instantiated micropp and at the end of the calculation.
- `set_strain(const int gp_id, const double *strain)`: the macro-scale uses it to send the strain of each of its integration points to the micro-scale.
- `get_stress(const int gp_id, double *stress)`: the macro-scale calls this to get the macro-scale stress tensor homogenized in the RVE.
- `get_ctan(const int gp_id, double *ctan)`: the macro-scale calls this to get the tangent constitutive tensor homogenized in the RVE.
- `void homogenize()`: this goes around every macro-scale integration point of a list and depending on the coupling scheme and the strain value $\bar{\epsilon}$ stored performs the homogenization and calculates $\overline{\sigma}$ and $\overline{\mathbb{C}}$.
- `void update_vars()`: This function updates the internal variables of each micro-scale problem and should be called in each time step when the macro-scale has converged to the equilibrium.
- `int is_non_linear(const int gp_id)`: Returns 1 is the macro-scale integration point has entered in the non-linear region at least one time and 0 if not.
- `int get_cost(int gp_id)`: Returns the computational cost of a macro-scale integration point. This is defined as the total number of iterations in the CG solver for all the Newton-Raphson iterations needed to localize the macro-strain psmacm.
- `bool has_converged(int gp_id)`: Returns `true` if the Newton-Raphson has applied to macro-scale has converged and achieved the equilibrium. Returns `\false` if not.
- `void output(int gp_id, const char *filename)`: Outputs a VTU file representing the RVE mesh with the most important variables of the micro-scale: displacements, strains, stresses and internal variables.
- `void write_restart(const int restart_id)` and `void read_restart(const int restart_id)` write and read a restart file respectivelly. The restart files stores the internal variables in the integration points of the micro-scale needed to compute the following time step.

All of these function has their correspondent C and Fortran wrapping interface in order to be called from different macro-scale implementations.

A.2 Micro-Scale Performance

Tab. A.1 indicates the computation time for the assembly of the jacobian matrix and the residue (two times) and the solver stage. These are the procedures that need to be taken to perform one Newton-Raphson iteration, i.e., for a linear or elastic problem. The measurements where obtain by performing the calculation with Micropp in an

IBM Power9 processor (using one core to solve the whole problem). Tab. A.2 displays the same measurements but run on a GPU NVIDIA Tesla V100.

Size	Matrix [mS]	RHS [mS]	Solver [mS]	Total [mS]
$10 \times 10 \times 10$	13	4.4	11	28
$20 \times 20 \times 20$	100	35	140	280
$30 \times 30 \times 30$	340	120	670	1100
$40 \times 40 \times 40$	800	280	2100	3100
$50 \times 50 \times 50$	1600	540	4900	7000
$60 \times 60 \times 60$	2700	940	9900	14 000
$70 \times 70 \times 70$	4400	1500	18 000	24 000
$80 \times 80 \times 80$	6400	2200	30 000	39 000
$90 \times 90 \times 90$	9200	3200	48 000	61 000
$100 \times 100 \times 100$	13 000	4400	73 000	90 000

Table A.1: Computation times in a CPU core IBM Power9 for different stages of the FE micro-scale problem and for different mesh sizes using micropp.

Size	Matrix [mS]	RHS [mS]	Solver [mS]	Total [mS]
$10 \times 10 \times 10$	24	10	7.1	42
$20 \times 20 \times 20$	41	27	20	87
$30 \times 30 \times 30$	88	66	48	200
$40 \times 40 \times 40$	180	160	99	440
$50 \times 50 \times 50$	330	290	190	800
$60 \times 60 \times 60$	530	470	330	1300
$70 \times 70 \times 70$	820	750	560	2100
$80 \times 80 \times 80$	1200	1100	900	3200
$90 \times 90 \times 90$	1800	1600	1400	4700
$100 \times 100 \times 100$	2400	2100	2100	6600

Table A.2: Computation times in a GPU NVIDIA Tesla V100 for different stages of the FE micro-scale problem and for different mesh sizes using micropp.

A.3 Material Models

Micropp V3.11 implements several constitutive laws to model the constituents of the composite material. The models implemented are the elastic isotropic law [30] (linear), the plastic model with linear hardening [32] (non-linear) and the damage model with linear hardening [33] (non-linear). These models are used to represent each of the constituents. Notice that the use of at least one non-linear material law to represent one of the constituents of the cell makes the mixture to adopt a non-linear behavior.

Isotropic elastic

The isotropic elastic material law expresses a linear relation between the stress σ and the strain ϵ is [30]

$$\sigma_{ij} = \lambda \epsilon_{kk} \delta_{ij} + \mu \epsilon_{ij}. \tag{A.1}$$

where λ is the Lame's first parameter and μ the shear module. These can be calculated using the Young module E and the Poisson ratio ν by:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2(1+\nu)}. \tag{A.2}$$

The tangent constitutive operator tensor can be written as [30]

$$C_{ijkl} = \lambda \delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{A.3}$$

Fig. A.2 (a) outlines the stress-strain relation for the elastic material implemented on Micropp. The result corresponds to the benchmark **`benchmark-elastic`** [10] that can be run following the procedure detail in the Micropp User Manual [10].

Plastic with linear hardening

The plasticity model implemented in Micropp corresponds to the isotropic linear hardening rule described in [32]. The model starts considering an elatic trial variables after the increment in the deformation:

$$\begin{cases} e_{n+1} = \epsilon_{n+1} - \frac{1}{3}(tr(\epsilon_{n+1})), \\ s^{\text{trial}}_{n+1} = 2\mu(e_{n+1} - e^p_n), \\ f^{\text{trial}}_{n+1} = ||s^{\text{trial}}_{n+1}|| - \sqrt{\frac{2}{3}}(\sigma_Y + K_a \alpha_n). \end{cases} \tag{A.4}$$

where ϵ^p is the plastic deformation and α is the hardening parameter, both are internal variables for this model. The f function is the Yield Condition that determines if the material remains on the elastic zone ($f \leq 0$) or if it has moved to the plastic region ($f > 0$).

For the case that the Yield Condition remains in the elastic zone ($f \leq 0$) the value of the variables correspond to the trial ones and the internal variables remain unchanged:

$$\begin{cases} e^p_{n+1} = e^p_n, \\ \alpha_{n+1} = \alpha_n, \\ s_{n+1} = s^{\text{trial}}_{n+1}. \end{cases} \tag{A.5}$$

On the opposite case, if the material has entered in the plastic zone ($f > 0$), the internal variables are updated as:

$$\begin{cases} \mathbf{n}_{n+1} = \frac{s^{\text{trial}}_{n+1}}{||s^{\text{trial}}_{n+1}||} \quad \Delta\gamma = \frac{f^{\text{trial}}_{n+1}}{2\mu(1+K_a)} \\ \epsilon^p_{n+1} = \epsilon^p_n + \Delta\gamma \mathbf{n}_{n+1}, \\ \alpha_{n+1} = \alpha_n + \sqrt{\frac{2}{3}}\Delta\,\gamma, \\ \sigma_{n+1} = k\,\text{tr}(\epsilon_{n+1}) + s^{\text{trial}}_{n+1} - 2\mu\Delta\gamma\mathbf{n}_{n+1}. \end{cases} \tag{A.6}$$

To calculate the tangent constitutive tensor the perturbation procedure described on Eqs.2.17, 2.18 and 2.19 is followed. The plastic model is tested through the Micropp **`benchmark-plastic`** [10]. In Fig. A.2 (b) the resulting non-linear stress-strain curve is visualised.

Damage with hardening

The damage model implemented on Micropp is based on [33]. As in the plastic model, the method starts considering a linear trial for the stress and evaluating a damage condition

$$\begin{cases} \sigma_{\text{elastic}} = \mathbb{C}_{\text{elastic}} : \epsilon, \\ r_{n+1} = \sqrt{\sigma_{\text{elastic}} : \epsilon}. \end{cases} \tag{A.7}$$

where $r_{t=0} = r_0$. For the case that $r_{n+1} \leq r_n$ then

$$\begin{cases} r_{n+1} = r_n, \\ D_{n+1} = 1 - \frac{q(r_n)}{r_n}. \end{cases} \tag{A.8}$$

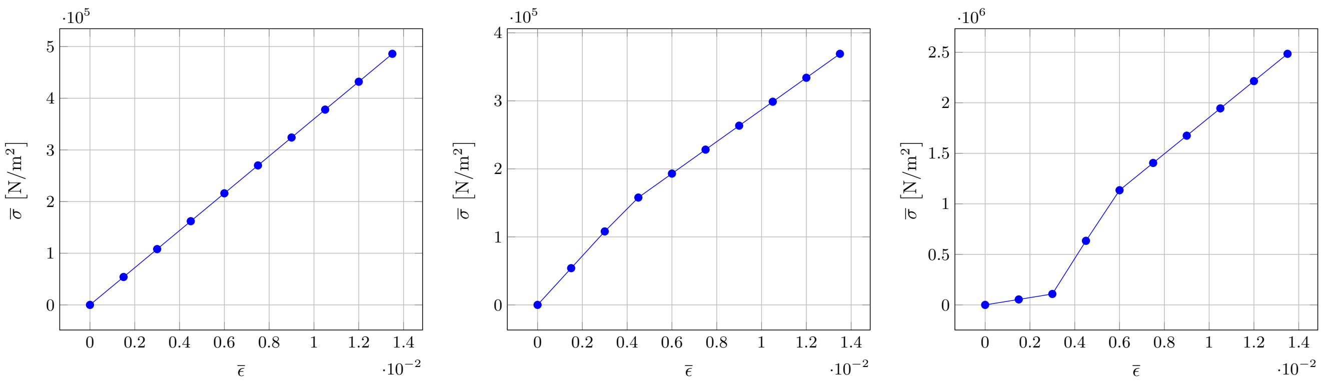
where $q(r)$ is a hardening law that in this case corresponds to a bilinear function detailed in [10] and D is the damage variable. For the case where $r_{n+1} > r_n$ then

$$\begin{cases} r_{n+1} = r_n, \\ D_{n+1} = 1 - \frac{q(r_{n+1})}{r_{n+1}}. \end{cases} \tag{A.9}$$

Finally for both cases the stress and the tangent constitutive tensors are evaluated with the new damage variable as:

$$\begin{cases} \sigma_{n+1} = (1 - D_{n+1})\sigma_{\text{elas}}, \\ \mathbb{C}_{n+1} = (1 - D_{n+1})\mathbb{C}_{\text{elas}}. \end{cases} \tag{A.10}$$

In Fig. A.2 the non-linear stress-strain relation curve corresponding to the damage model simulated with `benchmark-damage` [1 of Micropp is indicated.



(a) Elastic (b) Plastic (c) Damage

FIGURE A.2: Stress-strain relation ($\overline{\sigma} - \overline{\epsilon}$) for the material models implemented on Micropp. The results correspond to the benchmarks: (a) `benchmark-elastic`, (b) `benchmark-plastic` and (c) `benchmark-damage` [10].

A.4 Non FE-based Homogenization Coupling Models

Micropp implements an additional model based on the Chamis's Rule of Mixtures [36] to perform the homogenization on the cell without performing any FE computation on it. It works under the linear or elastic assumption and it is used mainly for validation of the FE coupling models.

The Chamis model [36] assumes a composite cell with long fibers and expresses the compliance matrix [15] in terms of six parameters E_{11}, E_{22}, G_{12}, G_{23}, ν_{12} and ν_{23}:

$$\mathbb{S} = \begin{pmatrix} 1/E_{11} & -\nu_{12}/E_{11} & -\nu_{12}/E_{11} & 0 & 0 & 0 \\ -\nu_{12}/E_{11} & 1/E_{22} & -\nu_{23}/E_{22} & 0 & 0 & 0 \\ -\nu_{12}/E_{11} & -\nu_{23}/E_{22} & 1/E_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G_{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G_{12} \end{pmatrix} \tag{A.11}$$

These last are obtained through:

$$\begin{aligned} E_{11} &= V^f E^f + V^m E^m, \\ E_{22} &= \frac{E^m}{1 - \sqrt{V^f}(1 - E^m/E^f)}, \\ \nu_{12} &= V^f \nu^f + V^m \nu^m, \\ G_{12} &= \frac{G^m}{1 - \sqrt{V^f}(1 - G^m/G^f_{12})}, \\ G_{23} &= \frac{G^m}{1 - \sqrt{V^f}(1 - G^m/G^f_{23})}, \\ \nu_{23} &= \nu_{12} \end{aligned}$$

where V^f, V^m, E_f, E_m, ν_f and ν_m are the volumetric fractions, the Young modules and the Poisson ratios of the fibers and cells respectively.

Micropp includes a test example `benchmark-mic-4` [10] that serves to perform a comparison of the Chamis model for two micro-structure cell kinds shown in Fig. A.3. The Chamis model works is designed to model parallel fibers arrangements which means that a more accurate result should be obtained for the model on the left of Fig. A.3 than the one on the right.

In Fig. A.4 a Stress vs. Deformation comparison is indicated for the FE element homogenization and the Chamis Rule of Mixture for both micro-structures. It can be noticed that the Stress-Deformation relation of both models is similar for the parallel oriented fiber micro-structures but starts differing for the random oriented example. This suggests that the FE model keeps the accuracy even when the complexity of the micro-structure increases. This

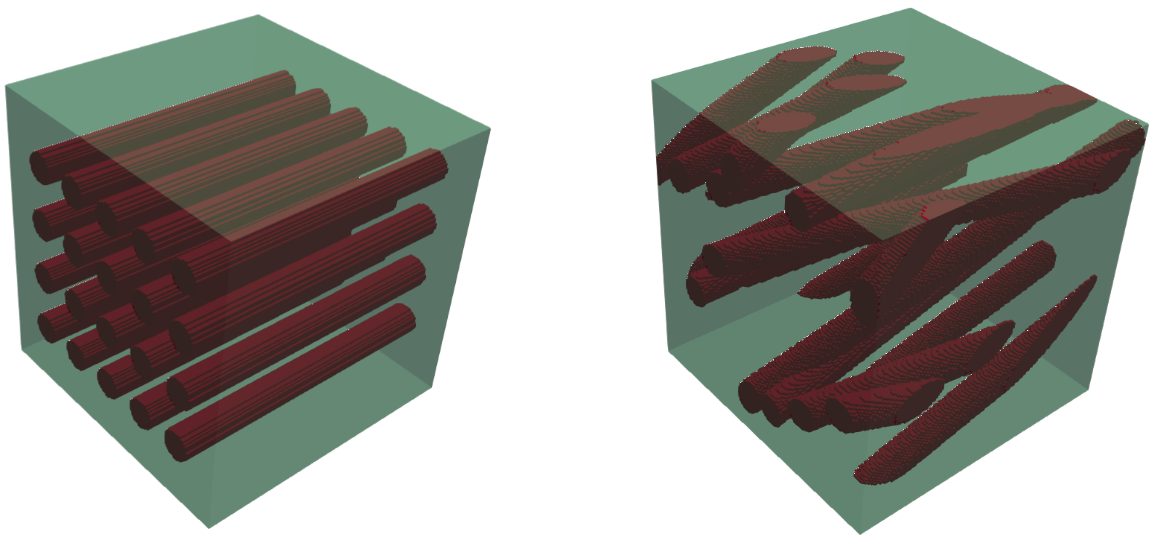


Figure A.3: Micro-structure of fiber arrangements corresponding to Micropp `benchmark-mic-4` [10] used to compare the Chamis Rule of Mixture with the FE coupling models. On the left, a parallel fiber arrangement composed. On the right, an arrangement of fibers with random directions.

simulation can be reproduce by running the Micropp benchmark: `benchmark-mic-4` described in the documentation [10].

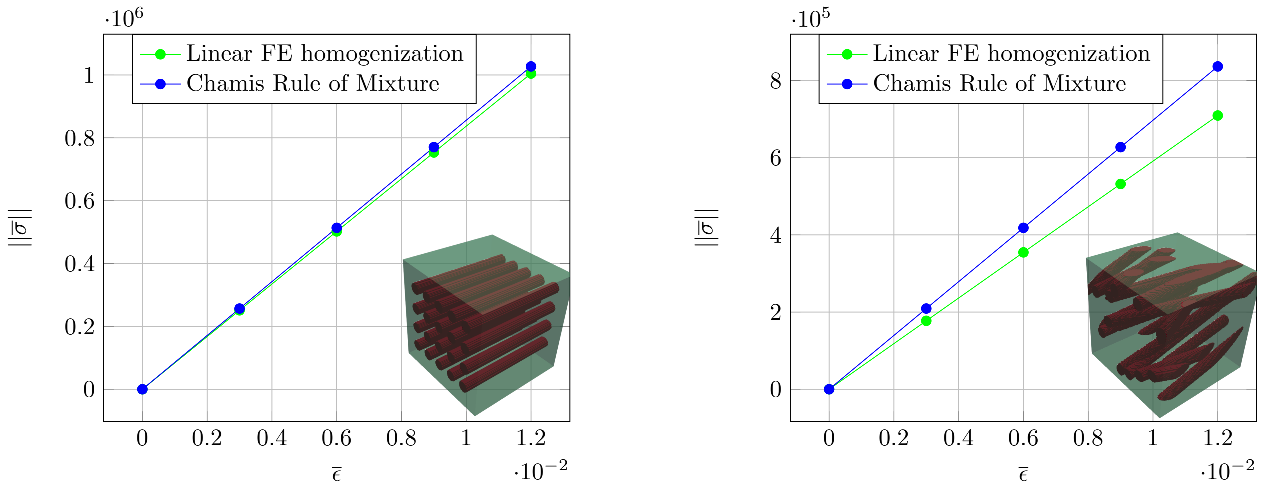


Figure A.4: Comparison between the FE linear coupling model and the Chamis Rule of Mixture of Micropp `benchmark-mic-4` [10] for two different micro-structures. One the left, parallel oriented fibers. On the right, random oriented fibers.

Acknowledgements

This research was thanks to the Barcelona Supercomputing Center (BSC) for providing the computational resources and to the European JTI-CleanSky2 Structural Health Monitoring, Manufacturing And Repair Technologies For Life Management Of Composite Fuselage project (SHERLOC) for the fundings. To acknowledge to the supervisors Dr. Mariano Vazquez and Dr. Sergio Oller for the opportunity of working in High-Performance Computing (HPC) applied to the aircraft field. To Guillaume Houzeaux for the support implementing some of the algorithms in the Alya code (like the weighted partitioning for improving the load balance). Thanks to Charles Moulinec, Judicaël Grasset, Stephen Longshaw, Ivan Scivetti and the STFC Daresbury Laboratory, in UK, for the collaboration established during 2019. To the 11th and 12th Severo Ochoa Mobility Grants awarded by BSC. To Jimmy Aguilar for improving, refactoring, testing, and optimizing Micropp for CPUs. Thanks to Miguel Zavala and Matias Rivero for the development of the contact mechanics algorithm in Alya, which has been used here to do impact simulations. Thanks to Herbert Owen for the fundings for going to SC19 Conference. To David Oks, Lucas Gasparino, Sarath Radhakrishnan, Anurag Surapaneni, Daniel Pastrana, Gerard Guillamet, Ricard Borrell, Guillermo Oyarsun, Samuel Gomez, Matias Avila, Alfonso Santiago, and for their technical support, knowledge transfer and fixing errors in this text and the motivation given.

Bibliography

[1] A. Baker, S. Dutton and D. Kelly. "Composite Materials for Aircraft Structures". American Institute of Aeronautics and Astronautics, Inc.

[2] "Code_Aster". Source Code. Version 13.6.

[3] "Code_Saturn". Source Code. Version 6.0.

[4] "Dealii". Source Code. Version 9.0.1.

[5] V. Eijkhout, E. Chow, R. van de Geijn. "Introduction to High Performance Scientific Computing". 2014.

[6] K. Matsui, K. Terada, K. Juge, "Two-scale finite element analysis of heterogeneous solids with periodic microstructures". Computers & Structures, 2014

[7] Rahul and Suvranu De. "An efficient coarse-grained parallel algorithm for global-local multiscale computations on massively parallel systems". Int. J. Numer. Meth. Engng., 2010.

[8] A. Klawonn, M. Lanser, O. Rheinbach. "FE2TI: Computational Scale Bridging for Dual-Phase Steels". Advances in Parallel Computing, 2015.

[9] D. Balzani, A. Gandhi, A. Klawonn, M. Lanser, O. Rheinbach, J. Schröder. "One-Way and Fully-Coupled FE2 Methods for Heterogeneous Elasticity and Plasticity Problems: Parallel Scalability and an Application to Thermo-Elastoplasticity of Dual-Phase Steels". Lecture Notes in Computational Science and Engineering, vol 113. Springer, Cham., 2016.

[10] G. Giuntoli, J. Aguilar, J. Grasset "Micropp Source Code". V3.11. github.com/GG1991/Micropp, 2019.

[11] G. Giuntoli, J. Aguilar, M. Vázquez, S. Oller and G. Houzeaux. "An FE2 multi-scale implementation for modeling composite materials on distributed architectures". Coupled Systems Mechanics, 8(2), 2018.

[12] G. Giuntoli, J. Grasset , A. Figueroa , C. Moulinec, M. Vázquez , G. Houzeaux, S. Longshaw & S. Oller. "Hybrid CPU/GPU FE2 multi-scale implementation coupling Alya and Micropp". Supercomputing Conference, 2019, Denver, US.

[13] P. Suquet. "Elements of homogenization for inelastic solid mechanics". Lecture Notes in Physics 272 pp. 193-278 Springer-Verlag, 1987.

[14] C. Johnson. "Numerical Solution of Partial Differential Equations by the Finite Element Method". 2009.

[15] K-J. Bathe. “Finite Element Procedures”. 2014.

[16] C. Miehe, J. Schröder, C. Bayreuther. “On the homogenization analysis of composite materials based on discretized fluctuations on the micro-structure”. C. Acta Mechanica, 2002.

[17] C. Miehe, A. Koch. “Computational micro-to-macro transitions of discretized micro-structures undergoing small strains”. Archive of Applied Mechanics (2002).

[18] G. Giuntoli (2018), “Microct: Scripts for Testing Micro-Scale Boundary Conditions” Source Code. V1.1.

[19] S. Hollister and N. Kikuchi. ”A comparison of homogenization and standard mechanics analyses for periodic porous composites.” Computational Mechanics 10(2): 73-95, 1992

[20] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout and W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini and H. Zhang, “PETSc Users Manual”, Argonne National Laboratory, 2015, ANL-95/11 - Revision 3.6.

[21] G. Giuntoli, “Microc Source Code”. github.com/GG1991/Microc, 2019.

[22] Barcelona Supercomputing Center (2019), “MareNostrum4 User’s Guide”.

[23] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Aris, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, J.M. Cela “Alya: Towards Exascale for Engineering Simulation Codes”. Cornell University Library. 2014.

[24] Barcelona Supercomputing Center (2019), “Power9 CTE User’s Guide”.

[25] G. Karypis and V. Kumar. (2009), “MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0”. University of Minnesota, Minneapolis, MN.

[26] Structural Health Monitoring, Manufacturing And Repair Technologies For Life Management Of Composite Fuselage project (SHERLOC). H2020 JTI-Clean Sky2 - Topic JTI-CS2-CPW-AIR-02-03 Advanced Technologies For More Affordable Composite Fuselage.

[27] E. Barbero. “Finite Element Analysis of Composite Materials”. CRC Press.

[28] Airbus Main Page: www.airbus.com.

[29] G. Lubin. “Editor- Handbook of Composites”. Van Nostrand Reinhold, 1982.

[30] E. Chavez (2012). “Mecánica del Medio Continuo”. CIMNE.

[31] M. Rivero (2018). PhD Thesis. “A Parallel Algorithm for Deformable Contact Problems”. Universitat Politècnica de Catalunya.

[32] J.C. Simo and T.J.R. Hughes. “Computational Inelasticity”. Springer 2002.

[33] G. Juárez-Luna, H. Méndez-Martínez and M.E. Ruiz-Sandoval. “An isotropic damage model to simulate collapse in reinforced concrete elements”. Latin American Journal of Solid and Structures, 2014.

[34] S. Oller. “Dinamica No Lineal”. CIMNE 2001.

[35] F. Feyel, J-L. Chabote. “FE2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials”. Comput. Methods Appl. Mech. Engrg., 1998.

[36] C. Chamis. “Mechanics of composite materials: past, present, and future”. J. Compos. Technol. Res. ASTM 1989; 11:3-14.