

UNIVERSIDADE FEDERAL FLUMINENSE

VICTOR WRIEDT SAPUCAIA

**AN EFFICIENT TIME DOMAIN IMAGE-BASED  
FINITE ELEMENT CUDA SOLVER TO  
COMPUTE EFFECTIVE WAVE SPEED IN  
POROUS MEDIA**

NITERÓI

2026

VICTOR WRIEDT SAPUCAIA

**AN EFFICIENT TIME DOMAIN IMAGE-BASED  
FINITE ELEMENT CUDA SOLVER TO  
COMPUTE EFFECTIVE WAVE SPEED IN  
POROUS MEDIA**

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação

Orientador:

RICARDO LEIDERMAN

Coorientador:

ANDRÉ MAUÉS BRABO PEREIRA

NITERÓI

2026

Ficha catalográfica automática - SDC/BCPV  
Gerada com informações fornecidas pelo autor

S241a Sapucaia, Victor Wriedt  
AN EFFICIENT TIME DOMAIN IMAGE-BASED FINITE ELEMENT CUDA  
SOLVER TO COMPUTE EFFECTIVE WAVE SPEED IN POROUS MEDIA /  
Victor Wriedt Sapucaia. - 2026.  
142 f.

Orientador: Ricardo Leiderman.  
Coorientador: André Maués Brabo Pereira.  
Tese (doutorado)-Universidade Federal Fluminense, Instituto  
de Computação, Niterói, 2026.

1. Física de Rocha Digital. 2. Transmissão de Pulso  
Ultrassônico. 3. Simulação de Ondas Elásticas. 4.  
Computação em GPU. 5. Produção intelectual. I. Leiderman,  
Ricardo, orientador. II. Pereira, André Maués Brabo,  
coorientador. III. Universidade Federal Fluminense. Instituto  
de Computação. IV. Título.

CDD - XXX

VICTOR WRIEDT SAPUCAIA

AN EFFICIENT TIME DOMAIN IMAGE-BASED FINITE ELEMENT CUDA SOLVER  
TO COMPUTE EFFECTIVE WAVE SPEED IN POROUS MEDIA

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação

BANCA EXAMINADORA

---

Prof. Dr. RICARDO LEIDERMAN - Orientador, UFF

---

Prof. Dr. ANDRÉ MAUÉS BRABO PEREIRA, UFF

---

Prof. Dr. DANIEL ALVES CASTELLO, UFRJ

---

Prof. Dr. BERNARDO FEIJÓ JUNQUEIRA, UERJ

---

Prof. Dr. PEDRO CORTEZ FETTER LOPES, UFF

---

Prof. Dr. LUIZ FERNANDO BEZ, UFF

Niterói

2026



*The most dangerous phrase in the language is, "We've always done it this way".*

*(Grace Hopper)*

# Agradecimentos

Agradeço primeiramente a Deus, por ter me dado forças para concluir este projeto, sem Ele nada seria possível.

À minha esposa, Stephanie, pelo companheirismo, incentivo, apoio, e paciência em todas as realizações dos meus sonhos. Essa conquista também é sua, meu amor.

Agradeço ao meu pai Helio e minha mãe Mirna por todo amor e cuidados no passado para que hoje eu possa ter uma trajetória repleta de oportunidades. Vocês são os pilares da minha formação como ser humano. Dedico também aos meus avós, irmãos, padrasto e madrasta.

Sou grato ao professor e amigo Ricardo pelo incentivo durante todo o projeto. Sua motivação tem sido fundamental no meu desenvolvimento. Você esteve sempre disponível a compartilhar todo o seu vasto conhecimento. Agradeço também ao meu professor, amigo e coorientador, André, pelas correções e ensinamentos neste projeto.

Agradeço a esta universidade e todo o corpo de docentes do Instituto de Computação pela dedicação e cooperação. Agradeço aos colegas do Laboratório de Computação Científica, foi uma grata surpresa conhecê-los e espero ainda muitas parcerias em trabalhos futuros, em especial ao Rafael, por ter trabalhado comigo desde o início do projeto.

# Resumo

Técnicas de transmissão de pulso ultrassônico (Ultrasonic Pulse Transmission – UPT) são essenciais para a caracterização não invasiva de formações rochosas na indústria do petróleo. Embora os ensaios de UPT em laboratório forneçam informações valiosas, desafios como a preparação de amostras e a precisão de medição persistem, especialmente para amostras rochosas com geometrias complexas ou superfícies irregulares. Como alternativa aos ensaios laboratoriais convencionais, simulações numéricas baseadas em modelos de rocha digital obtidos por microtomografia computadorizada de raios-X em escala microscópica ( $\mu$ CT) permitem realizar essa caracterização de forma não destrutiva e repetível. Contudo, os *solvers* de elementos finitos convencionais exigem a montagem explícita, em formato esparsa, da matriz de rigidez global. Essa estratégia torna proibitiva a avaliação de modelos de larga escala devido ao consumo de memória e ao tempo de simulação. Esta tese propõe um *solver* eficiente e massivamente paralelo de elementos finitos no domínio do tempo, focado exclusivamente na fase sólida, para simulações de UPT em amostras de rocha digital de larga escala obtidas por  $\mu$ CT. As principais contribuições deste trabalho são: (i) uma formulação elemento por elemento sem matriz (*matrix-free*) que elimina a montagem explícita da matriz de rigidez global, reduzindo o consumo de memória em relação às abordagens convencionais; e (ii) uma implementação em CUDA massivamente paralela, capaz de lidar com modelos com mais de  $10^8$  graus de liberdade em uma única GPU, alcançando *speedups* de até  $147.57\times$  em relação a uma implementação de referência em CPU. O *solver* proposto foi validado com soluções analíticas e com dados experimentais de ensaios de UPT em laboratório em diferentes meios porosos, como arenitos e carbonatos. As velocidades das ondas compressoriais (P) e cisalhantes (S) obtidas nas simulações são utilizadas para inferir diretamente os módulos elásticos efetivos. Esta pesquisa representa um avanço concreto no estado da arte ao viabilizar simulações massivamente paralelas de elementos finitos no domínio do tempo, com impacto direto em aplicações de física de rocha digital, ciência dos materiais e engenharia de petróleo.

**Palavras-chave:** física de rocha digital, transmissão de pulso ultrassônico, simulação de ondas elásticas, MEF baseado em imagem, método sem matriz, computação em GPU, paralelização massiva.

# Abstract

Ultrasonic pulse transmission (UPT) techniques are essential for the non-invasive characterization of rock formations in the petroleum industry. Although laboratory UPT tests provide valuable information, challenges such as sample preparation and measurement accuracy persist, especially for rock samples with complex geometries or irregular surfaces. As an alternative to conventional laboratory testing, numerical simulations based on digital rock models obtained from micro-scale X-ray computed tomography ( $\mu$ CT) allow this characterization to be performed non-destructively and reproducibly. However, conventional finite element solvers require explicit assembly of the global stiffness matrix in sparse format. This strategy renders the evaluation of large models prohibitive due to memory usage and simulation time. This thesis proposes an efficient, massively parallel finite element solver in the time domain, focused exclusively on the solid phase, for UPT simulations in large digital rock samples obtained by  $\mu$ CT. The main contributions of this work are: (i) a matrix-free element-by-element formulation that eliminates explicit global stiffness matrix assembly, reducing memory consumption compared to conventional approaches; and (ii) a massively parallel CUDA implementation capable of handling models with more than  $10^8$  degrees of freedom on a single GPU, achieving speedups of up to  $147.57\times$  over a single-core CPU reference. The proposed solver was validated against analytical solutions and experimental data from laboratory UPT tests on various porous media such as sandstones and carbonates. The predicted P- and S-wave velocities are used to directly infer effective elastic moduli. This research represents a concrete advancement in the state of the art by enabling massively parallel time domain finite element simulations, with direct impact on applications in digital rock physics, materials science, and petroleum engineering.

**Keywords:** Digital rock physics, ultrasonic pulse transmission, elastic wave simulation, image-based FEM, matrix-free method, GPU computing, massive parallelization.

# List of Figures

|    |                                                                                                                                                                                                                                                                         |    |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1  | Digital sample acquisition using Micro-scale X-ray Computed Tomography ( $\mu$ CT) workflow: sample preparation, image acquisition with individual radiographic projections, reconstructing the volume, and segmentation. . .                                           | 13 |
| 2  | Laboratory setup for ultrasonic pulse transmission showing transmitter and receiver transducers, time measurement display on oscilloscope, and calculation of seismic wave velocities through a rock sample. . . . .                                                    | 19 |
| 3  | Workflow showing the integration of digital rock physics and laboratory testing: from sample digitization via $\mu$ CT imaging, image processing and segmentation, numerical simulation of ultrasonic wave propagation, to experimental velocity determination. . . . . | 21 |
| 4  | Plane P-wave with particle motion parallel to the direction of propagation. The wave causes compressional deformation as it travels through the medium.                                                                                                                 | 24 |
| 5  | Plane S-wave with particle motion perpendicular to the direction of propagation. The transverse motion causes shear deformation in the medium, typically arriving after P-waves with lower phase velocity. . . . .                                                      | 24 |
| 6  | Example of element and node mapping in image-based porous media. . . .                                                                                                                                                                                                  | 35 |
| 7  | Schematic representation of the local numbering of nodes in a 3D voxel element. . . . .                                                                                                                                                                                 | 37 |
| 8  | Example of active node mapping in image-based porous media: (a) direct node mapping; (b) active node mapping; and (c) vector with active nodes mapped. . . . .                                                                                                          | 38 |
| 9  | Flowchart of the Active Node Mapping algorithm, including periodic boundary condition (PBC) mapping. . . . .                                                                                                                                                            | 40 |
| 10 | Static convergence of effective properties: (a) different volume sizes of interest; (b) convergence of effective properties according to boundary conditions. Source: Adapted from Nguyen et al. [85]. . . . .                                                          | 41 |

|    |                                                                                                                                                                                                                                                |    |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 11 | On the left, is a square unitary cell with periodic boundary conditions around all faces. On the right, is a rectangular unitary cell with periodic boundary conditions on the faces perpendicular to the Gaussian wave's propagation. . . . . | 42 |
| 12 | Illustration of the digital rock 2D model in a numerical Ultrasonic Pulse Transmission test using additional layers. . . . .                                                                                                                   | 43 |
| 13 | Example of the average displacement versus time sensors curves in a P-wave simulation. . . . .                                                                                                                                                 | 44 |
| 14 | Comparison of CPU and GPU architectures, illustrating differences in core organization and computational resource allocation. Source: Adapted from NVIDIA [96]. . . . .                                                                        | 47 |
| 15 | Summarized Host-Device execution model. All time-critical operations are encapsulated within the GPU loop to minimize PCI-e overhead. . . . .                                                                                                  | 49 |
| 16 | Graphical User Interface (GUI) developed for the Finite Element Method (FEM) UPT simulator. . . . .                                                                                                                                            | 53 |
| 17 | Implementation evolution and validation chain. Each node represents an implementation stage; the label below indicates the reference used for cross-validation. . . . .                                                                        | 54 |
| 18 | Example of the JSON input file used to configure the dynamic image-based FEM solver. . . . .                                                                                                                                                   | 56 |
| 19 | Diagram of possible implementation options for EbE. . . . .                                                                                                                                                                                    | 58 |
| 20 | Wave propagation in a one-dimensional rod: (a) model, geometry, and forces; (b) Heaviside loading pattern; and (c) displacement ratio at end-node and mid-span. . . . .                                                                        | 73 |
| 21 | Numerical normalized displacement at the end node and mid-span in the finite homogeneous rod models with varying space discretization. . . . .                                                                                                 | 76 |
| 22 | Zoomed of end node displacements and analytical solution in models with varying space discretization. . . . .                                                                                                                                  | 76 |
| 23 | Zoomed of mid-span displacements and analytical solution in models with varying space discretization. . . . .                                                                                                                                  | 77 |
| 24 | Overall relative error as a function of CFL factor for refined meshes. . . . .                                                                                                                                                                 | 78 |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |    |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 25 | Numerical normalized displacement at the end node and mid-span in the finite homogeneous rod models with discretization. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 79 |
| 26 | Zoomed of end node displacements and analytical solution in models with varying time discretization. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 80 |
| 27 | Zoomed of mid-span displacements and analytical solution in models with varying time discretization. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 80 |
| 28 | Error map for rod simulation using a $100 \times 10 \times 10$ mesh, while varying the CFL factor through changes in time step $\Delta t$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 81 |
| 29 | Overall relative error as a function of CFL factor for models with varying time discretization. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 82 |
| 30 | Average processing time per simulation with varying time discretization. . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 82 |
| 31 | Binary two-dimensional slices of the digital rock dataset with pore space in black, and solid phase in white: (a) Berea Brown (BB), (b) Berea Gray (BG), and (c) Crab Orchard (CO). . . . .                                                                                                                                                                                                                                                                                                                                                                                                                            | 83 |
| 32 | The Gaussian pulses and frequencies used in the model with the 200-pixel buffer zone, time step size $\Delta t = 3.182\text{e-}9$ s, and spatial discretization size $\Delta L = 2.00\text{e-}5$ m. . . . .                                                                                                                                                                                                                                                                                                                                                                                                            | 84 |
| 33 | The Gaussian pulses and frequencies used in the model with the 400-pixel buffer zone, time step size $\Delta t = 2.500\text{e-}9$ s, and spatial discretization size $\Delta L = 2.00\text{e-}5$ m. . . . .                                                                                                                                                                                                                                                                                                                                                                                                            | 85 |
| 34 | Effective velocities and average displacement in sensors S1 and S2 for various frequencies of Gaussian pulse using a 200-pixel buffer zone model. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 86 |
| 35 | Effective velocities and average displacement in sensors S1 and S2 for various frequencies of Gaussian pulse using a 400-pixel buffer zone model. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 87 |
| 36 | Displacement maps for 200 and 400-pixel buffer zone models at frequencies $f = 1.51$ MHz and $f = 1.44$ MHz showing: (a) average displacement in sensors S1 and S2 in the 200-pixel model; (b) average displacement in sensors S1 and S2 in the 400-pixel model; (c) displacement map for $t_1$ in the 200-pixel model; (d) displacement map for $t_2$ in the 200-pixel model; (e) displacement map for $t_3$ in the 200-pixel model; (f) displacement map for $t_1$ in the 400-pixel model; (g) displacement map for $t_2$ in the 400-pixel model; and (h) displacement map for $t_3$ in the 400-pixel model. . . . . | 88 |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                          |     |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 37 | Displacement of the $x$ -component for the homogeneous $300 \times 100 \times 100$ benchmark model computed with the CUDA_EW solver, showing snapshots at time steps 200, 400, 600, and 800. The color scale represents the magnitude of displacement. The inlet (source), sensors S1 and S2, and the outlet face are indicated. . . . .                                                                                                                 | 91  |
| 38 | Displacement of the $y$ -component (transverse shear) for the homogeneous $300 \times 100 \times 100$ benchmark model computed with the CUDA_EW solver at time steps 200, 400, 600, and 800. The wave propagates along the $x$ -axis, and the $y$ -displacement field characterizes the shear wave transverse to the propagation direction. . . . .                                                                                                      | 92  |
| 39 | Displacement of the $z$ -component (transverse shear) for the homogeneous $300 \times 100 \times 100$ benchmark model computed with the CUDA_EW solver at time steps 200, 400, 600, and 800. The $z$ -displacement field is analogous to the $y$ -component and confirms the expected symmetric shear wave response in both transverse directions. . . . .                                                                                               | 93  |
| 40 | Workflow of the digital rock benchmarks, illustrating the parallel computation paths: the dynamic solver yielding $V_P, V_S$ with subsequent conversion to $K, G$ , and the static solver yielding $K, G$ with subsequent conversion to $V_P, V_S$ . . . . .                                                                                                                                                                                             | 96  |
| 41 | 3D visualizations of the Fontainebleau sandstone voxel-based domains used in the numerical simulations, showing the grain and pore network distribution at different scales: (a) $150^3$ voxels; (b) $200^3$ voxels; (c) $250^3$ voxels; and (d) $288 \times 288 \times 300$ voxels (original sample). The axes indicate spatial dimensions in voxel units, highlighting the increase in total degrees of freedom across the varied sub-volumes. . . . . | 97  |
| 42 | 3D visualizations of the Berea sandstone voxel-based domains used in the benchmark analyses, illustrating the pore-structure heterogeneity across four model scales: (a) $250^3$ voxels; (b) $300^3$ voxels; (c) $350^3$ voxels; and (d) $400^3$ voxels. The visualizations emphasize the volumetric expansion of the sample used to evaluate the stability of effective elastic properties as a function of the domain size. . . . .                    | 102 |



- 
- 43 3D visualizations of the Grosmont carbonate voxel-based domains used in the benchmark analyses, showing the complex vuggy porosity [114] and solid matrix distribution at different scales: (a)  $250^3$  voxels; (b)  $300^3$  voxels; (c)  $350^3$  voxels; and (d)  $400^3$  voxels. The models highlight the significant increase in computational complexity and total degrees of freedom required for the characterization of highly heterogeneous carbonate samples. . . . . 106
- 44 Four-node quadrilateral finite element (pixel) used in 2D image-based analysis. The element is defined by its corner nodes and linear shape functions for strain-displacement computations. . . . . 131
- 45 Eight-node hexahedral finite element (voxel) used in 3D image-based analysis. The element is defined by its eight corner nodes and linear shape functions for strain-displacement computations in three-dimensional elasticity problems. 131
- 46 Gaussian pulse used as source-time function in the UPT simulations. . . . 133
- 47 Relation between temporal standard deviation  $\sigma$  and full width at half maximum (FWHM). . . . . 133

# List of Tables

|    |                                                                                                                                               |     |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 1  | Memory allocation for sensor displacement history. . . . .                                                                                    | 45  |
| 2  | Description of the JSON fields used to build the model of Algorithm 2. . .                                                                    | 56  |
| 3  | Memory allocation details for different EbE implementations. . . . .                                                                          | 59  |
| 4  | Memory allocated in image-based Element-by-Element implementation. . .                                                                        | 60  |
| 5  | Comparison between CUDA solver modes. . . . .                                                                                                 | 62  |
| 6  | Implemented solver modes in the current implementation. . . . .                                                                               | 71  |
| 7  | Hardware and software configuration of the workstation used in the numerical experiments. . . . .                                             | 73  |
| 8  | Adjustable mesh discretization with a constant time step $\Delta t$ . . . . .                                                                 | 75  |
| 9  | Adjustable time step $\Delta t$ with a constant mesh discretization. . . . .                                                                  | 78  |
| 10 | Characteristics of investigated rocks samples. . . . .                                                                                        | 83  |
| 11 | Effective velocities results and times metrics for analyzing rock samples. . .                                                                | 89  |
| 12 | Execution time, speedup, and memory consumption for the evaluated solver variants on the model $300 \times 100 \times 100$ benchmark. . . . . | 90  |
| 13 | Average wall-clock time per time step for the five solver variants applied to Fontainebleau sandstone sub-volumes of increasing size. . . . . | 94  |
| 14 | Geometric and computational properties of the Fontainebleau samples. . .                                                                      | 98  |
| 15 | Static elastic properties ( $K, G$ ) calculated for the Fontainebleau samples in different directions. . . . .                                | 99  |
| 16 | Velocity properties ( $V_P, V_S$ ) for the Fontainebleau samples along the $x, y$ , and $z$ axes using dynamic CUDA-based analyses. . . . .   | 100 |

|    |                                                                                                                                                                                                                                                       |     |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 17 | Fontainebleau: comparison of Bulk ( $K$ ) and Shear ( $G$ ) moduli between this work the Andrä et al. [14] report, and static solver [19]. Andrä values correspond to the $288 \times 288 \times 300$ full-scale model with porosity $\phi = 0.147$ . | 101 |
| 18 | Geometric and computational properties of the Berea samples. . . . .                                                                                                                                                                                  | 103 |
| 19 | Static elastic properties ( $K, G$ ) calculated for the Berea samples in different directions. . . . .                                                                                                                                                | 103 |
| 20 | Velocity properties ( $V_P, V_S$ ) for the Berea samples along the $x, y$ , and $z$ axes.                                                                                                                                                             | 104 |
| 21 | Berea: comparison of Bulk ( $K$ ) and Shear ( $G$ ) moduli between this work, the Andrä et al. [14] report, and static solver [19]. Andrä values correspond to a $400^3$ subsample with porosity $\phi = 0.184$ . . . . .                             | 105 |
| 22 | Geometric and computational properties of the Grosmont samples. . . . .                                                                                                                                                                               | 107 |
| 23 | Static elastic properties ( $K, G$ ) calculated for the Grosmont samples in different directions. . . . .                                                                                                                                             | 107 |
| 24 | Velocity properties ( $V_P, V_S$ ) for the Grosmont samples along the $x, y$ , and $z$ axes. . . . .                                                                                                                                                  | 108 |
| 25 | Grosmont: comparison of Bulk ( $K$ ) and Shear ( $G$ ) moduli between this work, the Andrä et al. [14] report, and static solver [19]. Andrä values correspond to a $400^3$ subsample with porosity $\phi = 0.247$ . . . . .                          | 109 |
| 26 | Comparison of static (CHFEM) and dynamic (FEM-CUDA) total execution times for the Fontainebleau, Berea, and Grosmont benchmark samples. The static time corresponds to the largest value measured across the tested execution environments. . . . .   | 110 |
| 27 | Conversion formulas for homogeneous isotropic linear elastic material properties, adapted from Mavko, Mukerji, and Dvorkin [2]. . . . .                                                                                                               | 127 |

# List of Abbreviations and Acronyms

**$\mu$ CT** Micro-scale X-ray Computed Tomography

**ANM** Active Node Mapping

**CFL** Courant-Friedrichs-Lewy

**CHFEM** Computational Homogenization with the image-based Finite Element Method

**CPU** Central Processing Unit

**CUDA** Compute Unified Device Architecture

**DNM** Direct Node Mapping

**DOF** degrees of freedom

**DOFMAP** Degree of Freedom Map

**DRP** Digital Rock Physics

**EbE** Element-by-Element

**FDM** Finite Difference Method

**FEM** Finite Element Method

**FWHM** Full Width at Half Maximum

**GPU** Graphics Processing Unit

**GUI** Graphical User Interface

**HPC** High-Performance Computing

**NbN** Node-by-Node

**ODE** Ordinary Differential Equations

**OpenMP** Open Multi-Processing

**PBC** Periodic Boundary Condition

**PML** Perfectly Matched Layer

**RAM** Random Access Memory

**RAW** Raw image format

**RVE** Representative Volume Element

**SIMT** Single Instruction, Multiple Threads

**SM** Streaming Multiprocessor

**UPT** Ultrasonic Pulse Transmission

**VRAM** Video Random Access Memory

# Contents

|          |                                                                  |           |
|----------|------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                              | <b>12</b> |
| 1.1      | Context and Motivation . . . . .                                 | 14        |
| 1.2      | Goals . . . . .                                                  | 15        |
| 1.3      | Organization . . . . .                                           | 16        |
| <b>2</b> | <b>Literature Review</b>                                         | <b>17</b> |
| 2.1      | Elastic Properties of Porous Media . . . . .                     | 17        |
| 2.2      | Ultrasonic Pulse Transmission Technique . . . . .                | 18        |
| 2.3      | Digital Rock Physics and Ultrasonic Pulse Transmission . . . . . | 20        |
| <b>3</b> | <b>Mathematical Modeling</b>                                     | <b>23</b> |
| 3.1      | Theory of Wave Propagation in Elastic Media . . . . .            | 23        |
| 3.2      | Finite Element Method . . . . .                                  | 26        |
| 3.2.1    | Formulation . . . . .                                            | 27        |
| 3.2.2    | Assembly-Free and Stiffness-Displacement Product . . . . .       | 28        |
| 3.2.3    | Lumped Mass Matrix and Inversion . . . . .                       | 29        |
| 3.3      | Stability and Accuracy of the Numerical Scheme . . . . .         | 31        |
| 3.3.1    | Explicit Method and the Critical Time Step . . . . .             | 31        |
| 3.3.2    | Spatial Dispersion Criterion . . . . .                           | 33        |
| 3.4      | Image-Based Finite Element Modeling . . . . .                    | 34        |
| 3.4.1    | Element and Node Mapping . . . . .                               | 35        |
| 3.4.2    | Direct Node Mapping . . . . .                                    | 36        |

|          |                                                         |           |
|----------|---------------------------------------------------------|-----------|
| 3.4.3    | Active Node Mapping . . . . .                           | 38        |
| 3.4.4    | Periodic Boundary Conditions . . . . .                  | 41        |
| 3.5      | Ultrasonic Pulse Transmission Model . . . . .           | 42        |
| 3.6      | Parallel Computing with CUDA . . . . .                  | 45        |
| 3.6.1    | The GPU Architecture . . . . .                          | 46        |
| 3.6.2    | Host and Device Model . . . . .                         | 48        |
| 3.6.3    | Memory Hierarchy . . . . .                              | 49        |
| 3.6.4    | CUDA Kernels . . . . .                                  | 50        |
| 3.6.5    | Race Conditions and Atomic Operations . . . . .         | 51        |
| <b>4</b> | <b>Proposed Implementations</b>                         | <b>53</b> |
| 4.1      | Time-Domain Image-Based Finite Element Method . . . . . | 54        |
| 4.2      | Element-by-Element Algorithm . . . . .                  | 57        |
| 4.3      | Active Node Mapping Generation . . . . .                | 61        |
| 4.4      | CUDA Solver Architecture . . . . .                      | 61        |
| 4.4.1    | CUDA Standard vs. CUDA Element-Wise . . . . .           | 62        |
| 4.4.2    | Benchmarking and Cross-Validation Methodology . . . . . | 63        |
| 4.5      | CUDA Kernels . . . . .                                  | 63        |
| 4.5.1    | DOF Mapping Kernels . . . . .                           | 63        |
| 4.5.2    | Internal Force Kernels . . . . .                        | 64        |
| 4.5.3    | Mass Kernels . . . . .                                  | 66        |
| 4.5.4    | Time Marching Kernels . . . . .                         | 67        |
| 4.5.5    | History Kernels . . . . .                               | 68        |
| 4.5.6    | Pulse Kernels . . . . .                                 | 68        |
| 4.5.7    | Utility Kernels . . . . .                               | 69        |
| 4.6      | Summary of the Implemented Solvers . . . . .            | 69        |
|          | Mode 1: Serial CPU Solver . . . . .                     | 69        |

|                                                                        |           |
|------------------------------------------------------------------------|-----------|
| Mode 2: OpenMP CPU Solver . . . . .                                    | 70        |
| Mode 3: CUDA GPU Solver . . . . .                                      | 70        |
| Mode 4: CUDA GPU Element-wise Solver . . . . .                         | 70        |
| Mode 5: Serial CPU Element-wise Solver . . . . .                       | 70        |
| <b>5 Numerical Results</b>                                             | <b>72</b> |
| 5.1 Finite Homogeneous Rod . . . . .                                   | 73        |
| 5.1.1 Convergence of Space Discretization . . . . .                    | 75        |
| 5.1.2 Convergence of Time Discretization . . . . .                     | 78        |
| 5.2 MicroCT 2D Images . . . . .                                        | 83        |
| 5.2.1 Gaussian Pulse and Buffer Zone . . . . .                         | 84        |
| 5.2.2 Effective Velocities of Digital Cores . . . . .                  | 89        |
| 5.3 Performance and Speedup Analysis . . . . .                         | 89        |
| 5.3.1 Homogeneous Benchmark Model . . . . .                            | 89        |
| 5.3.2 Scalability on a Digital Rock: Fontainebleau Sandstone . . . . . | 93        |
| 5.4 Validation against Digital Rock Benchmarks . . . . .               | 94        |
| 5.4.1 Fontainebleau Sandstone . . . . .                                | 96        |
| 5.4.1.1 Static Results (CHFEM) . . . . .                               | 98        |
| 5.4.1.2 Dynamic Results (FEM-CUDA) . . . . .                           | 99        |
| 5.4.1.3 RVE Convergence . . . . .                                      | 100       |
| 5.4.1.4 Comparison with the Andrä et al. Benchmark . . . . .           | 100       |
| 5.4.2 Berea Sandstones . . . . .                                       | 101       |
| 5.4.2.1 Static Results (CHFEM) . . . . .                               | 103       |
| 5.4.2.2 Dynamic Results (FEM-CUDA) . . . . .                           | 104       |
| 5.4.2.3 RVE Convergence . . . . .                                      | 104       |
| 5.4.2.4 Comparison with the Andrä et al. Benchmark . . . . .           | 105       |
| 5.4.3 Grosmont Carbonate . . . . .                                     | 105       |



---

|          |                                                                                                  |            |
|----------|--------------------------------------------------------------------------------------------------|------------|
| 5.4.3.1  | Static Results (CHFEM) . . . . .                                                                 | 107        |
| 5.4.3.2  | Dynamic Results (FEM-CUDA) . . . . .                                                             | 107        |
| 5.4.3.3  | RVE Convergence . . . . .                                                                        | 108        |
| 5.4.3.4  | Comparison with the Andrä et al. Benchmark . . . . .                                             | 108        |
| 5.4.4    | Computational Performance: Static versus Dynamic Execution Times                                 | 109        |
| <b>6</b> | <b>Conclusions and Future Work</b>                                                               | <b>111</b> |
| 6.1      | Conclusions . . . . .                                                                            | 111        |
| 6.2      | Future Work . . . . .                                                                            | 112        |
| 6.3      | Final Remarks . . . . .                                                                          | 114        |
|          | <b>REFERENCES</b>                                                                                | <b>115</b> |
|          | <b>Appendix A - Homogeneous Isotropic Materials</b>                                              | <b>127</b> |
|          | <b>Appendix B - Analytical Solutions for Local Image-Based 2D and 3D Finite Element Matrices</b> | <b>128</b> |
|          | <b>Appendix C - Gaussian Pulse</b>                                                               | <b>132</b> |

# 1 Introduction

Ultrasonic Pulse Transmission (UPT) techniques represent an important non-invasive tool in the petroleum industry to explore and characterize rock formations. These techniques have proven to be valuable for analyzing rock samples under laboratory conditions [1]. The samples can be measured under dry and reservoir pressure conditions. Under dry conditions, UPT can be used to measure the velocity and attenuation of ultrasonic waves in rock samples. These measurements are intrinsically linked to the effective elastic properties of rocks [2] and can therefore serve as proxies for inferring key petrophysical parameters such as porosity and permeability. Furthermore, under reservoir pressure conditions, UPT can be used to monitor dynamic changes in rock properties caused by fluid injection or production (4D seismic) [3].

UPT works by sending a pulse of ultrasonic waves into the rock sample and measuring the time the pulse takes to travel through the sample and achieve the transducer [4, 5]. The velocity of ultrasonic waves is generally inversely related to rock porosity, so higher velocities tend to indicate lower porosity. Sometimes, UPT tests can be challenging due to various factors, including sample preparation and measurement accuracy. Sample preparation becomes particularly demanding for rocks with complex geometries or irregular surfaces. This process requires careful grinding and polishing to guarantee uniform contact with the transducers. Moreover, achieving precise measurements of ultrasonic velocities is subject to several sources of uncertainty, including transducer coupling, signal noise, and sample heterogeneities such as microcracks. In this context, numerical simulation can emerge as a valuable tool to address the challenges associated with laboratory-based UPT tests. By simulating the propagation of ultrasonic waves through digital rock models, researchers can gain insights into the complex relationships between acoustic properties, rock microstructure, and stress conditions. Additionally, numerical simulations can also provide valuable information about the sensitivity of ultrasonic measurements to various factors, such as transducer placement, sample geometry, and fluid saturation. This work is focused on measuring the rock velocities in digital dry rock samples using UPT and

numerical simulations on domestic Graphics Processing Units (GPUs).

During the late 20th century, the non-destructive imaging technique of micro-scale X-ray computed tomography ( $\mu$ CT or microCT) became essential to understand the internal structures of rock samples or other materials [6]. The  $\mu$ CT can be used to create 3D images of the materials at a microscopic scale. This imaging process commonly follows a systematic workflow, as in Figure 1, encompassing multiple essential steps. Starting with sample preparation, the procedure advances through image acquisition, reconstruction, and segmentation. This interconnected series of steps, in the petroleum industry, is part of a relatively new field called Digital Rock Physics (DRP).

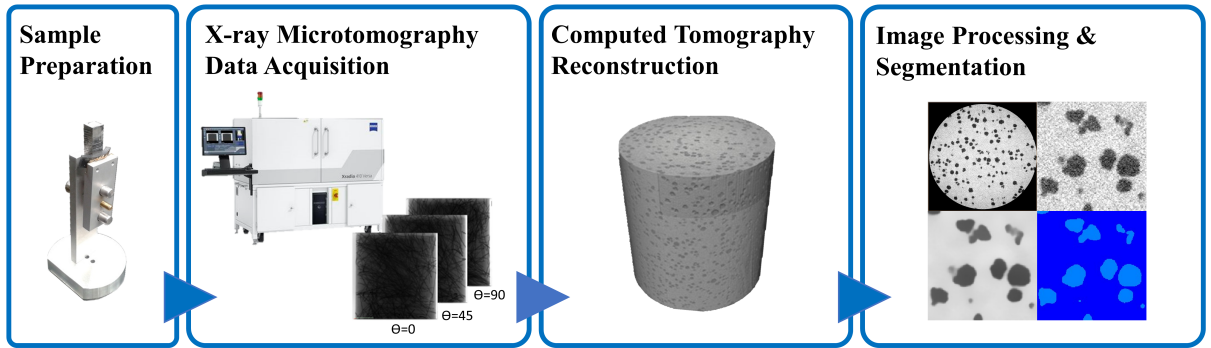


Figure 1: Digital sample acquisition using  $\mu$ CT workflow: sample preparation, image acquisition with individual radiographic projections, reconstructing the volume, and segmentation.

Within DRP, a limited number of studies focus on the simulation of UPT to estimate effective elastic properties. The review by Saenger [7] discusses different numerical approaches to determine those effective elastic values. Many of the studies found in the literature rely on the Finite Difference Method (FDM) to simulate wave propagation in porous media, notably Saenger, Krüger, and Shapiro [8] and Saenger, Krüger, and Shapiro [9], complemented by related studies [10, 11, 12, 13]. These previous studies were limited to small model sizes due to the high computational demands in memory and CPU time. In the study by Andrä et al. [14], simulations of size  $724^3$  and  $1024^3$  were subdivided or coarsened to achieve the reported results. Simulations of this scale demand high-performance computing infrastructure that is not available to the broader research community. To overcome these restrictions in similar scenarios, strategies have been used to increase performance, such as matrix-free methods [15, 16], which involve solving numerical problems without explicitly storing the entire sparse matrix. Furthermore, parallel computing techniques have been harnessed, enabling multiple processors to collaborate simultaneously. These methodologies have found application across various

computational methods on personal computers. On this basis, we aim to develop an efficient time-domain image-based finite element solver to perform the UPT simulation in a digital rock to estimate effective elastic properties. Subsequently, we use matrix-free and parallel computing strategies from our previous studies, such as Lopes et al. [17] and Bez et al. [18], as well as [19, 20, 21]. These strategies are capable of simulating physical phenomena using digital images with more than  $10^8$  degrees of freedom by using only domestic GPUs.

More recent studies show that this research direction remains active. Faisal et al. [22] validated numerical elastic-moduli estimates against acoustic experiments on multi-resolution CT-scanned carbonate samples. Bucher, Zyserman, and Monachesi [23] and Campos and Gracie [24] extended FEM-based dynamic wave propagation to coupled porous-fluid media, reporting convergence rates for their implementations. Ichimura et al. [25] demonstrated GPU-accelerated voxel FEM for large-scale explicit elastic wave propagation, which corresponds to a similar computational setting addressed in this work.

## 1.1 Context and Motivation

Numerical simulations in dry rock environments serve as a crucial initial step toward conducting analyses at reservoir pressure conditions with fluids. This preliminary stage gives useful information on the structural and mechanical properties of the rock formation and provides the basis for the subsequent simulations that include fluid dynamics. Such a progressive approach ensures a comprehensive and accurate representation of real-world scenarios, aiding in the optimization of well performance and the assessment of potential reservoir challenges.

In Lopes et al. [19], the improved performance in GPU simulations was limited to static analyses. To advance in the development of effective material properties, the next step involves simulating these materials under dynamic conditions, as demonstrated in this study. Motivated by this gap, our focus sharpens towards the development of an efficient time-domain image-based finite element solver for UPT simulations in digital dry rocks. Inspired by the state of the art of matrix-free and parallel computing strategies, we aspire to make this technology accessible to every researcher, rendering it feasible even on domestic GPUs for simulations with degrees of freedom exceeding  $10^8$ .

## 1.2 Goals

This work aims to develop an efficient time-domain image-based finite element solver to perform the UPT test on large rock images acquired via  $\mu$ CT. Certainly, incorporating parallel computing using GPU into the implementation is a crucial aspect, particularly for enhancing the efficiency of complex porous media simulations that model only the solid phase. As the GPU memory is relatively smaller than the CPU, matrix-free methods such as Element-by-Element (EbE) approach will also be employed. These techniques aim to minimize memory usage and facilitate the handling of large-scale models.

The specific goals of this study include:

1. Development of a robust algorithm that generates a structured finite element mesh of the solid phase of porous media and performs the UPT test on the CPU. This algorithm incorporates optimization schemes to increase the computational efficiency of the dynamic solver based on NVIDIA's Compute Unified Device Architecture (CUDA) without compromising accuracy. The tool was designed with an emphasis on usability, requiring minimal computational expertise to configure and run. This design choice allows researchers to spend more time analyzing simulations than on the technical aspects of the tool.
2. Validate the proposed solver against known analytical solutions, experimental data, or established numerical benchmarks to assess the accuracy and reliability of the developed solver. Perform sensitivity analyses to explore how variations in ultrasonic pulse and time step parameters influence the calculated elastic properties. This will provide insights into the behavior of porous media models and help identify critical factors influencing predictions.
3. Apply the developed solver to real-world case studies involving diverse porous media. This includes  $\mu$ CT images of sandstones and carbonates. Compare the effective wave velocities estimated by the proposed dynamic solver against those obtained by the static image-based finite element solver [19]. Highlight the strengths and weaknesses of the proposed solver.
4. Develop and implement multiple solver modes that progressively exploit parallel computing resources: a serial CPU baseline (C, single-thread), a shared-memory parallel CPU version using OpenMP, and two CUDA GPU solvers. The first GPU solver uses a global Degree of Freedom Map (DOFMAP), while the second utilizes an

element-wise strategy to avoid storing the full DOFMAP, reducing memory pressure for large-scale models. Both GPU implementations follow a host-orchestrated, device-dominant execution model, in which DOF mapping, lumped mass computation, and the full leapfrog time-marching loop are executed on the GPU, minimizing CPU-GPU data transfers.

5. Benchmark the parallel implementation against the serial version to quantify the speed-up achieved through parallelization. This comparison will provide insights and identify any potential bottlenecks in the effectiveness of parallel computing in the context of the proposed time-domain image-based Finite Element Method.

By achieving these goals, this research aims to contribute to the advancement of efficient and accurate computational tools for predicting elastic properties in large porous media models, with potential applications in geosciences, materials science, and engineering.

## 1.3 Organization

In what follows, we present in Chapter 2 the literature review about elastic properties in porous media, ultrasonic pulse transmission technique, and digital rock physics. Chapter 3 describes the mathematical formulation used for modeling the ultrasonic pulse transmission in time domain through image-based finite element. Then, in Chapter 4, we show the proposal algorithms and describe the implementations. In Chapter 5, we present the benchmarks, validations, and results for real cases. Finally, in Chapter 6 we present our conclusions and propose future works.

## 2 Literature Review

### 2.1 Elastic Properties of Porous Media

Elastic properties are key parameters used to characterize the mechanical behavior of porous media, including rocks. From a macroscopic perspective, diverse factors influence the elastic effective properties of porous media, including porosity, mineralogical composition, pore geometry, and fluid saturation. Notably, among these factors, porosity and pore geometry are the most critical factors that control the elastic behavior of porous media [26], which is the main cause of variation in effective properties within a specific porosity lies in changes in pore geometry.

When studying the effective properties of random heterogeneous media, like porous materials, researchers have the option to utilize variational approaches and approximations, as discussed in [27]. Additionally, the paper review by Torquato [28] sheds light on significant advancements in determining and calculating improved bounds for the effective properties of random heterogeneous media, with a particular emphasis on elasticity.

Regarding experimental tests for the determination of effective elastic properties of porous media, we have to define two categories based on how they acquire them: static and dynamic. The static computations of effective elastic properties are obtained from experiments focused on sample deformation under loading. In contrast, dynamic elastic properties are derived from measurements of longitudinal waves, shear waves, and sample density. The research conducted by [29] involving 174 laboratory measurements of igneous and metamorphic rocks' static and dynamic elastic properties indicated that dynamic values were generally greater than static values and exhibited a strong linear correlation suitable for regression analysis.

A linear correlation between static and dynamic elastic properties has been reported for certain rock types, notably by B. Belikov and Rysova [30], Nur and Wang [31], and Brotons et al. [32], among others [33, 34, 35, 36]. On the other hand, non-linear relationships have

also been proposed, e.g., by Lacy [37] and Najibi et al. [38], together with [33, 36, 39, 40]. Regardless of the approach adopted, Davarpanah, Ván, and Vásárhelyi [41] attributed the discrepancies between static and dynamic values to grain boundaries and microcracks. Finally, another possible explanation for the discrepancies is the order of strain magnitude. The strain magnitude for static measurements is  $10^{-2}$ , whereas the strain magnitude due to elastic wave propagation is  $10^{-7}$ , as suggested by Mavko, Mukerji, and Dvorkin [2].

A comprehensive review contrasting static and dynamic numerical methodologies for computing effective elastic properties of porous media is provided by Saenger [7]. The author demonstrates that static FEM approaches, through the application of distinct boundary conditions (Dirichlet for upper bounds and Neumann for lower bounds), bracket the effective elastic moduli, while dynamic wave velocity transmission methods yield effective properties representative of specific frequency regimes. Both methodologies are complementary: static approaches are best suited for establishing rigorous bounds, whereas dynamic simulations provide a closer analogue to laboratory ultrasonic measurements. Their numerical discrepancy can be traced to the same strain-magnitude effect observed in experiments, since static computations operate at finite deformation levels while dynamic simulations correspond to infinitesimally small elastic perturbations.

In this thesis, we aim to explore the literature gaps of static and dynamic elastic properties in numerical simulations. Acknowledging porosity and pore geometry, we recognize the necessity for a more detailed examination of specific pore structure characteristics using digital images of rock samples and numerical simulations. Additionally, we explore the examination of factors influencing dynamics analysis, such as the frequency range in measurements.

## 2.2 Ultrasonic Pulse Transmission Technique

The porous medium can be characterized by both its Young's modulus  $E$  and Poisson's ratio  $\nu$ , as well as by P-wave velocity ( $V_P$ ) and S-wave velocity ( $V_S$ ). The relationship between these properties is explained in Appendix A. In geotechnical and civil engineering, various experimental techniques are available to measure the P- and S-waves of porous media in fields or laboratories, such as ultrasonic measurements. For small rock samples, ultrasonic techniques (Figure 2) are highly beneficial as they are non-invasive, adaptable, and proficient in providing reliable measurements of P- and S-wave velocities of porous media.



An initial reference for laboratory and field tests in rock mechanics can be found in [42]. Following that, the American Society for Testing and Materials (ASTM) 1984 [4] and the International Society for Rock Mechanics (ISRM) 1981 [5] established standards for performing UPT testing on rock samples. Over the last decade, ISRM 2015 [43] upgraded the guidelines for predicting dynamic elastic constants based on laboratory test results. Specifically, this updated suggested method refined the experimental procedures for characterizing transverse isotropy in rock cores, standardizing sensor coupling techniques and frequency selection to minimize wave dispersion during ultrasonic pulse transmission.

The UPT, as in Figure 2, consists of measuring the travel time of P- or S-waves through a sample of length  $L$ . This method employs a pair of transducers (transmitter–receiver) to measure the time elapsed between wave emission and arrival. In certain contexts, an oscilloscope can provide a visual representation of the waveform. Finally, the velocity of P- and S-waves can be calculated using the equations  $V_P = L/t_P$  and  $V_S = L/t_S$ , where  $t_P$  and  $t_S$  denote the travel times for P- and S-waves, respectively.

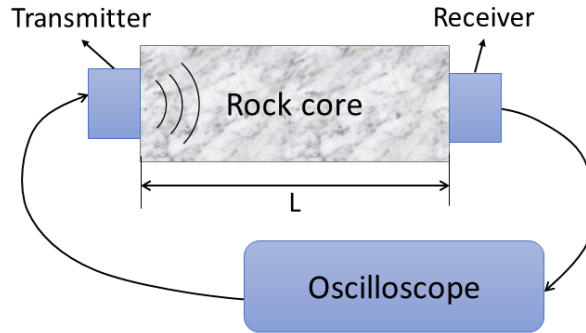


Figure 2: Laboratory setup for ultrasonic pulse transmission showing transmitter and receiver transducers, time measurement display on oscilloscope, and calculation of seismic wave velocities through a rock sample.

Over time, numerous researchers have published articles examining the correlation between wave velocity and elastic properties in a wide range of rock samples. Early studies established this relationship for igneous and sedimentary rocks, notably by Birch [44] and Deere and Miller [45], together with related contributions [46, 47, 48]. This empirical basis was later extended to anisotropic and metamorphic rocks by Barruol and Kern [49] and Kern [50], as well as [51, 52]. More recently, similar correlations have been applied to the engineering classification of building stones, as demonstrated by Kahraman [53] and Yasar and Erdogan [54], alongside [55, 56, 57]. Researchers have also been exploring new ways to use the capabilities of UPT techniques for determining rock properties. Some of these methodologies consist of the utilization of alternative instruments and methodologies that can provide supplementary insights into the characteristics of rocks in different contexts.

In their study, Pimienta et al. [58] conducted experiments using a pressure vessel in dry and fluid-saturated conditions to measure simultaneously ultrasonic wave velocity and electrical resistivity in rocks. On the other hand, researchers such as [59, 60, 61, 62] have made use of methods including micro-scale X-ray computed tomography ( $\mu$ CT) and ultrasonic testing to comprehend the material microstructure better and establish relationships between wave propagation and pore geometry. In this context, a significant application of  $\mu$ CT related to wave propagation lies in the generation of digital representations of rock samples, followed by numerical simulations. The use of high-resolution imaging and numerical simulations is commonly known as Digital Rock Physics and holds a crucial role in fields such as the oil and gas industry, geology, and reservoir engineering.

This thesis intends to showcase that the use of DRP is an important tool for calculating the effective elastic properties of porous media, and that computer simulation allows for a better understanding of material porosity and structure.

## 2.3 Digital Rock Physics and Ultrasonic Pulse Transmission

DRP is a field that integrates digital imaging and computational modeling to analyze and interpret rock properties at a microscopic level. The DRP process comprises various steps, which may differ depending on the study's goals. However, the process includes rock sampling, sample digitization, image segmentation, rock geometry modeling, and physical property simulation for many applications. An example of these steps can be observed in the workflow of Figure 3, which presents the approach to measuring the P- and S-waves of porous media using DRP and experimental tests. It is important to emphasize that, despite the experimental determination being described in only one step, the experimental process presents various challenges such as signal attenuation, signal scattering, coupling issues, surface roughness, temperature sensitivity, frequency selection, calibration issues, interference, among others.

During the 1980s, there was a growing interest in sample digitization using  $\mu$ CT for studying different materials. In particular, Flannery et al. [6] conducted pioneering work on the microtomography technique in sandstone at 8 keV to distinguish slight differences in density and to selectively map the spatial distribution. Over the years, scientists and researchers have made significant progress in developing advanced techniques for digitizing samples [63, 64, 65] and simulating porous media [66, 67, 14]. These achievements played a critical role in enhancing our comprehension of the microstructure of porous media and

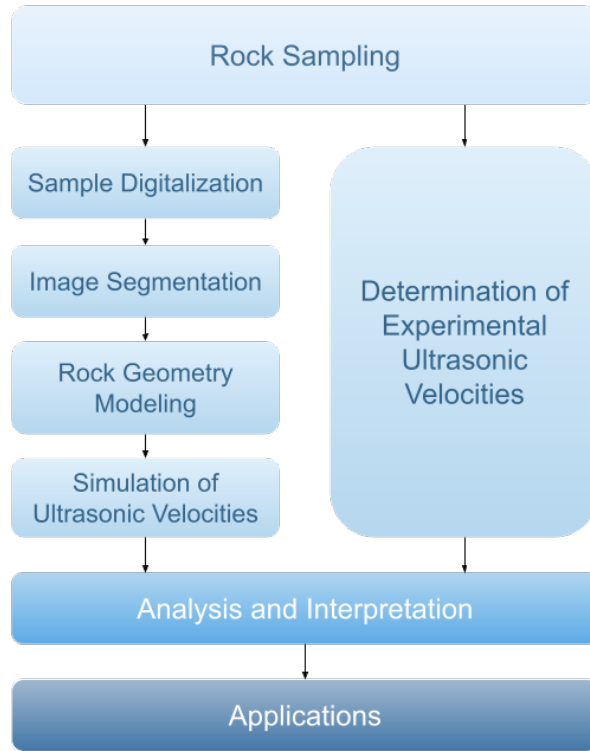


Figure 3: Workflow showing the integration of digital rock physics and laboratory testing: from sample digitization via  $\mu$ CT imaging, image processing and segmentation, numerical simulation of ultrasonic wave propagation, to experimental velocity determination.

its impact on the comprehension of elastic properties.

The evolution of knowledge in DRP and UPT can be traced back to the pioneering work of researchers such as Saenger and Bohlen [68]. Their study simulated wave propagation in an anisotropic and viscoelastic medium using the FDM. The work enabled the modeling of complex media with high-contrast discontinuities. In [9], the effective elastic properties of a Fontainebleau sandstone were calculated using wave propagation and the results were compared with a static finite-element approach. The DRP approach used to compute the effective elastic properties of porous media is similar to the experimental strategy depicted in Figure 2 and was explained in [10, 8, 11, 12, 13]. In the numerical model, the solid regions are assigned the elastic properties of the grain material, while free boundary conditions are applied to the porous regions. The simulation is performed by propagating a P- or S-wave parallel to the transducer pair. Regarding the pulse shape, Andrä et al. [14] used a Gaussian pulse and Periodic Boundary Conditions (PBCs), ensuring a smooth and continuous signal. These same waveforms and boundary conditions will also be implemented in the present work.

Studies such as Andrä et al. [14] were conducted to investigate the effectiveness of

DRP in wave propagation. These studies compared experimental results in sandstone with numerical outcomes. One of the most significant findings of these studies was that DRP produced rock-physics patterns that were highly consistent with laboratory observations on a large scale. This suggests that DRP is a promising tool for accurately predicting the effective elastic properties of porous media. On the other hand, the study was limited in the dynamic elasticity analysis to images of  $400^3$  voxels. Larger images were reduced through resampling to a coarsened sample due to limitation of computational resources. It has been shown by Faisal et al. [22] that numerical and experimental results can also be compared in the case of carbonates. However, there were limitations in the size of wave propagation models to  $648^3$  voxels even using High-Performance Computing (HPC). In Saenger, Krüger, and Shapiro [8], the large-scale computer CRAY T3E with MPI implementation was employed primarily due to RAM and CPU usage constraints. The maximum model analyzed in their study is limited to  $400^3$  voxels. This means that researchers and engineers had to develop simulation strategies using parallel computing techniques [13]. Nevertheless, there is still a significant need for improved algorithms that can generate more efficient simulations by handling larger models in reduced time.

In this study, we employed several strategies to determine the effective elastic properties of digital porous media by utilizing the UPT technique with the FEM. The first distinctive approach in this work is the modeling of only the solid phase using the image-based method. This approach, successfully employed in Vianna et al. [16], allows for greater model control and reduced memory consumption, making it feasible for large-scale simulations. Moreover, this type of problem is highly efficient and can be applied in both serial and parallel computing environments. Additionally, the implementation of this work will make use of matrix-free techniques such as EbE and Node-by-Node (NbN) methods [17, 19]. The EbE and NbN methods change the process of storing a large sparse matrix, by the process of computing a matrix-vector product at each step of the solution using an element map and a node map, respectively. These techniques allow further reduction for both memory usage reduction in serial and parallel environments and non-concurrency in parallel settings. The results of the present study are valuable for several applications, including oil and gas exploration, geothermal energy, and civil engineering.

## 3 Mathematical Modeling

The investigation of P- and S-waves, also known as primary and secondary waves, facilitates our comprehension of material properties in porous media. P-waves transmit variations in pressure, while S-waves trigger displacements perpendicular to the direction of propagation. By analyzing differences in P- and S-wave velocities using the FEM,  $\mu$ CT images, and UPT, it is possible to uncover information about porosity, fluid saturation, and spatial distribution [2]. This powerful combination of methods is significant for applications such as aquifers, oil fields, sedimentary rocks, and several other geological and industrial scenarios [31, 2].

### 3.1 Theory of Wave Propagation in Elastic Media

In this section, we will explore the underlying principles of wave propagation in elastic media, specifically P- and S-waves. These waves result from mechanical disturbances and govern these complex dynamic phenomena. The underlying mathematical and physical principles is presented as a foundation for further analysis. Understanding the characteristics and behavior of P- and S-waves is essential for investigations in porous media.

The P-wave is similar to how acoustic waves travel through fluids. The wave moves through the material's interior with a phase velocity represented by  $V_P$  (Figure 4) and is characterized by particle motion that results in pure dilatation. This means that the particles in the rock are compressed and expanded in the direction of the wave's propagation. Because of this motion, the P-wave is the fastest seismic wave and the first to arrive at an observation point. On the other hand, the S-wave involves shear motions of particles perpendicular to their direction of propagation, as shown in Figure 5. Similar to the P-wave, the S-wave travels through space, but its phase velocity  $V_S$  is generally lower than  $V_P$ . The S-wave can be further divided into two components: the SV-wave and the SH-wave. The H and V represent horizontal and vertical directions, respectively, referring to the particle motion's orientation in Cartesian coordinates [2]. It is worth mentioning

that when encountering a free surface or an interface between two materials, a partial transition between P-waves and S-waves, and vice versa, may occur.

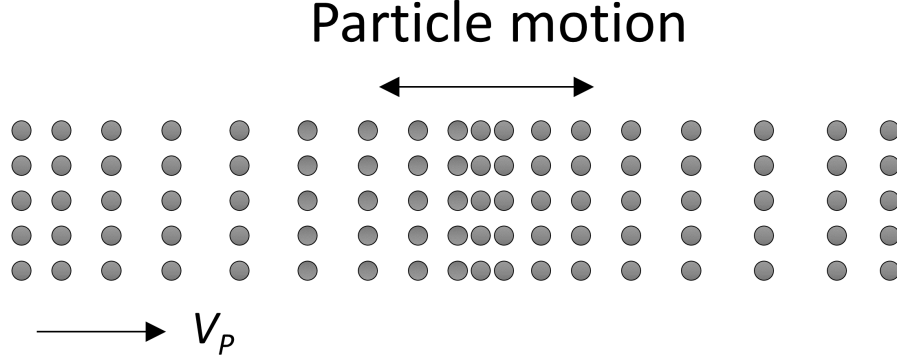


Figure 4: Plane P-wave with particle motion parallel to the direction of propagation. The wave causes compressional deformation as it travels through the medium.

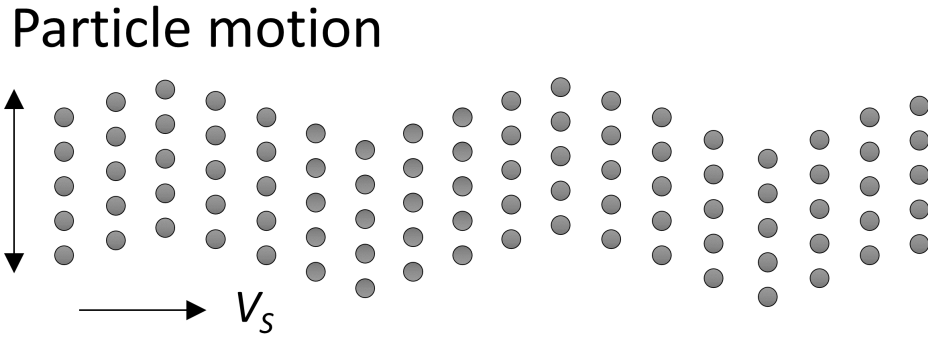


Figure 5: Plane S-wave with particle motion perpendicular to the direction of propagation. The transverse motion causes shear deformation in the medium, typically arriving after P-waves with lower phase velocity.

One of the mathematical models used to describe the behavior of a continuous medium under the influence of external forces is the Cauchy equation of motion. The equation is based on the principles of conservation of linear momentum and angular momentum to establish a relationship between the forces acting on a body and the resulting motion of that body [69, 70]. In an infinite elastic domain  $\Omega$  the Cauchy equation of motion can be expressed as follows:

$$\sigma_{ij,j} + \rho b_i = \rho \ddot{u}_i, \quad (3.1)$$

where  $\sigma_{ij}$  is the Cauchy stress tensor,  $\rho$  is the mass density,  $b_i$  expressing the body forces per unit mass,  $u_i$  are the components of the displacement field, and  $\ddot{u}_i$  are the second

partial derivative of displacement field to time, indicating the acceleration of the material points. Notably, no assumptions regarding material behavior have been made up to this point.

Assuming that the material is considered linear elastic with small deformations, the generalized Hooke's law can be used to express the stress in terms of the strains:

$$\sigma_{ij} = C_{ijkl}\epsilon_{kl} , \quad (3.2)$$

where  $C_{ijkl}$  is the stiffness tensor, and  $\epsilon_{kl}$  is the infinitesimal strain tensor by

$$\epsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) . \quad (3.3)$$

Further, using the constitutive Equation 3.2, the compatibility Equation 3.3, the natural symmetry of the Cauchy stress tensor, and the symmetry of the strain tensor we can rewrite the Cauchy stress tensor as

$$\sigma_{ij} = \lambda\delta_{ij}\epsilon_{kk} + 2\mu\epsilon_{ij} , \quad (3.4)$$

where  $\lambda$  and  $\mu$  are Lamé's first and second elasticity constants, and  $\delta_{ij}$  is the Kronecker delta.

For the linear elastic homogeneous isotropic medium, and assuming zero body forces ( $b_i = 0$ ) as is typical for high-frequency wave propagation, the Cauchy equation of motion reduces to the Navier equations,

$$(\mu + \lambda)u_{j,ij} + \mu u_{i,jj} = \rho\ddot{u}_i , \quad (3.5)$$

where, for isotropic linear elasticity,

$$\lambda = \frac{\nu E}{(1 + \nu)(1 - 2\nu)} , \quad \mu = \frac{E}{2(1 + \nu)} , \quad (3.6)$$

where  $E$  is the Young's modulus and  $\nu$  is the Poisson's ratio. As described, the Navier equations are essential for performing numerical simulations in this work and will be further explored in the next sections.

It is also possible to define the P- and S-wave velocities by using the Equation 3.5 for

a continuous elastic medium by taking the divergence and rotational, rearranging, and substituting terms that involve repeated indices [2] reducing the equation to

$$u_{i,ijj} = \frac{1}{V_P^2} \ddot{u}_{i,i}, \quad V_P = \sqrt{\frac{\lambda + 2\mu}{\rho}}, \quad (3.7)$$

where  $u_i$  denotes the components of the displacement field,

$$\omega_{i,jj} = \frac{1}{V_S^2} \ddot{\omega}_i, \quad V_S = \sqrt{\frac{\mu}{\rho}}, \quad (3.8)$$

$$\omega_i = \frac{1}{2} \varepsilon_{ijk} u_{k,j}, \quad (3.9)$$

where  $\omega_i$  is the rotation of the displacement field and  $\varepsilon_{ijk}$  is the permutation symbol. However, for highly heterogeneous porous media derived from  $\mu$ CT imaging, simple analytical formulas are insufficient to capture the true dynamic response. While effective medium theories exist, the complex microstructural geometry makes it challenging to accurately derive wave velocities mathematically without idealized assumptions. Instead, this work relies on the Navier equation (Equation 3.5) to perform numerical simulations of UPT on the exact micro-geometry, thereby estimating these effective wave velocities accurately.

## 3.2 Finite Element Method

The FEM applied in this work is highly suitable for performing elastodynamic simulations in the time domain. It is well known that the flexibility of the FEM makes it a popular choice for modeling complex geometries in porous media under dynamic conditions [71]. To handle the massive scale of effective properties problems, where meshes frequently reach millions of degrees of freedom (DOF), the traditional FEM approach would require a prohibitive amount of computational memory to store the data. Thus, using the strategies introduced by advanced image-based methods, we implement highly efficient solvers focused on the elimination of global matrix assembly (assembly-free), which enables reduction in memory usage and facilitates parallelization on GPU architectures.



### 3.2.1 Formulation

The foundation of our numerical approach relies on the transformation of the continuous strong form of the governing partial differential equations into an equivalent weak formulation. The classic finite element literature provides a thorough explanation of these basic variational procedures [72, 70]. Specifically, the weak form of the Navier equation (Equation 3.5) is obtained by multiplying the equation by a test (weighting) function, denoted  $\delta u_i$ , and integrating over the domain  $\Omega$ , yielding the integral weak formulation, expressed as

$$\int_{\Omega} \delta u_i ((\mu + \lambda) u_{j,ij} + \mu u_{i,jj} - \rho \ddot{u}_i) d\Omega = 0 . \quad (3.10)$$

In FEM, the domain is divided into smaller parts called elements. Piece-wise polynomial functions are used to approximate the displacement inside each element and, following the Galerkin method adopted in this work, the same functions are used as the weighting functions  $\delta u_i$ . In the present work, tri-linear shape functions are adopted, consistent with the eight-node hexahedral (voxel) element formulation detailed in Section 3.4. By incorporating boundary conditions and performing numerical integration over each element, a system of linear algebraic equations for nodal displacements can be derived. This results in the final FEM form of the equation of motion for the entire domain,

$$\mathbf{K}\mathbf{u} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{F} \quad (3.11)$$

or

$$\ddot{\mathbf{u}} = \mathbf{M}^{-1}(\mathbf{F} - \mathbf{K}\mathbf{u}) , \quad (3.12)$$

where  $\mathbf{u}$  and  $\ddot{\mathbf{u}}$  are the global displacement and acceleration vectors, respectively. The global stiffness and mass matrices are  $\mathbf{K}$  and  $\mathbf{M}$ , while the global nodal forces vector is  $\mathbf{F}$ . These are defined as

$$\mathbf{K} = \sum_{e=1}^{N_{elem}} \int_{\Omega_e} \mathbf{B}^T \mathbf{C} \mathbf{B} d\Omega , \quad (3.13)$$

$$\mathbf{M} = \sum_{e=1}^{N_{elem}} \int_{\Omega_e} \rho \mathbf{N}^T \mathbf{N} d\Omega , \quad (3.14)$$

$$\mathbf{F} = \sum_{e=1}^{N_{elem}} \int_{\Gamma_e} \mathbf{N}^T \mathbf{t} d\Gamma, \quad (3.15)$$

$$\mathbf{C} = \begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix}. \quad (3.16)$$

In the above equations, the summation includes the assembly of  $N_{elem}$  elements in the domain. The matrix  $\mathbf{N}$  is the tri-linear shape functions,  $\mathbf{B}$  is the strain-displacement matrix containing the spatial derivatives of  $\mathbf{N}$ ,  $\Omega_e$  and  $\Gamma_e$  are the volume and boundary of element  $e$ ,  $\mathbf{t}$  is the boundary traction vector, and  $\mathbf{C}$  is the elasticity matrix for a homogeneous isotropic material in 3D.

### 3.2.2 Assembly-Free and Stiffness-Displacement Product

The explicit assembly and storage of the global stiffness matrix  $\mathbf{K}$  represent limiting factors that frequently render the simulation of large-scale RVEs unfeasible, quickly exhausting the available computational memory [70]. The matrix  $\mathbf{K}$  is traditionally obtained by assembling the contributions of the local stiffness matrix  $\mathbf{k}_e$  of each element  $e$ , such that:

$$\mathbf{K} = \sum_{e=1}^{N_{elem}} \mathbf{A}_e^T \mathbf{k}_e \mathbf{A}_e, \quad (3.17)$$

with

$$\mathbf{k}_e = \int_{\Omega_e} \mathbf{B}_e^T \mathbf{C}_e \mathbf{B}_e d\Omega, \quad (3.18)$$

where  $\mathbf{A}_e$  is the Boolean connectivity matrix that maps the global degrees of freedom to the local ones,  $\mathbf{B}_e$  is the element strain-displacement matrix, and  $\mathbf{C}_e$  is the element elasticity matrix.

To mitigate this memory bottleneck, the present implementation adopts a strategy based on the EbE technique, a consolidated paradigm for bypassing storage constraints in large-scale FEM computations, as seen in Vianna et al. [16] and Lopes et al. [73,

19]. In this context, the present thesis utilizes an assembly-free strategy to compute the stiffness-displacement product  $\mathbf{v} = \mathbf{K}\mathbf{u}$  defined in Equation 3.11. Leveraging the algebraic properties of the assembly operator, the global product is rewritten and evaluated strictly as the superposition of local products:

$$\mathbf{v} = \sum_{e=1}^{N_{elem}} \mathbf{A}_e^T (\mathbf{k}_e \mathbf{u}_e) , \quad (3.19)$$

where  $\mathbf{u}_e = \mathbf{A}_e \mathbf{u}$  represents the gather operation of the local displacement vector from the global vector. In Section 3.4, further considerations regarding the optimization of this mapping procedure will be addressed.

Instead of allocating the global data structure for  $\mathbf{K}$ , the algorithm iterates directly over each finite element of the domain. During this execution, the local internal force vector  $\mathbf{v}_e = \mathbf{k}_e \mathbf{u}_e$  is evaluated and mapped to the corresponding global degrees of freedom *on-the-fly* through the scatter operation represented by  $\mathbf{A}_e^T$ . These internal forces are accumulated directly into the global vector  $\mathbf{v}$  (programmatically represented as  $\mathbf{K}\mathbf{u}$ ). This approach overcomes the massive memory allocation inherent to structuring global sparse matrices.

Consequently, the localized nature of the matrix-vector computation provides an ideal mathematical framework for massively parallel execution using CUDA. Addressing the limitations of existing implementations, this work introduces a novel, purpose-built CUDA kernel dedicated to the assembly-free evaluation of the stiffness-displacement product in dynamics problems. Through this formulation, the local matrix-vector multiplications are performed completely *on-the-fly*, and data dependencies during the scatter operation are resolved using atomic additions. By using the parallelism of the GPU, this approach speeds up the solver's execution, while also avoiding the memory problems that are common in traditional storage methods.

### 3.2.3 Lumped Mass Matrix and Inversion

In addition, to ensure the computational efficiency necessary to simulate large-scale models, this work uses lumped mass matrices [72] as opposed to the consistent mass matrices from Equation 3.14. The lumped mass matrices decouple the equations of motion by yielding a diagonal system, making them particularly well-suited for reducing memory usage and computational cost in FEM models. Specifically, the lumped mass matrix can be written

as

$$\mathbf{M}_{\text{lumped}} = \sum_{e=1}^{N_{\text{elem}}} \delta_{ij} \int_{\Omega_e} \rho \mathbf{N} d\Omega . \quad (3.20)$$

Applying the integration defined in Equation 3.20 to a tri-linear hexahedral finite element (voxel) you obtain an exact analytical solution. Assuming the density of the element  $\rho_e$  is uniform across the element and  $V_e$  is its volume, the mass of the element is distributed evenly among its eight nodes. To align with the previously established assembly-free EbE strategy, these local diagonal contributions are linked to the global DOF through the Boolean connectivity matrices  $\mathbf{A}_e$ . It follows, then, that the global lumped mass matrix can be written, using the identity matrix  $\mathbf{I}$ , as

$$\mathbf{M}_{\text{lumped}} = \sum_{e=1}^{N_{\text{elem}}} \mathbf{A}_e^T \left( \frac{\rho_e V_e}{8} \mathbf{I} \right) \mathbf{A}_e . \quad (3.21)$$

It is worth noting that while lumped mass matrices may sacrifice some accuracy compared to consistent mass matrices, they offer practical advantages. Although both formulations exhibit  $\mathcal{O}(n)$  storage complexity, where  $n$  is the total number of degrees of freedom. The lumped mass matrix requires less memory because only its diagonal entries are stored. More importantly, its inversion reduces to a simple element-wise operation, avoiding the solution of a coupled linear system.

This trade-off between accuracy and efficiency depends on frequency. Dispersion analyses of the semi-discretized wave equation show that the two mass formulations bias the numerical phase velocity in opposite directions: the consistent mass matrix overestimates the frequency of the higher, while the lumped mass matrix underestimates them by a similar amount [74, 75]. Marfurt [76] quantified this effect for scalar and elastic wave propagation, showing that mass lumping affects accuracy mainly in the short-wavelength range, i.e., when few grid points are available per wavelength, and that this effect is reduced once the spatial dispersion criterion of Section 3.3.2 is met. Since the source spectrum adopted in this work satisfies  $N_{\text{min}} \geq 30$ , the additional dispersion error introduced by mass lumping is expected to remain within the accuracy already set by the spatial discretization, in exchange for the memory and computational savings described above.

The solver implemented reflects the logic presented in Algorithm 1: the mass of each voxel is computed by the product of the material density and the element volume.

Subsequently, the mass fractions are assigned to the mapped DOFs of the element using the parallelism of the GPU. Finally, the matrix inversion is resolved by computing the reciprocal of each non-zero diagonal element, maximizing computational throughput through GPU parallelization.

---

**Algorithm 1** Lumped Mass Assembly
 

---

**Input:** DOFMAP, image, materialInfo,  $\Delta L$ 
**Output:**  $\mathbf{M}^{-1}$ 

```

1:  $V \leftarrow \Delta L^3$  ▷ Compute voxel volume
2:  $w \leftarrow \frac{1}{8}$  ▷ Compute node weight
3: for  $k \leftarrow 0$  to  $n_{materials} - 1$  do ▷ Compute mass for each material
4:    $\text{voxelMass}[k] \leftarrow \text{materialInfo}[k] \times V$ 
5: end for
6: for  $e \leftarrow 0$  to  $n_{elem} - 1$  do ▷ Compute lumped mass matrix
7:   if  $\text{image}[e] \neq \text{poro}$  then
8:      $m_e \leftarrow \text{voxelMass}[\text{image}[e]] \times w$ 
9:     for  $i \leftarrow 0$  to 7 do ▷ Iterate over 8 nodes
10:       $d \leftarrow \text{DOFMAP}[e \times 8 + i]$ 
11:       $\mathbf{M}[d] \leftarrow \mathbf{M}[d] + m_e$ 
12:    end for
13:  end if
14: end for
15: for  $i \leftarrow 0$  to  $n_{nodes} - 1$  do ▷ Invert non-zero entries of  $\mathbf{M}$ 
16:   if  $\mathbf{M}[i] \neq 0$  then
17:      $\mathbf{M}^{-1}[i] \leftarrow \frac{1.0}{\mathbf{M}[i]}$ 
18:   end if
19: end for

```

---

### 3.3 Stability and Accuracy of the Numerical Scheme

To ensure the reliability of the elastodynamic simulations, the numerical scheme must satisfy two fundamental criteria: temporal stability and spatial accuracy. While the stability condition prevents the numerical solution from diverging over time, the accuracy criterion ensures that the spatial discretization is fine enough to represent the physical phenomena without significant distortion. The following subsections detail the Leap-Frog method's stability limits and the requirements for suppressing numerical dispersion.

#### 3.3.1 Explicit Method and the Critical Time Step

The Leap-Frog explicit method, closely related to the Verlet scheme [77], is a numerical integration technique used to solve ordinary differential equations (ODEs). It is widely

employed in various fields, including structural mechanics and earthquake engineering, to describe the motion response of structures subjected to dynamic loads. The Leap-Frog scheme works by dividing time into discrete steps and estimating the velocity and displacement at each time step. It uses a straightforward midpoint rule to estimate the velocity and displacement at half-step intervals, making it computationally efficient as compared to an implicit Newmark method. At each increment, the stable algorithm does not require complex iterations to update the velocity and displacement using force-induced acceleration. The equations for velocity and displacement, respectively, can be written as

$$\dot{\mathbf{u}}(t + \Delta t/2) = \dot{\mathbf{u}}(t - \Delta t/2) + \ddot{\mathbf{u}}(t)\Delta t, \quad (3.22)$$

$$\mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \dot{\mathbf{u}}(t + \Delta t/2)\Delta t, \quad (3.23)$$

where  $\ddot{\mathbf{u}}(t)$  is the acceleration at time  $t$ ,  $\dot{\mathbf{u}}(t)$  is the velocity,  $\mathbf{u}(t)$  is the displacement, and  $\Delta t$  is the time step size.

The Newmark method is a more accurate numerical integration technique than the Leap-Frog scheme for problems with strong nonlinearities or large excitations. On the other hand, the explicit method is particularly well-suited to linear elastodynamics models due to the Leap-Frog scheme being a symplectic integration method. This implies that it can conserve the energy of the system over extended periods.

Furthermore, it is necessary to find a certain critical time step for the stability of the Leap-Frog scheme. According to O'Sullivan and Bray [78], the critical time step in a linear undamped system is the relationship given by:

$$\Delta t \leq \Delta t_{cr} = \frac{2}{\omega_{max}}, \quad (3.24)$$

where  $\Delta t_{cr}$  is the critical time step and  $\omega_{max}$  is the highest frequency in the smallest element. This condition ensures that the numerical solution does not propagate too quickly, avoiding instabilities and providing a stable and accurate numerical solution.

In wave propagation problem [69], for a regular mesh composed of 2D square elements and assuming small deformations of elements is possible to use the CFL condition [79], also known as the Courant–Friedrichs–Lewy condition, to compute the critical time step:

$$\Delta t \leq \Delta t_{cr} = \frac{\Delta L}{V_P}, \quad (3.25)$$

$$CFL \equiv \frac{V_P \Delta t}{\Delta L} \leq 1, \quad (3.26)$$

where  $\Delta L$  is the square element size and  $V_P$  is the velocity of the propagation wave in the media.

When analyzing a model with a regular mesh and multiple materials in three dimensions, since all elements share the same size  $\Delta L$ , the critical time step depends solely on the material-specific wave velocity  $V_P$  (Equation 3.26). Therefore, it is more efficient to evaluate  $\Delta t_{cr}$  once for each material present in the model instead of once for each element of the mesh, and the global time step adopted in the simulation is set as the minimum among these material-specific critical values.

### 3.3.2 Spatial Dispersion Criterion

While the CFL condition (Equation 3.26) ensures temporal stability by preventing information from traveling more than one element per time step, it does not guarantee the faithful representation of the wave's spatial profile. An insufficient number of grid points per wavelength causes numerical dispersion: the simulated wave travels at an artificially altered phase velocity, introducing systematic errors in the effective-velocity estimates extracted from the sensor signals.

Based on the accuracy assessment conducted by Bohlen and Saenger [80] for staggered-grid finite-difference methods, which can be analogously applied to voxel-based FEM, the spatial dispersion condition sets a limit on the maximum size of the elements:

$$\Delta L \leq \frac{\lambda_{\min}}{N_{\min}} = \frac{V_{S,\min}}{f_{\max} N_{\min}}, \quad (3.27)$$

where  $\lambda_{\min} = V_{S,\min}/f_{\max}$  is the minimum wavelength, governed by the slowest wave speed  $V_{S,\min}$  and the maximum frequency component  $f_{\max}$  of the source pulse (see Appendix C), and  $N_{\min}$  is the minimum number of grid points per wavelength required for acceptable accuracy. For planar free surfaces aligned with the grid,  $N_{\min} \approx 15\text{--}30$  is typically sufficient [80]. For irregular pore-grain interfaces of the type present in  $\mu$ CT-derived rock images, adopting  $N_{\min} \geq 30$  is recommended to reliably suppress dispersion artifacts.

In this work, the source peak frequency  $f_0$  and the time step  $\Delta t$  are chosen such that both the CFL stability criterion (Equation 3.26) and the spatial dispersion criterion (Equation 3.27) are simultaneously satisfied, ensuring that the computed effective velocities are free from significant dispersion-induced bias.

### 3.4 Image-Based Finite Element Modeling

FEM is a numerical technique composed of two key processes: modeling and simulation. These processes are divided into multiple steps, including problem definition, discretization, formulation of governing equations, assembly, solution, and post-processing. It is important to recognize that any alterations made within these steps can have an impact on the expected results of FEM. In order to effectively model and simulate the UPT using FEM, it is imperative to consider certain critical observations and adjustments.

In this work, similar to the works of Vianna et al. [16] and Lopes et al. [19], only one phase of the porous medium (solid or pores) will be modeled using FEM. As the interest of this work is to perform the UPT, only the solid part (heterogeneous medium composed of multiple materials) is modeled in dry rock condition. This choice helps reduce the amount of stored data and computation costs in different parts of the algorithm. The interface between the solid part and the pores will be treated as free surfaces with Neumann boundary conditions. Moreover, each solid element will be considered homogeneous and isotropic. Regarding the domain discretization, we will adopt the image-based strategy [16], wherein each pixel or voxel in the  $\mu$ CT image corresponds to a respective FEM element, quadrilateral element in 2D and a hexahedral element in 3D. Since the mapping of the finite element with the image is straightforward, there is no need for the use of any mesh generation algorithms or the storage of node coordinate values. As all elements have the same geometry, it is also possible to analytically precompute and store (see Appendix B) only one stiffness and mass matrix per material. Then again, reducing storage and setup costs compared to structured mesh approaches. For a regular hexahedral element (voxel) with side length  $\Delta L$  and three displacement degrees of freedom per node, the local stiffness matrix  $\mathbf{k}_e$  is a  $24 \times 24$  symmetric matrix. Its entries depend solely on the Lamé parameters  $\lambda$  and  $\mu$  of the material (Equation 3.6) and on the element volume  $V_e = \Delta L^3$ , and are obtained by analytical integration of the product  $\mathbf{B}^T \mathbf{C} \mathbf{B}$  over the element domain, as detailed in Appendix B.



### 3.4.1 Element and Node Mapping

The process of numbering elements and nodes after reading a  $\mu$ CT image involves following a specific order. This order is determined by a top-to-bottom, left-to-right approach as seen in Figure 6 for a 2D image. This means that the numbering starts at the first element from the top left corner and proceeds to the bottom right corner of the image at the last solid element. By following this systematic approach, each node and element is accurately labeled with a unique number, making it easier to access and compute the contributions of each element.

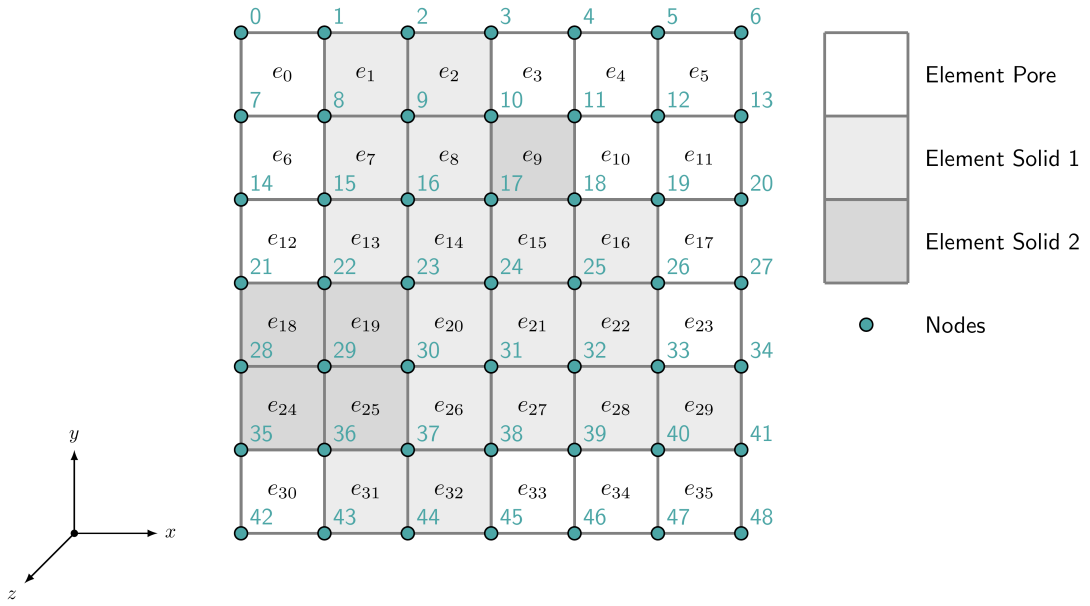


Figure 6: Example of element and node mapping in image-based porous media.

The three-dimensional image domain map for each element at grid position  $(e_x, e_y, e_z)$  is assigned a linear index

$$\begin{aligned}
 e &= e_x + n_x e_y + n_x n_y e_z \\
 e_x &= e \bmod n_x \\
 e_y &= \left\lfloor \frac{e \bmod (n_x n_y)}{n_x} \right\rfloor \\
 e_z &= \left\lfloor \frac{e}{n_x n_y} \right\rfloor,
 \end{aligned} \tag{3.28}$$

where  $e$  is the element index,  $n_x, n_y$  and  $n_z$  are the domain dimensions in the  $x$ -,  $y$ - and  $z$ -directions. The operator  $a \bmod b$  denotes the remainder of the integer division of  $a$  by  $b$ , while  $\lfloor \cdot \rfloor$  denotes the floor function, which returns the greatest integer less than or equal to its argument.

For the *node*, the equation is given by:

$$\begin{aligned}
 node &= i_x + (n_x + 1) i_y + (n_x + 1)(n_y + 1) i_z \\
 i_x &= e \bmod (n_x + 1) \\
 i_y &= \left\lfloor \frac{e \bmod ((n_x + 1)(n_y + 1))}{(n_x + 1)} \right\rfloor \\
 i_z &= \left\lfloor \frac{e}{(n_x + 1)(n_y + 1)} \right\rfloor .
 \end{aligned} \tag{3.29}$$

where  $(i_x, i_y, i_z)$  are the discrete grid coordinates corresponding to the global node index *node*.

Equations 3.28 and 3.29 enable an individualized mapping of both elements and nodes. This feature is fundamental to the parallel computing strategies employed in this thesis, as it ensures that each node or element possesses the necessary local information to perform FEM independently. This data self-sufficiency forms the foundation for the various solvers implemented in this work. Regarding node mapping, this thesis presents two distinct approaches that dictate the total number of DOF within the models: *Direct Node Mapping* and *Active Node Mapping*. Considerations regarding the amount of memory saved with the two approaches will be reported in Section 4.2.

### 3.4.2 Direct Node Mapping

The implementation of the Direct Node Mapping (DNM) represents the most conservative strategy in terms of computational feasibility for image-based simulations. By using the algebraic relations defined in Equation 3.28, the global connectivity can be determined algorithmically *on-the-fly* solely from the linear index of the element. This approach not only eliminates the need to store an explicit connectivity vector, but also improves performance on parallel computing architectures (such as GPUs), where the primary bottleneck is often memory access rather than arithmetic operations. However, for complex porous geometries, nodes are unnecessarily introduced within pore regions, leading to an increase in the number of DOFs and, consequently, in the overall memory requirements of the model.

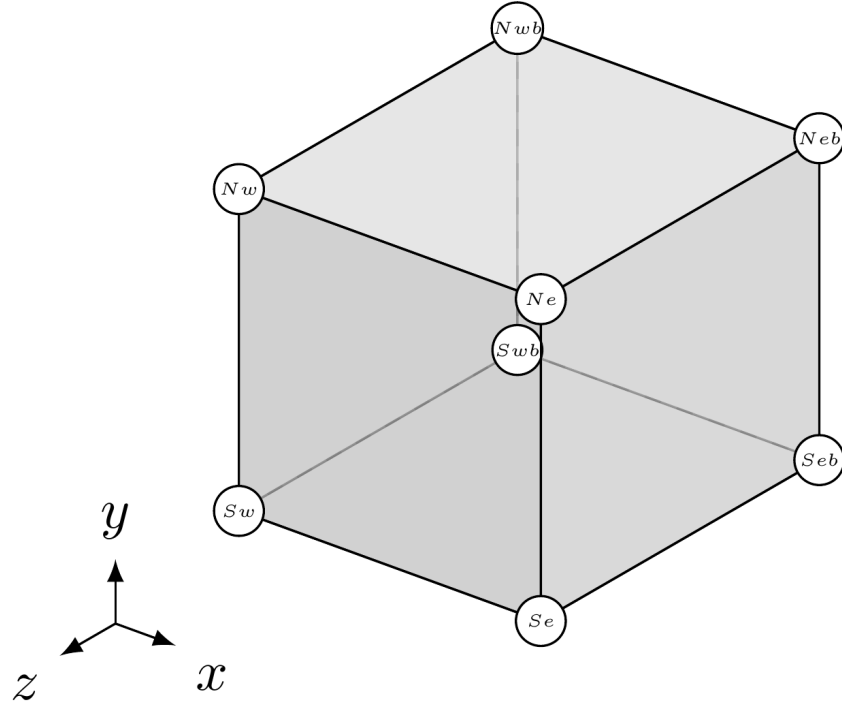


Figure 7: Schematic representation of the local numbering of nodes in a 3D voxel element.

Based on the grid position  $(e_x, e_y, e_z)$  of an element  $e$ , the eight nodes that define the element are identified according to the topology shown in Figure 7. The mapping follows the rule defined by the system of equations

$$\begin{aligned}
 Nw &= e + e_y + e_z(n_x + n_y + 1) \\
 Sw &= Nw + n_x + 1 \\
 Se &= Nw + n_x + 2 \\
 Ne &= Nw + 1 \\
 Nwb &= Nw + (n_x + 1)(n_y + 1) \\
 Swb &= Sw + (n_x + 1)(n_y + 1) \\
 Seb &= Se + (n_x + 1)(n_y + 1) \\
 Neb &= Ne + (n_x + 1)(n_y + 1) .
 \end{aligned} \tag{3.30}$$

The element nodes are named according to their relative position in the local coordinate system, where Nw, Ne, Sw, and Se denote the North-West, North-East, South-West, and South-East nodes on the front face, respectively, while the suffix b (back) indicates the corresponding nodes located on the back face of the element along the z-direction,

$$n_{\text{nodes}} = (n_x + 1)(n_y + 1)(n_z + 1)$$

$$\text{DOF}_{\text{total}} = n_{\text{nodes}} \times \text{DOF}_{\text{node}}, \quad \text{where} \quad \text{DOF}_{\text{node}} = \begin{cases} 2, & \text{for 2D images} \\ 3, & \text{for 3D images.} \end{cases} \quad (3.31)$$

### 3.4.3 Active Node Mapping

The Active Node Mapping (ANM) algorithm plays a fundamental role in the preprocessing phase in the simulations presented in this thesis. In contrast to DNM, in which all nodes are considered, this algorithm employs a selective strategy based on the spatial distribution of material phases (solid vs. pore). This approach is essential for reducing the total number of DOF in porous media models. The strategy consists in evaluating nodal connectivity and label the nodes as: active or inactive. A node is classified as active if it is associated with at least one solid element. Conversely, inactive nodes are those exclusively adjacent to pore elements. This methodology aligns with the approach proposed by Vianna et al. [16]. The final output of this algorithm is a mapping vector containing only the nodes required for the FEM assembly, as illustrated in Figure 8. Currently, this is an iterative serial process shown in the Figure 9 and has not been parallelized.

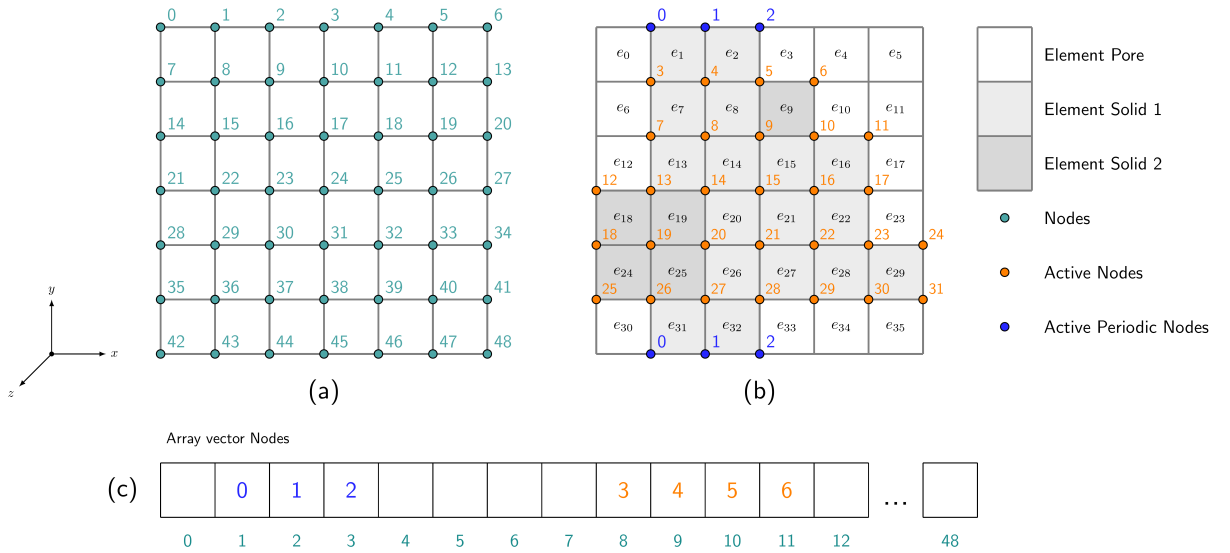


Figure 8: Example of active node mapping in image-based porous media: (a) direct node mapping; (b) active node mapping; and (c) vector with active nodes mapped.

The procedure operates iteratively over the complete set of nodes, denoted by  $n_{\text{nodes}}$ , and performs three critical steps for each node: (i) reconstruction of topological coordinates,

(ii) material classification based on neighborhood information, and (iii) application of periodic boundary conditions.

**(i) Topological Reconstruction and Neighborhood Identification:** Iterating through all the nodes of the model, the algorithm first decomposes using Equation 3.29 the global node index (*node*) into discrete coordinates ( $i_x, i_y, i_z$ ) corresponding to the  $x$ ,  $y$ , and  $z$  directions. This decomposition enables the identification of adjacent elements that share the node under consideration. The algorithm computes the linear indices of up to eight neighboring elements (NE, NW, SE, and SW quadrants, both on the front and back slices), while properly handling domain boundary cases.

**(ii) Material Classification (Solid vs. Pore):** The decision to include a node in the active mapping is based on the physical properties of its adjacent elements. A logical inclusion criterion is employed: a node is classified as active (belonging to the solid phase) if, and only if, at least one of its neighboring elements is identified as solid material. Nodes surrounded exclusively by pore voxels are marked as inactive, thereby reducing the dimensionality of the global system.

**(iii) Periodicity and Indexing Treatment:** To ensure physical continuity in Representative Volume Element (RVE) simulations, the algorithm enforces Periodic Boundary Conditions (Section 3.4.4).

At the end of the procedure, the total number of DOF in the system is defined as the product of the number of unique solid nodes ( $n_{\text{solid}}$ ) and the number of degrees of freedom per node ( $\text{DOF}_{\text{node}}$ ),

$$n_{\text{solid}} \leq n_{\text{nodes}}$$

$$\text{DOF}_{\text{solid}} = n_{\text{solid}} \times \text{DOF}_{\text{node}}, \quad \text{where} \quad \text{DOF}_{\text{node}} = \begin{cases} 2, & \text{for 2D images} \\ 3, & \text{for 3D images.} \end{cases} \quad (3.32)$$

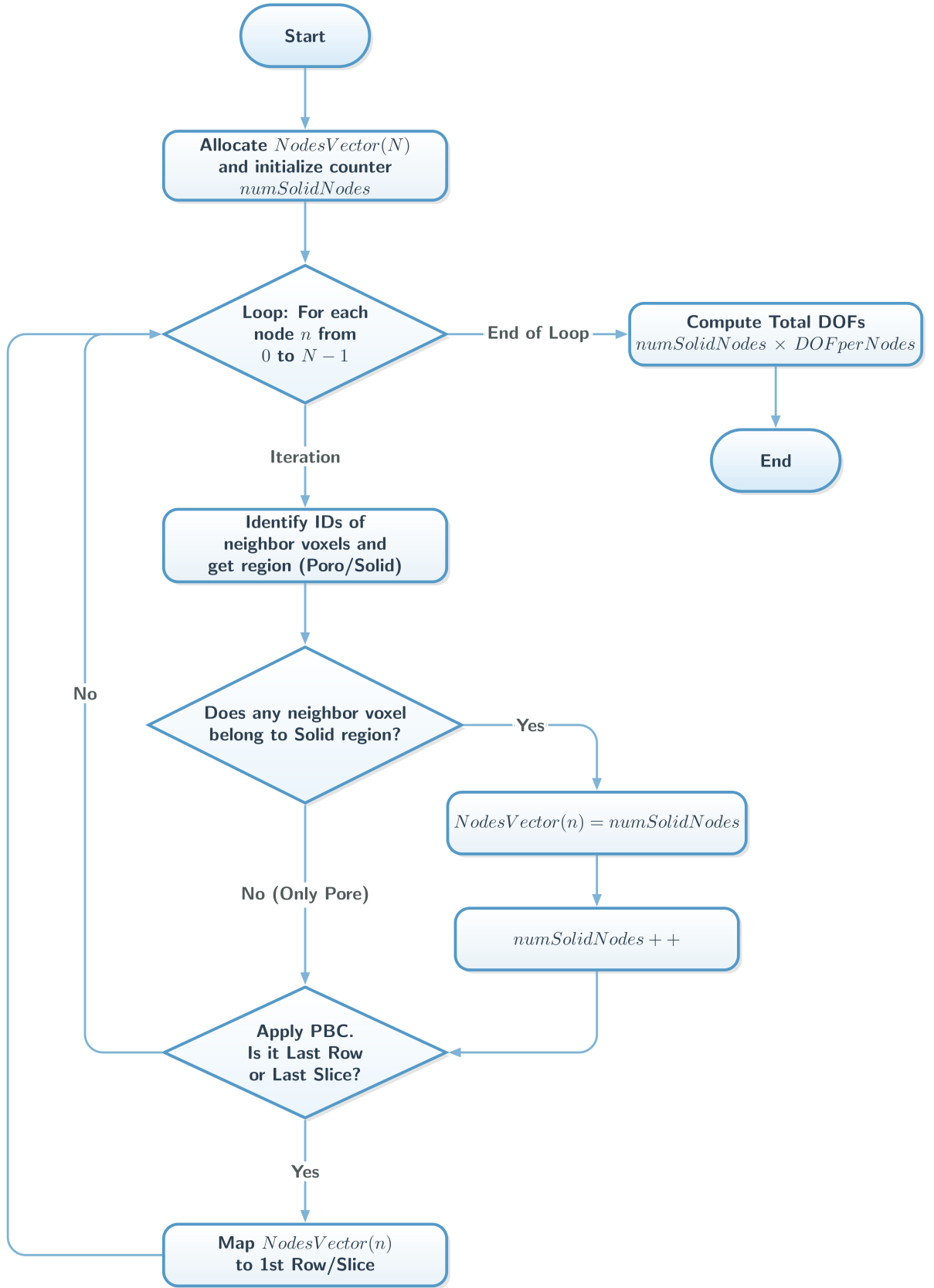


Figure 9: Flowchart of the Active Node Mapping algorithm, including periodic boundary condition (PBC) mapping.

### 3.4.4 Periodic Boundary Conditions

In image-based analysis, the Representative Volume Element (RVE) [81] is a volume selected from the image data that captures the statistically relevant microstructural heterogeneities of a porous medium. The RVE must be sufficiently large to include the diversity of grains, inclusions, and pores while remaining computationally tractable. In static computational homogenization, the boundary conditions affect the computed effective elastic properties. This effect is well documented by [82, 83, 84]; Figure 10 illustrates the convergence of static effective properties for different boundary conditions. In particular, Nguyen et al. [85] showed that periodic boundary conditions can improve the convergence of static homogenized properties compared with Dirichlet and Neumann conditions.

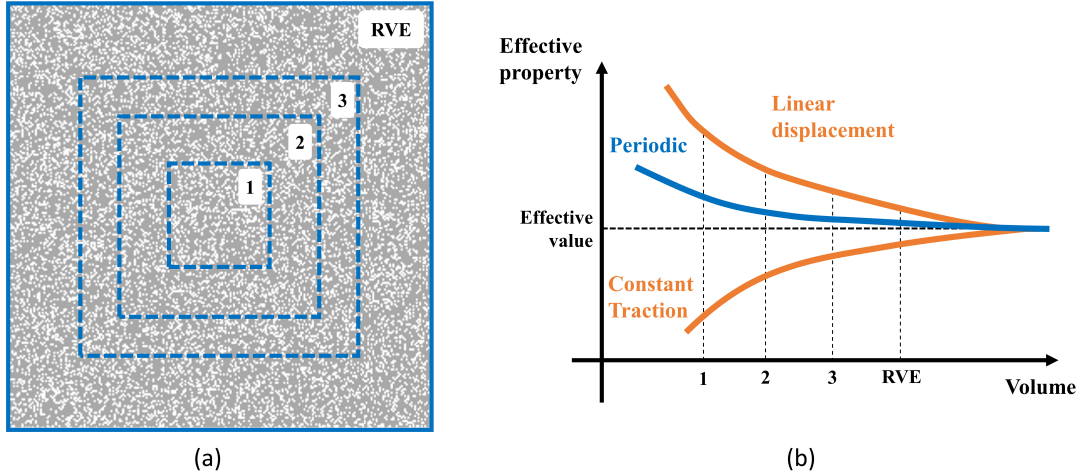


Figure 10: Static convergence of effective properties: (a) different volume sizes of interest; (b) convergence of effective properties according to boundary conditions. Source: Adapted from Nguyen et al. [85].

For wave propagation, representativeness is additionally frequency dependent. The dynamic effective response is valid when there is a separation between the characteristic microstructural length  $l_{\text{micro}}$ , the computational volume size  $L$ , and the shortest wavelength contained in the excitation  $\lambda_{\text{min}}$ , such that  $l_{\text{micro}} \ll L \ll \lambda_{\text{min}}$  [86, 12]. Under this condition, convergence of measured  $V_P$  and  $V_S$  as  $L$  increases indicates that finite-size effects are controlled. If this condition is not satisfied, scattering and dispersion can make the apparent properties frequency dependent [12, 10].

The PBC allows simulating the RVE within a repeating periodic region. The volume of interest is replicated infinitely in the periodic directions, as illustrated in Figure 11 on the left, thereby reducing transverse edge effects. In this work, PBCs are applied only to

the faces perpendicular to the Gaussian wave's propagation, as demonstrated in Figure 11 on the right. This configuration follows the wave-propagation model of Saenger et al. [87]: buffer zones and sensors occupy the two faces normal to propagation, making PBCs on those faces. Corresponding DOFs on opposite periodic boundaries are constrained to equal values.

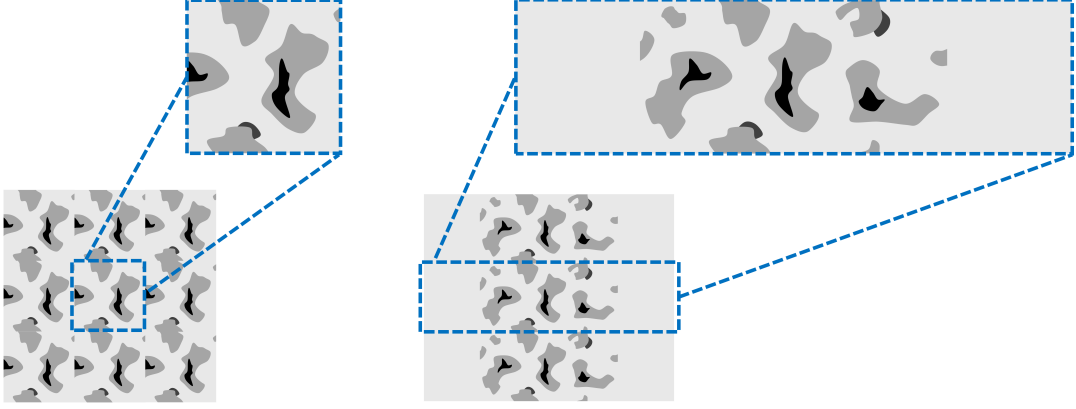


Figure 11: On the left, is a square unitary cell with periodic boundary conditions around all faces. On the right, is a rectangular unitary cell with periodic boundary conditions on the faces perpendicular to the Gaussian wave's propagation.

The suitability of the selected volume and PBC treatment is evaluated in Chapter 5 through convergence studies on real digital rock images. The analysis tracks the wave velocities  $V_P$  and  $V_S$  as the domain size increases, following the dynamic pulse-propagation approach employed in the digital-rock benchmark of Andrä et al. [14].

### 3.5 Ultrasonic Pulse Transmission Model

UPT is a commonly utilized technique in non-destructive laboratory tests for measuring material properties, particularly in the context of routine applications in the oil and gas industry. According to ASTM [4] and ISRM [5], UPT involves the propagation of ultrasonic pulses through a material, with subsequent analysis of the transmitted signals to gather information about its internal structure. Over the years, ASTM and ISRM have established standardized procedures and guidelines for conducting UPT assessments, ensuring consistency, reliability, and comparability of results across different applications. These standards play a crucial role in promoting best practices, quality control, and the advancement of UPT as a valuable tool in material characterization and non-destructive evaluation.

In this work, the approach chosen to simulate UPT using FEM mirrors the standardized



experimental test. The fundamental concept involves propagating a known wave pulse through the material and observing the time it takes for this pulse to traverse it. The transmitted pulse manifests in two forms: the P-wave and the S-wave. Notably, our proposed test shares striking similarities with the methodologies employed by Saenger and Bohlen [68] and Zhu and Shan [88]. Despite these resemblances, our primary objective is to maintain high accuracy while significantly reducing computational time. To attain this goal, the main implemented differences lie in the type of numerical method employed, the approach to time integration, and optimizations related to memory and processing.

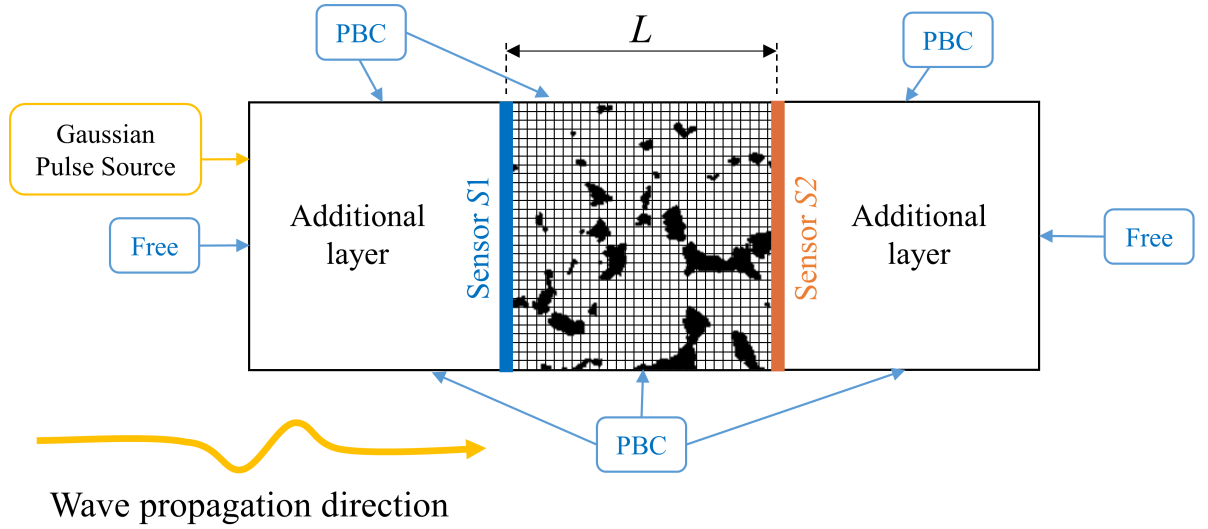


Figure 12: Illustration of the digital rock 2D model in a numerical Ultrasonic Pulse Transmission test using additional layers.

An illustrative example showcasing the modeling of UPT can be found in Figure 12. The model includes a digital rock of size  $L$  at the center (solids and pores) and additional buffer zones within the problem domain. The figure also highlights the direction of the Gaussian pulse propagation and the placement of sensors that help calculate the effective property. These buffer zones, a well-known modeling technique [89, 90] added to the model, play a role in smoothing the interface between boundary conditions and the digital rock, minimizing potential numerical errors from abrupt discontinuities at the interface. In line with the pixel-based model adopted for the digital rock, the same choice extends to the buffer zone, as described in Section 3.4. It is worth noting that in our modeling, the material assigned to the buffer zone aligns with the material exhibiting the highest proportion in the rock. Consequently, the interface experiences minimal reflection effects when the Gaussian pulse arrives. However, the incorporation of buffer zones may increase the number of elements in the model, making the simulation process more time-consuming and memory-intensive. In future work, we are exploring the use of alternative strategies to

eliminate or minimize the need for buffer zones, such as Perfectly Matched Layers (PML), which create an additional layer designed to absorb wave propagation without reflection [91, 92].

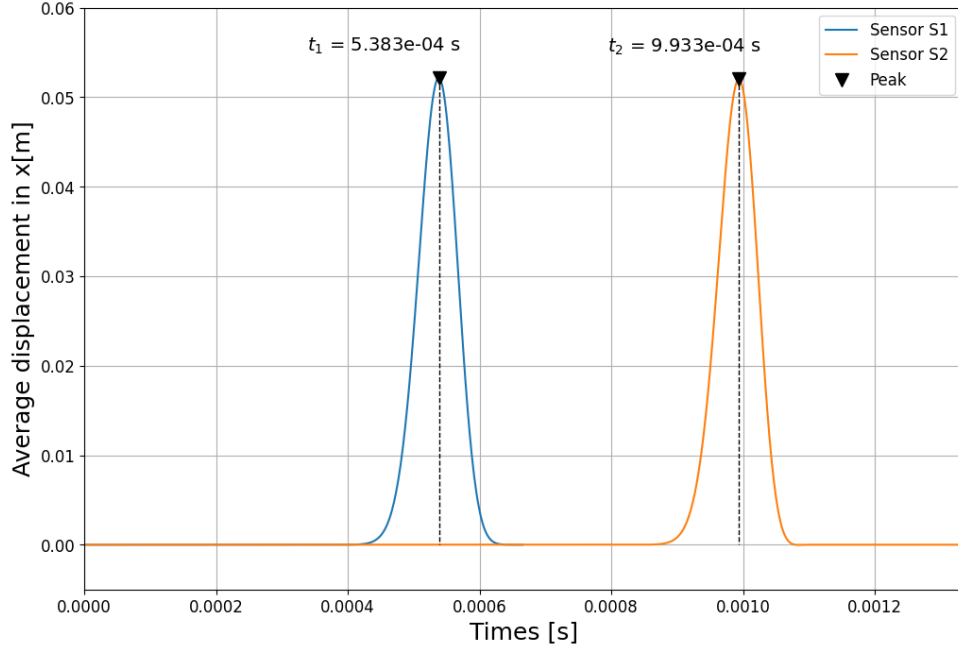


Figure 13: Example of the average displacement versus time sensors curves in a P-wave simulation.

The Gaussian function, known for its smoothness and ease of manipulation, exhibits particular properties that make it interesting to use in this modeling [93]. The pulse is characterized by two adjustable parameters: amplitude and width, allowing for adjustments to meet the specific needs of the problem, and providing flexibility in modeling different pulses depending on digital rocks. Moreover, two essential characteristics stand out: the minimum side spectrum property, concentrating most of the energy at the pulse peak, and the preservation of shape during propagation in linear systems. These features are particularly useful in identifying the moment when the pulse reaches the sensors as shown in Figure 13. Besides that, the preference for using a continuous pulse is also justified for reasons related to precision, numerical stability, and computational efficiency. Additional information about the Gaussian pulse can be found in Appendix C.

Finally, to determine the effective velocities  $V_P$  and  $V_S$  of the porous medium, it is necessary to analyze the average displacements versus time curves recorded by both sensors, as illustrated in Figure 13. The displacement components recorded at each sensor depend on the wave type being simulated: for P-wave simulations, displacements in the  $x$ -direction are monitored; for  $V_{Sy}$  (S-wave with shear in the  $y$ -direction), displacements in the  $y$ -

direction are recorded; and for  $V_{Sz}$  (S-wave with shear in the  $z$ -direction), displacements in the  $z$ -direction are registered. By examining the peaks in the curves obtained from Sensors 1 and 2, we can determine the transit times for both wave types, denoted as  $\Delta t_P$  and  $\Delta t_S$ . These transit times correspond to the duration it takes for the Gaussian pulse to traverse the length  $L$  of the porous medium, as shown in Figure 12. Calculated by finding the difference in peak arrival times between the second sensor ( $t_2$ ) and first sensor ( $t_1$ ) (i.e.,  $\Delta t = t_2 - t_1$ ), the effective wave velocities can then be expressed as

$$V_P = \frac{L}{\Delta t_P}, \quad (3.33)$$

$$V_S = \frac{L}{\Delta t_S}. \quad (3.34)$$

Unlike the system's global variables, the memory allocation for the sensor arrays in the implementation of this thesis exhibits a distinct growth behavior. As detailed in Table 1, while the memory allocation for sensors based on the average of the displacement mean values ( $S1_{mean}$ ,  $S2_{mean}$ ) incurs a linear computational cost with respect to the number of time steps  $N_t$ , the implementation that stores all sensor displacements on the face ( $S1_{max}$ ,  $S2_{max}$ ) grows quadratically with the size of the model. This occurs because the dimension of these arrays is the product of the number of nodes on the boundary faces ( $n_{face} = n_y \times n_z$ ) and the total number of time steps ( $N_t$ ). In this way, the approach adopted in this thesis uses vectors of average displacements, thereby reducing total memory consumption and improving GPU resource efficiency during simulation.

Table 1: Memory allocation for sensor displacement history.

| Vector                 | Array dimension             |
|------------------------|-----------------------------|
| $S1_{mean}, S2_{mean}$ | $1 \times N_t$              |
| $S1_{max}, S2_{max}$   | $n_y \times n_z \times N_t$ |

## 3.6 Parallel Computing with CUDA

As discussed in the preceding sections, the assembly-free EbE approach provides a framework for bypassing prohibitive memory constraints associated with traditional FEM implementations for static [16, 17] and now for dynamics problems. However, the temporal

complexity of time-integration schemes could be a critical bottleneck in practice. A single dynamic simulation may require multiple time steps, during which the Leap-Frog integrator must repeatedly compute stiffness-displacement products over millions of degrees of freedom. On conventional multi-core CPUs, this translates to wall-clock times of hours or even days, rendering many practical investigations infeasible within reasonable computational resources.

GPUs have emerged as a transformative technology for accelerating numerically intensive workloads in computational mechanics. Originally designed to parallelize graphics rendering pipelines, modern GPU architectures have proven equally effective for accelerating dense linear algebra and spatially local computations inherent to finite element codes [94, 95]. The key advantage lies in their massively parallel structure: thousands of lightweight processing cores executing identical instructions simultaneously over different data elements, enabling speedups over CPU implementations for comparable problems.

This work leverages CUDA to implement a purpose-built GPU solver for the elastodynamic assembly-free method described in Section 3.2.2. The implementation departs from existing GPU FEM codes by combining several design choices specifically tailored to dynamic simulations of heterogeneous porous media: (i) elimination of global matrix storage through EbE parallelization, (ii) efficient management of memory using only one phase of porous medium, (iii) atomic operations to resolve race conditions, and (iv) kernel scheduling to maximize hardware occupancy. These optimizations collectively enable simulations that would otherwise exceed the memory capacity of the domestic GPU while maintaining sustained high performance throughout the time-integration loop.

The remainder of this section details the GPU architecture exploited in this implementation, strategies for optimizing memory access patterns, the specific CUDA kernels developed, and the handling of race conditions.

### 3.6.1 The GPU Architecture

The fundamental distinction between CPU and GPU computing architectures lies in their hardware organization and execution models, as illustrated in Figure 14. CPUs are designed to maximize the performance of sequential, latency-sensitive operations through deep instruction pipelines and sophisticated branch prediction. In contrast, GPUs prioritize throughput over latency by equipping thousands of lightweight processing cores, organized into Streaming Multiprocessor (SM), each of which schedules threads in groups of 32 called warps. The execution model exploited by CUDA is more accurately described as Single

Instruction, Multiple Threads (SIMT) [96]. The SIMT model makes GPUs exceptionally well-suited for highly parallelizable workloads, such as EbE assembly operations in FEM, where independent threads can process disjoint elements with minimal synchronization overhead.

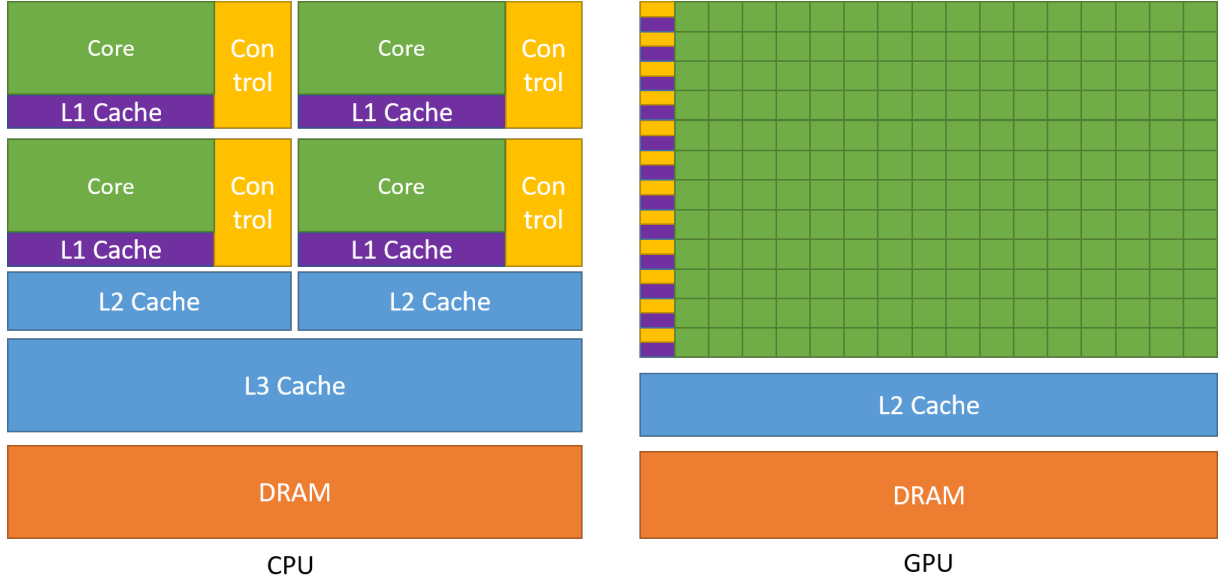


Figure 14: Comparison of CPU and GPU architectures, illustrating differences in core organization and computational resource allocation. Source: Adapted from NVIDIA [96].

CUDA exposes GPU parallelism through a hierarchical thread organization: individual threads are grouped into blocks, and blocks are organized into grids. In the context of the present solver, the parallelization strategy employs a one-dimensional grid of one-dimensional blocks:

$$\text{blockSize} = 256 \text{ threads per block}, \quad \text{numBlocks} = \lceil \frac{N}{256} \rceil, \quad (3.35)$$

where  $N$  denotes either the total number of elements or the total number of degrees of freedom, depending on the kernel. Each thread maps to a single work unit (one element or one DOF). The value of 256 threads per block was adopted as the default configuration after benchmarking representative block sizes (128, 256, and 512) on the available GPUs, since it consistently yielded the best occupancy-to-overhead trade-off across the tested hardware. Nevertheless, the current implementation also allows users to test different block sizes and select the configuration best suited to their own hardware.

### 3.6.2 Host and Device Model

The Host-Device model partitions computation and memory between the CPU (Host) and GPU (Device). For the dynamic solver, the Host initializes data transfer, launches kernels, and manages overall program flow, while the Device executes parallel kernels and maintains all problem data during the simulation. Explicit data movement occurs only at initialization (Host-to-Device) and termination (Device-to-Host), minimizing the PCIe bus bandwidth bottleneck. A critical aspect of the implementation is that the entire time-integration loop remains on the GPU, with no intermediate transfers to the Host. This design choice eliminates the dominant performance penalty that would arise if displacement or force vectors were transferred back to Host at each time step.

As summarized in the flowchart of Figure 15, the execution begins on the Host with setup and validation, followed by the initial Host-to-Device transfer of model data and constant parameters. On the Device side, arrays are allocated and initialized, and the lumped mass field is computed before entering the main transient loop. The time-integration stage is then executed entirely on the GPU through the sequence of kernels for pulse application, displacement update, internal-force evaluation, velocity update, and sensor recording, while the Host remains responsible for launch orchestration. After the final time step, only the required outputs are transferred back to the Host for post-processing, which closes the workflow.

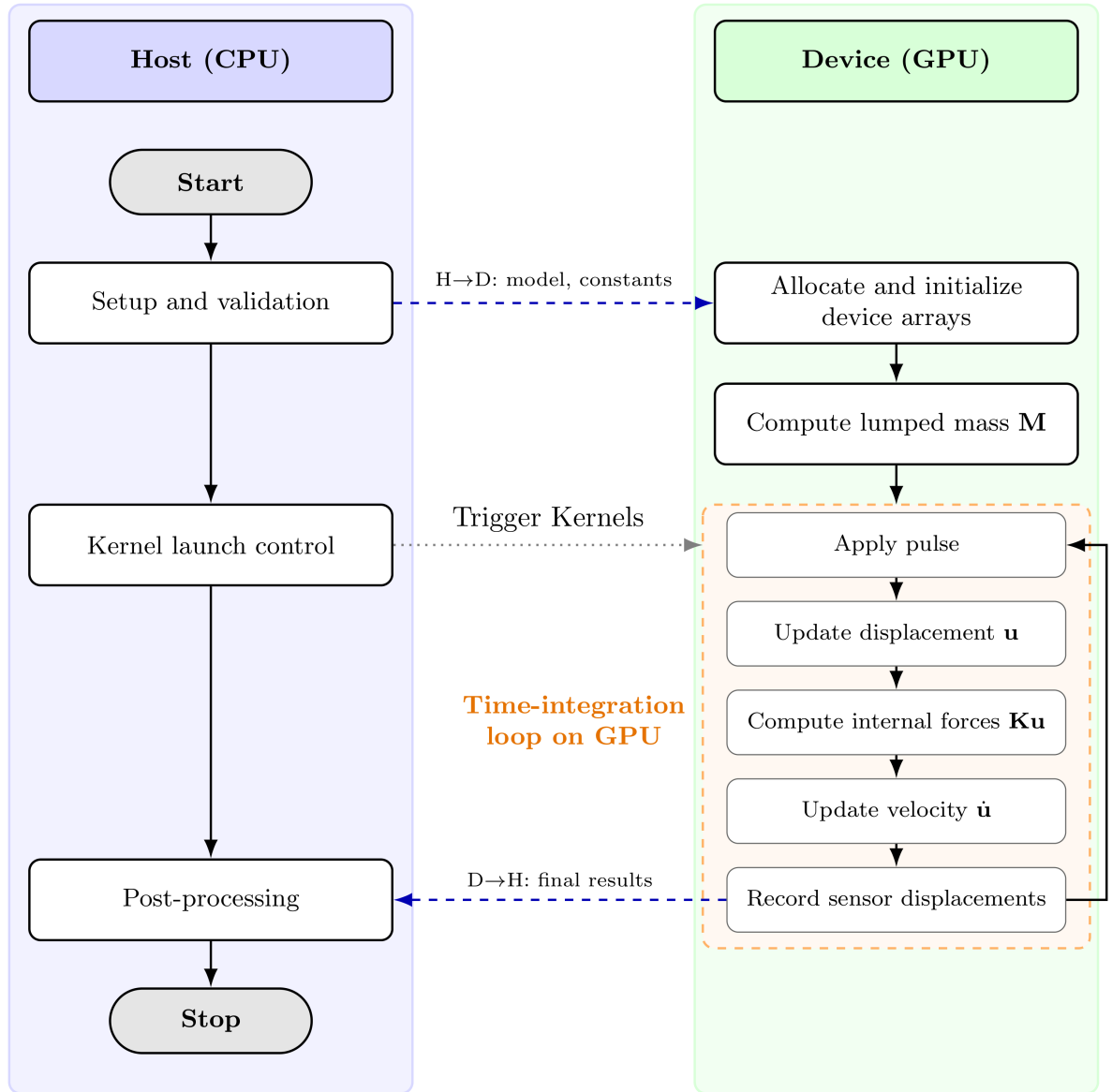


Figure 15: Summarized Host-Device execution model. All time-critical operations are encapsulated within the GPU loop to minimize PCI-e overhead.

### 3.6.3 Memory Hierarchy

The GPU memory hierarchy comprises four key levels, each with distinct latency and bandwidth characteristics:

1. **Registers:** The fastest memory tier, private to each thread, with effectively zero latency. The SM provides each thread with a limited register pool (typically 32–64 KB per SM). Register allocation is implicit during compilation, but careful algorithmic design can exploit registers for accumulating local contributions before global writes.

2. **Shared Memory:** Fast, per-block memory ( $\sim 48$  KB per SM in modern architectures) with single-digit cycle latency. Shared memory is visible to all threads within a block and serves as an intermediate cache for cooperative data reuse. In the present implementation, shared memory is not extensively leveraged because the EbE kernel has minimal data reuse across threads.
3. **Global Memory:** Large but slow memory (typically 8–80 GB per GPU) with latency on the order of hundreds of cycles. All persistent problem data, such as displacement vectors, velocity vectors, stiffness matrices, and sensor histories, reside in global memory. Access performance depends critically on memory access patterns; sequential (coalesced) access to global memory can sustain peak bandwidth, while scattered or strided access incurs significant performance degradation.
4. **Constant Memory:** A read-only, cached memory segment (64 KB per GPU) used to store frequently accessed model parameters. The implementation packs all scalar FEM parameters (grid dimensions, time step, voxel size, element/DOF counts, material flags) into a single constant-memory struct.

An additional optimization exploited in the present work is the use of constant memory broadcasts. At the start of each analysis, the host copies this struct to the GPU’s constant memory via `cudaMemcpyToSymbol`. During kernel execution, when all threads in a warp (32 threads executing in lockstep) read the same constant parameter, the GPU broadcasts a single value to all threads in a single cache cycle. This eliminates redundant global memory accesses that would otherwise serialize accesses to the same parameter across many threads. For a grid-scale simulation with millions of elements, this broadcast optimization accumulates substantial latency savings.

Based on this memory organization, the next subsection details how each CUDA kernel maps the FEM operations to parallel work units while controlling memory traffic and synchronization overhead.

### 3.6.4 CUDA Kernels

Rather than assembling a monolithic GPU routine, the implementation is structured as a sequence of specialized kernels that mirror the numerical workflow at each time step: pulse application, displacement update, stiffness-displacement evaluation, velocity update, and sensor-history recording. In practical terms, the transient loop is executed by kernels such as `applyPulse_kernel`, `updateDisplacement_kernel`, `productKU_kernel`,



`updateVelocity_kernel`, and `history_kernel`. The code was implemented directly with native CUDA C/C++ primitives, without relying on internal high-level libraries (e.g., cuBLAS, cuSPARSE, or Thrust), which preserves full control over arithmetic order, synchronization points, and data movement.

The implementation adopts mixed precision: the main state vectors are stored in single precision (`float`), while selected accumulation steps are performed in double precision before the final write-back, balancing higher throughput with reduced accumulation error.

In addition to the time-integration kernels, the solver includes preprocessing kernels for connectivity and sensor indexing (for example, `generateDOFmap_kernel` and `sensor_kernel`), as well as mass-related kernels (`lumpedMass_kernel` and `invertMass_kernel`). For the matrix-free EbE product, multiple kernel variants were developed and benchmarked to identify the most efficient solution. This modular decomposition follows the recommended CUDA software-design practices [96]: each kernel has a narrow responsibility, can be profiled independently, and can be tuned without changing the mathematical formulation of the FEM solver.

### 3.6.5 Race Conditions and Atomic Operations

In CUDA-based FEM kernels, race conditions arise naturally during the scatter stage of the EbE formulation. The root cause is topological: neighboring elements share nodes, and therefore multiple GPU threads may attempt to update the same global degree of freedom at the same time. Following the CUDA execution model described by Guide [96], threads within and across warps execute independently, so unsynchronized read-modify-write operations on the same address are not deterministic.

For the assembly-free product  $\mathbf{v} = \mathbf{K}\mathbf{u}$ , each element contributes a local vector  $\mathbf{v}_e$  to the global vector. Programmatically, this corresponds to repeated updates of the form

$$v_i \leftarrow v_i + \Delta v_i^{(e)}, \quad (3.36)$$

where different elements  $e$  may target the same global index  $i$ . Without synchronization, one thread can overwrite another thread's partial sum, producing non-reproducible and physically incorrect internal forces.

To guarantee correctness, the implementation uses CUDA atomic additions in all concurrent accumulation points (for example, in the stiffness-displacement product and in

the lumped-mass assembly). In practical terms, the operation

$$v_i = \sum_{e \in \mathcal{E}_i} \Delta v_i^{(e)}, \quad (3.37)$$

where  $\mathcal{E}_i \subseteq \{1, 2, \dots, E\}$  denotes the subset of all elements that share the global degree of freedom  $i$ , is enforced through atomic updates to each shared address, ensuring that every contribution is incorporated exactly once. This strategy follows the recommended CUDA synchronization semantics for concurrent global-memory writes [96].

Finally, the implementation contains a relevant accumulation pattern to deal with race conditions. In the direct element kernel, each thread may execute up to 72 atomic updates for one element contribution ( $8 \text{ local nodes} \times 3 \text{ row components} \times 3 \text{ column components}$ ). In the optimized kernel, local contributions are first accumulated in registers and only then committed to global memory, reducing the number of atomic operations to 3 per thread (one per displacement component). This redesign reduces contention while preserving algebraic equivalence with the original FEM formulation. Consequently, performance depends on both hardware behavior and mesh topology: highly connected regions produce stronger contention, whereas more uniformly distributed updates improve throughput. For this reason, the present implementation combines atomic correctness with contention-reduction strategies, including local register accumulation, kernel decomposition, and minimization of redundant global updates.

## 4 Proposed Implementations

In this chapter, we introduce and explore the sequential implementation of the algorithm designed to model the UPT using image-based Finite Element Method in the time domain, as outlined and formulated in Chapter 3. To enhance understanding and facilitate implementation, we will provide a pseudocode for this algorithm, including the allocation of global matrices in memory.

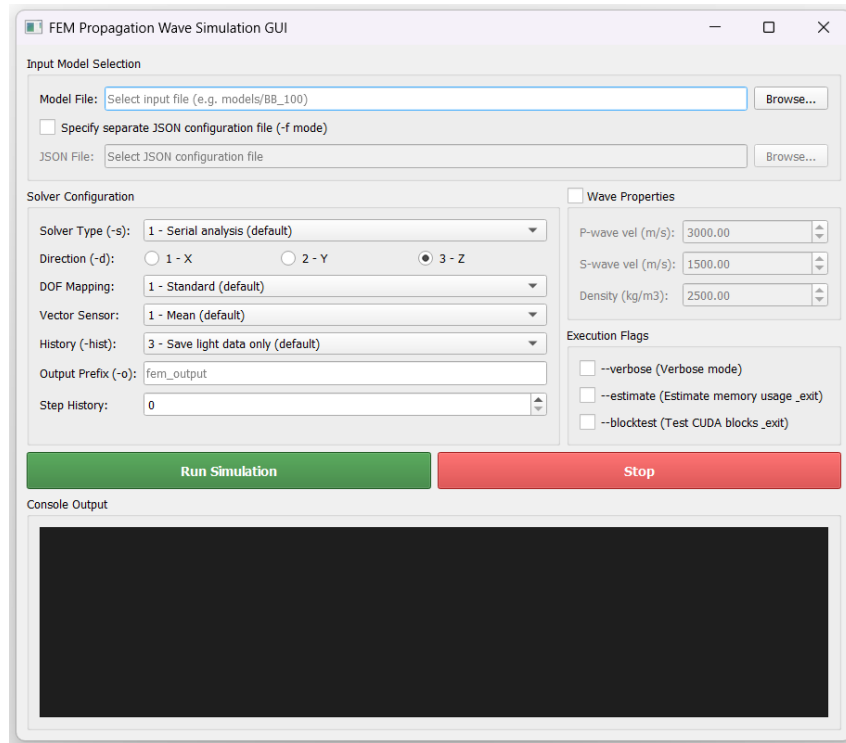


Figure 16: Graphical User Interface (GUI) developed for the FEM UPT simulator.

Before proceeding with the GPU implementation, accessible through the GUI shown in Figure 16, the algorithm was prototyped on the CPU using the Julia language [97]. Julia was chosen for its ability to bridge the gap between high-level ease of use and low-level performance, offering execution speeds comparable to C while maintaining the productivity of Python. Additionally, the language's friendly and expressive syntax further contributes to the ease of writing concise and readable code.

The implementation followed an incremental development and cross-validation strategy, summarized in Figure 17. The first step was the development of a dynamic FEM solver and its validation against an analytical solution (Sec. 5.1), confirming the correctness of the time integrator. Building on this, a 2D Julia prototype was implemented, and the effective properties of the digital rocks were compared with a static FEM (Sec. 5.2). The solver was then extended to 3D, with the same static comparison approach used for verification. Once the algorithm was consolidated in Julia, it was ported to the C language to improve raw single-thread performance, with the static solution serving as a baseline. Two parallel execution strategies were subsequently developed on top of the C implementation: a CUDA GPU solver, cross-validated with the dynamic C results to confirm numerical equivalence on a different architecture, and an Open Multi-Processing (OpenMP) multi-thread CPU solver, benchmarked with the CUDA outputs to assess consistency between the parallel CPU and GPU paths.

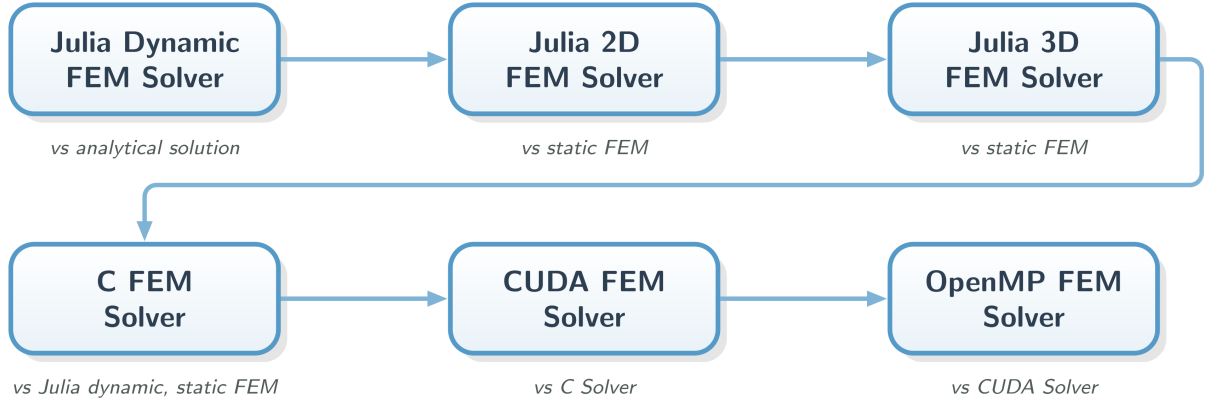


Figure 17: Implementation evolution and validation chain. Each node represents an implementation stage; the label below indicates the reference used for cross-validation.

## 4.1 Time-Domain Image-Based Finite Element Method

There are several ways to implement the UPT test using the image-based FEM in the time domain for 2D or 3D problems. One possible approach is presented as simplified pseudocode in Algorithm 2. For this UPT implementation, the input parameters include an *image* representing the material structure (solids and pores), a *buffer zone* specification, the solids properties, spatial discretization size  $\Delta L$ , total simulation time  $T_f$ , and  $\Delta t$  the time step size.

**Algorithm 2** Ultrasonic Pulse Transmission using Time-Domain Image-Based FEM**Input:** `image`, `materialInfo`, buffer zone, Pulse properties,  $\Delta t$ ,  $N_t$ **Output:**  $V_P$ ,  $V_S$ 

- 1: Create **model** based on **image**, buffer zone, and **materialInfo** ▷ Fig. 12
- 2: Generate DOFs (DNM or ANM, **DOFMAP** or Element-wise) ▷ Sec. 3.4.2 and 3.4.3
- 3:  $CFL \leftarrow \frac{V_P \Delta t}{\Delta L}$
- 4: **if**  $CFL > 1$  **then**
- 5:     **terminate** execution ▷ CFL condition violated (see Eq. 3.26)
- 6: **end if**
- 7: **if**  $\Delta L > \frac{V_{S,min}}{f_{max} N_{min}}$  **then**
- 8:     **terminate** execution ▷ Spatial dispersion violated (see Eq. 3.27)
- 9: **end if**
- 10: Compute local stiffness matrices  $\mathbf{k}_e$  and lumped mass matrix inverse  $\mathbf{M}^{-1}$
- 11: Apply initial pulse to displacement  $\mathbf{u}$  at  $t = 0$  ▷ **Initialization phase** ( $t = 0$ )
- 12:  $\mathbf{v} \leftarrow -\mathbf{K}\mathbf{u}$  ▷ Initial internal forces
- 13:  $\dot{\mathbf{u}} \leftarrow \dot{\mathbf{u}} + \mathbf{M}^{-1}\mathbf{v}\frac{\Delta t}{2}$  ▷ Half-step velocity (Leap-Frog offset)
- 14:  $\mathbf{S}_1[0] \leftarrow \mathbf{u}$ ,  $\mathbf{S}_2[0] \leftarrow \mathbf{u}$  ▷ Record sensors at  $t = 0$
- 15: **for**  $n \leftarrow 1$  **to**  $N_t - 1$  **do** ▷ **Time integration loop**
- 16:     Apply pulse to displacement  $\mathbf{u}$  at step  $n$  ▷ Update Dirichlet BCs
- 17:      $\mathbf{u} \leftarrow \mathbf{u} + \dot{\mathbf{u}}\Delta t$  ▷  $\mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \dot{\mathbf{u}}(t + \frac{\Delta t}{2})\Delta t$
- 18:      $\mathbf{v} \leftarrow -\mathbf{K}\mathbf{u}$  ▷ Recompute internal forces for new  $\mathbf{u}$
- 19:      $\dot{\mathbf{u}} \leftarrow \dot{\mathbf{u}} + \mathbf{M}^{-1}\mathbf{v}\Delta t$  ▷  $\dot{\mathbf{u}}(t + \frac{\Delta t}{2}) = \dot{\mathbf{u}}(t - \frac{\Delta t}{2}) + \ddot{\mathbf{u}}(t)\Delta t$
- 20:      $\mathbf{S}_1[n] \leftarrow \mathbf{u}$ ,  $\mathbf{S}_2[n] \leftarrow \mathbf{u}$  ▷ Record sensors at step  $n$
- 21: **end for**
- 22: Calculate  $V_P$  and  $V_S$  based on sensor histories  $\mathbf{S}_1$  and  $\mathbf{S}_2$

To begin the simulation, the first step is to create a *model* that includes both the *image* and the *buffer zone*. This model is constructed from two input files: a RAW file containing the volumetric image data and a JSON file specifying the simulation properties. The input files can be generated using the *pytomoviewer* toolkit, which is available on GitHub [19]. The buffer zone refers to an area adjacent to the region of interest that is kept free of obstacles or materials that could interfere with the propagation of ultrasonic pulses. It is important to note that during the creation of the model, only the solid phase is taken into consideration. Figure 18 presents a representative example of the JSON file used to configure the dynamic image-based FEM solver, and Table 2 describes the

meaning of each field.

```
{
  "image_dimensions": [100, 100, 100],
  "voxel_size": 1.0,
  "buffer": 100,
  "npulse": 100,
  "solid": 255,
  "poro": 0,
  "number_of_materials": 2,
  "properties_of_materials": [
    [0, 0.00, 0.00, 0.00],
    [255, 94530000000.0, 0.07, 2650.0]
  ],
  "total_time": 0.075,
  "time_discretization": 600
}
```

Figure 18: Example of the JSON input file used to configure the dynamic image-based FEM solver.

Table 2: Description of the JSON fields used to build the model of Algorithm 2.

| Field                   | Type   | Description                                                                                                                                     |
|-------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| image_dimensions        | int[3] | Number of voxels $n_x$ , $n_y$ , $n_z$ of the digital rock image.                                                                               |
| voxel_size              | float  | Spatial discretization size $\Delta L$ , i.e., the physical size of the voxel element edge (m).                                                 |
| buffer                  | int    | Number of solid layers added on each side of the image along the pulse propagation direction, defining the buffer zone.                         |
| npulse                  | int    | Duration of the excitation pulse.                                                                                                               |
| solid                   | int    | Material ID (0–255) assigned to the buffer region.                                                                                              |
| poro                    | int    | Material ID (0–255) corresponding to the pore phase.                                                                                            |
| number_of_materials     | int    | Number of distinct material identifiers present in the image.                                                                                   |
| properties_of_materials | array  | For each material: $[matID, E \text{ (Pa)}, \nu, \rho \text{ (kg/m}^3\text{)}]$ , used to compute the local stiffness matrices $\mathbf{k}_e$ . |
| total_time              | float  | Total simulation time $T_f$ (s).                                                                                                                |
| time_discretization     | int    | Number of time steps $N_t$ , used to compute the time step size $\Delta t = T_f/N_t$ .                                                          |

The second step is to create the degree of freedom mapping of the model through

the DNM or ANM algorithms considering the periodic boundary conditions. The solver calculates local stiffness matrices  $\mathbf{k}_e$  for each solid material and evaluates the critical time step based on the P-wave velocity, spatial step size, and time step size. If the criterion from Equation 3.26 or 3.27 is not satisfied, the solver is terminated. Concluding the initial setup, the lumped mass matrix  $\mathbf{M}^{-1}$  is computed, and the gaussian pulse  $\mathbf{u}_g$  are determined.

The algorithm core is the time integration loop that utilizes the Leap-Frog scheme (C, OpenMP, CUDA), lines 15-21, to update the displacement vector  $\mathbf{u}$  and velocity vector  $\dot{\mathbf{u}}$  iteratively. Within the loop, there are several optimization possibilities, and the works by Bez et al. [18] and Lopes et al. [19] have demonstrated some alternatives. For this work, the focus is optimize the computation of internal forces. The matrix  $\mathbf{K}$  inline 18 stands out as the memory-intensive component, demanding careful consideration due to its memory consumption. Even in sparse format, as image dimensions increase, memory allocation for the matrix  $\mathbf{K}$  grows, making the FEM quickly unfeasible for large-scale simulations. Finally, the algorithm ends up calculating P-wave or S-wave velocities based on the sensor data and chosen direction ( $x$ ,  $y$  or  $z$ ). To calculate in other directions, the simulation must be run again.

## 4.2 Element-by-Element Algorithm

To address the memory allocation problem in updating internal forces in large-scale simulations we proposed the pseudocode in Algorithm 3. This algorithm is rooted in the well-established matrix-free techniques, drawing inspiration from the EbE approach [98, 99]. In practical terms, an EbE algorithm breaks down the complex problem into smaller, more manageable components, addressing each element individually rather than relying on a global matrix for the entire system. By doing this, the EbE approach minimizes memory requirements while increasing computations. Consequently, this methodology not only reduces computational complexity but also enhances overall efficiency, striking a balance in the memory vs. time trade-off, contributing to improved performance and resource utilization for large-scale simulations.

We describe a procedure for computing the matrix-vector product  $\mathbf{K}\mathbf{u}$  or  $\mathbf{v}$  in a finite element simulation, where  $n_{elem}$  is the total number of elements,  $\mathbf{K}$  is the global stiffness matrix,  $\mathbf{u}$  is the vector of nodal displacements, and  $\mathbf{v}$  is the resulting vector of internal forces. The input parameters include the nodal displacements, the local stiffness matrix for each element  $\mathbf{k}_e$ , the global degrees of freedom **DOFMAP**, and the *image*. The algorithm

**Algorithm 3** Image-based Element-by-Element matrix-vector product**Input:** *model*, *u*, *DOFMAP*, pore identifier *poro***Output:** *v*

```

1:  $\mathbf{v} \leftarrow 0$  ▷ Initialize global vector
2: for  $e \leftarrow 0$  to  $n_{elem} - 1$  do ▷ Loop over elements
3:    $matKey \leftarrow \mathbf{model}[e]$  ▷ Get the element material key
4:   if  $matKey = poro$  then ▷ Check if element is a pore
5:     continue ▷ Skip to next element
6:   end if
7:    $\mathbf{k}_e \leftarrow \text{GETSTIFFNESSMATRIX}(matKey)$  ▷ Get local stiffness matrix
8:    $\mathbf{ndofs} \leftarrow \mathbf{DOFMAP}[e]$  ▷ Get indexes of global DOFs
9:    $row \leftarrow 0$ 
10:  for each  $i$  in  $\mathbf{ndofs}$  do
11:     $col \leftarrow 0$ 
12:    for each  $j$  in  $\mathbf{ndofs}$  do
13:       $\mathbf{v}[i] \leftarrow \mathbf{v}[i] + \mathbf{k}_e[row][col]\mathbf{u}[j]$  (atomic) ▷ Update internal forces ( $\mathbf{Ku}$ )
14:       $col \leftarrow col + 1$ 
15:    end for
16:     $row \leftarrow row + 1$ 
17:  end for
18: end for

```

initializes the internal forces vector  $\mathbf{v}$  to zero and then iterates over each finite element in the simulation domain. For each element, it retrieves the material key and uses it to obtain the local stiffness matrix  $\mathbf{k}_e$ . The algorithm then identifies the global degrees of freedom associated with the element and performs a nested loop over these degrees of freedom to compute the matrix-vector product, updating  $\mathbf{v}$ . This process efficiently assembles the contributions from each element to the overall internal forces vector in a finite element simulation.

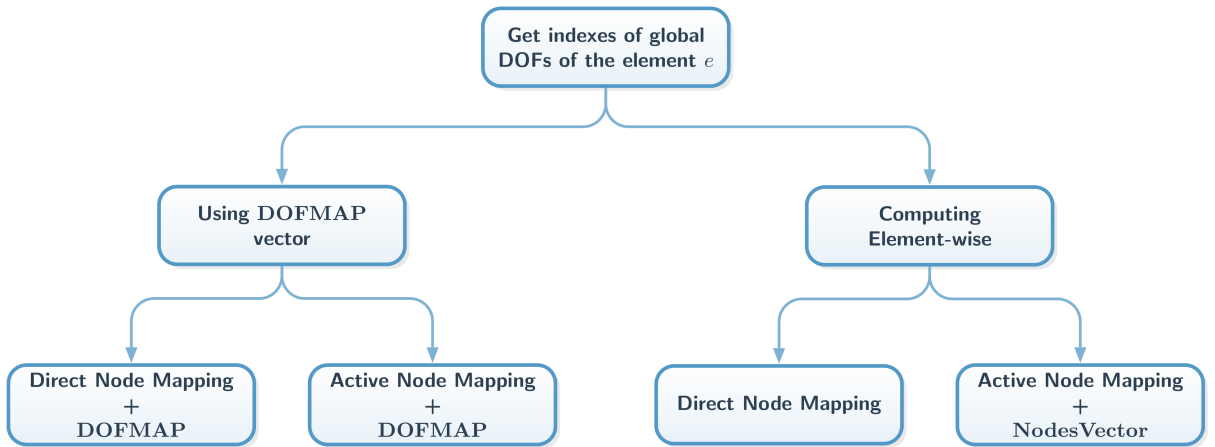


Figure 19: Diagram of possible implementation options for EbE.

Regarding line 8 of the Algorithm 3, four choices can be made, each generating different



solver with distinct execution times and memory requirements. First, we can choose to either precompute a vector **DOFMAP** containing all the DOF of the model or calculate the element's degrees of freedom at every iteration Element-wise. Furthermore, there are two methods of DOF generation: Direct Node Mapping (DNM) and Active Node Mapping (ANM). A visual representation of these paths is illustrated in Figure 19. Regarding the precomputed **DOFMAP** vector path, memory usage varies because the number of DOFs generated by DNM and ANM differs. Specifically, the difference between the maximum number of DOFs  $n_{dofmax}$  using the DNM and the optimized number of DOFs  $n_{dofsolid}$  using the ANM will depend on the porosity and tortuosity properties of the image used. In the Element-wise path, there are additional memory differences. Since the implemented ANM algorithm utilizes all the digital rock information, we must store this data in a **NodesVector**. On the other hand, the Element-wise approach with DNM does not require an additional vector, as each element can calculate its own DOFs using Equation 3.29. The trade-off between these latter two implementations will be evaluated in the following sections.

Table 3 presents the vectors and memory footprints for each possible EbE implementation. Since  $n_{dofmax} \geq n_{dofsolid}$ , the DOFMAP + ANM implementation requires less memory than DOFMAP + DNM. For the two Element-wise implementations, it is not possible, a priori, to determine which one uses less memory. Because the number of degrees of freedom,  $n_{dofsolid}$ , depends on the sample under analysis, the only robust conclusion is that using smaller vectors for  $\mathbf{u}$ ,  $\dot{\mathbf{u}}$ , and  $\mathbf{Ku}$  leads to fewer operations.

Table 3: Memory allocation details for different EbE implementations.

| <b>EbE Type</b>    | <b>Vector</b>                                                                           | <b>Array dimension</b> |
|--------------------|-----------------------------------------------------------------------------------------|------------------------|
| DOFMAP + DNM       | <b>DOFMAP</b>                                                                           | $n_{elem} \times 8$    |
|                    | <b><math>\mathbf{u}</math>, <math>\dot{\mathbf{u}}</math>, <math>\mathbf{Ku}</math></b> | $n_{dofmax}$           |
| DOFMAP + ANM       | <b>DOFMAP</b>                                                                           | $n_{elem} \times 8$    |
|                    | <b><math>\mathbf{u}</math>, <math>\dot{\mathbf{u}}</math>, <math>\mathbf{Ku}</math></b> | $n_{dofsolid}$         |
| Element-wise + DNM | -                                                                                       | -                      |
|                    | <b><math>\mathbf{u}</math>, <math>\dot{\mathbf{u}}</math>, <math>\mathbf{Ku}</math></b> | $n_{dofmax}$           |
| Element-wise + ANM | <b>NodesVector</b>                                                                      | $n_{nodes}$            |
|                    | <b><math>\mathbf{u}</math>, <math>\dot{\mathbf{u}}</math>, <math>\mathbf{Ku}</math></b> | $n_{dofsolid}$         |

Table 4 offers an overview of the variable types and the array dimensions associated

with key variables, based on Algorithms 2 and 3. This table provides insights into the computational aspects of the simulation, allowing users to have a better understanding of the underlying mechanism. Notably,  $\mathbf{u}$ ,  $\dot{\mathbf{u}}$ ,  $\mathbf{M}$ , and  $\mathbf{Ku}$  are single-precision floating-point vectors, each utilizing 32 bits of memory and linked to the number of degrees of freedom denoted as  $n_{dof}$ . The parameter  $n_{dof}$  is defined as  $n_{dofmax}$  or  $n_{dofsolid}$  based on the chosen EbE type. Importantly, these vectors represent the elements that impose the most substantial memory demands in the context under consideration. The **DOFMAP** represents degrees of freedom map and is of type uint32 (32 bits) with an array dimension of  $n_{elem} \times 8$ . Additionally,  $\mathbf{S1}$  and  $\mathbf{S2}$  are single-precision arrays with 32 bits, and their array dimensions could be  $N_t$  or  $n_y \times n_z \times N_t$  as mentioned in Table 1. The variable *image* is a uint8 (8 bits) type where  $n_{elem}$  is the number of voxels, while  $\mathbf{k_e}$  is a single-precision matrix with 32 bits where  $n_{materials}$  is the quantity of different materials.

Table 4: Memory allocated in image-based Element-by-Element implementation.

| Vector                                      | Type             | Array dimension                        |
|---------------------------------------------|------------------|----------------------------------------|
| $\mathbf{u}, \dot{\mathbf{u}}, \mathbf{Ku}$ | float (32 bits)* | $n_{dof}$                              |
| $\mathbf{M}$                                | float (32 bits)* | $n_{nodes}$                            |
| <b>DOFMAP</b>                               | uint32 (32 bits) | $n_{elem} \times 8$                    |
| $\mathbf{S}_1, \mathbf{S}_2$                | float (32 bits)* | $N_t$                                  |
| <b>image</b>                                | uint8 (8 bits)   | $n_{elem}$                             |
| <b>NodesVector</b>                          | uint32 (32 bits) | $n_{nodes}$                            |
| $\mathbf{k_e}$                              | float (32 bits)  | $24 \times 24 \times n_{materials} **$ |

\* **Note:** Although the vectors are stored in single precision (float, 32 bits), selected CUDA kernels use double-precision intermediate arithmetic. \*\* **Note 2:**  $n_{materials} \leq 256$ , corresponding to the maximum range of voxel intensity variations in 8-bit microtomography images.

The EbE CUDA variant that alters the addressing pattern, removing the dependency on a pre-stored global **DOFMAP** and reconstructing local access to solid elements when necessary, has not been reported for dynamic FEM simulations in the reviewed literature [16, 73, 19]. This element-wise reconstruction strategy is one of the main contributions of this work's implementation.

### 4.3 Active Node Mapping Generation

The Active Node Mapping used in the Element-wise + ANM implementations is generated through the `generate_ActiveNodes` routine in Algorithm 4, which creates the **NodesVector** by scanning all nodes in the voxelized image and assigning indices only to nodes connected to at least one solid element. For each node, the algorithm identifies the neighboring elements under periodic boundary conditions, checks whether the surrounding stencil contains at least one solid voxel and then updates the active node numbering accordingly. In addition, the remapping of nodes on the last row and last slice guarantees consistency with the periodic topology before the final number of DOFs is stored in the model.

---

**Algorithm 4** `generate_ActiveNodes` (Active node mapping generation)

---

**Input:** `model`,  $n_x$ ,  $n_y$ ,  $n_z$ ,  $n_{buffer}$ ,  $n_{nodes}$ , `poro` (pore identifier)

**Output:** **NodesVector**,  $n_{dofsolid}$

---

```

1: Allocate NodesVector with size  $n_{nodes}$ 
2:  $numSolidNodes \leftarrow 0$ 
3: for  $n \leftarrow 0$  to  $n_{nodes} - 1$  do
4:   Compute  $i_x$ ,  $i_y$ , and  $i_z$  of node  $n$  ▷ Eq. 3.29
5:   Compute the neighboring elements (back, front, top, and down) of node  $n$ 
6:    $hasSolidNeighbor \leftarrow$  Evaluate 8 elements around node  $n$  ▷ At least one solid
7:   if  $hasSolidNeighbor$  then
8:     NodesVector[ $n$ ]  $\leftarrow numSolidNodes$ 
9:     if node  $n$  is on the last row then
10:      Remap NodesVector[ $n$ ] to the corresponding node in the first row
11:     end if
12:     if node  $n$  is on the last slice then
13:      Remap NodesVector[ $n$ ] to the corresponding node in the first slice
14:     end if
15:      $numSolidNodes \leftarrow numSolidNodes + 1$ 
16:   end if
17: end for
18:  $n_{dofsolid} \leftarrow numSolidNodes \cdot 3$ 

```

---

### 4.4 CUDA Solver Architecture

The CUDA implementation follows a host-orchestrated, device-dominant execution model, as shown in Figure 15. The host (CPU) is responsible for input validation, model construction, and final post-processing, while the heavy numerical workload is executed on the GPU through specialized kernels.

At high level, the CUDA pipeline is organized into three stages:

1. **Preparation stage:** allocation of device buffers, transfer of static data, and initialization of constant memory with model parameters.
2. **Numerical stage:** DOFMAP and sensor index generation, lumped mass computation, and explicit leapfrog time marching on the GPU.
3. **Finalization stage:** transfer of required outputs back to host memory and controlled release/reset of the CUDA runtime state.

This architecture reduces the number of host-device synchronization points inside the time loop and keeps the per-step operations close to the hardware where arithmetic throughput is highest.

To improve robustness and reproducibility across benchmarks, the implementation includes explicit CUDA reliability mechanisms, such as host-side API and kernel-launch error checks, synchronization at critical points, dedicated initialization/finalization routines, and explicit device reset between analyses. In addition, kernel execution is parameterized by block size and problem-dependent grid dimensions, and practical tuning is performed by sweeping representative block sizes (e.g., 128, 256, and 512 threads), and measuring average leapfrog iteration time.

#### 4.4.1 CUDA Standard vs. CUDA Element-Wise

The two GPU modes solve the same physical problem but differ in memory-access strategy:

Table 5: Comparison between CUDA solver modes.

| Aspect         | CUDA standard (mode 3)    | CUDA element-wise (mode 4)         |
|----------------|---------------------------|------------------------------------|
| Topology data  | Explicit global DOFMAP    | Element-local indexing/NodesVector |
| Memory profile | Higher indexing storage   | Reduced global mapping storage     |
| Access pattern | Map-driven gather/scatter | Element-oriented accumulation      |
| Typical use    | Small rock samples        | Memory-sensitive large cases       |

The choice between modes is not purely about speed; it is about balancing memory capacity, indexing overhead, and access locality for a given mesh size and hardware profile.

### 4.4.2 Benchmarking and Cross-Validation Methodology

Benchmarking and cross-validation were conducted under a verification-and-validation (V&V) framework [100], in which performance and numerical fidelity are assessed jointly:

- **Performance:** wall-clock runtime, average time per time step, and speedup relative to serial and OpenMP baselines.
- **Numerical:** waveform agreement at sensors, arrival-time consistency, and normalized error against CPU references.

Speedup can be reported as

$$S_p = \frac{T_{\text{CPU}}}{T_{\text{CUDA}}}, \quad (4.1)$$

and a normalized waveform error at a sensor can be reported as

$$\varepsilon_2 = \frac{\|\mathbf{u}_{\text{ref}} - \mathbf{u}_{\text{test}}\|_2}{\|\mathbf{u}_{\text{ref}}\|_2}. \quad (4.2)$$

In this thesis, this dual-axis methodology is essential: acceleration alone is insufficient if not accompanied by controlled agreement with trusted baselines. The serial and OpenMP solvers are therefore not only alternatives for execution, but reference instruments for validating CUDA behavior.

## 4.5 CUDA Kernels

This section details the CUDA kernels that implement the UPT workflow presented in Chapter 2 and discretized in Chapter 3. Instead of a single monolithic GPU routine, the solver is decomposed into specialized kernels that follow the numerical pipeline: topology mapping, mass processing, leapfrog updates (internal force, displacement, and velocity), and sensor history recording.

### 4.5.1 DOF Mapping Kernels

This kernel builds the global DOF connectivity under periodic boundary conditions, including edge and corner remapping. The mapping is generated per element and supports

both Direct Node Mapping and Active Node Mapping (*NodesVector*) through the runtime parameter *dofmap\_type*.

---

**Algorithm 5** generateDOFMAP\_kernel (Global DOFMAP with PBC)
 

---

**Input:** NodesVector, *dofmap\_type*

**Output:** DOFMAP

```

1:  $e \leftarrow$  global thread index
2: if  $e \geq n_{elem}$  then
3:   return
4: end if
5: for  $i \leftarrow 0$  to 7 do
6:    $node \leftarrow$  GETLOCALMAP( $i, dofmap\_type$ )     $\triangleright$  Local 8-node map (Direct/Active)
7:   if last row in  $y$  then
8:      $node \leftarrow$  Apply periodic remap in  $y$ 
9:   end if
10:  if last slice in  $z$  then
11:     $node \leftarrow$  Apply periodic remap in  $z$ 
12:  end if
13:  if corner case  $y$  and  $z$  periodic then
14:     $node \leftarrow$  Apply corner remap
15:  end if
16:  DOFMAP[ $e, i$ ]  $\leftarrow node$ 
17: end for

```

---

### 4.5.2 Internal Force Kernels

The internal force stage evaluates the assembly-free product  $\mathbf{K}\mathbf{u}$ , which is the dominant cost inside the leapfrog loop. In the DOFMAP-based variant, each thread processes one element and scatters local contributions to global DOFs using atomic accumulation, ensuring deterministic updates when neighboring elements share nodes.

The Element-wise variant computes local interactions on-the-fly and avoids dependency on a fully stored global connectivity map. This strategy follows the memory-reduction rationale discussed in Chapter 3 and is particularly useful in large models where indexing storage competes with state vectors and sensor history.

---

**Algorithm 6** productKu\_kernel (DOFMAP-based)

---

**Input:** image, DOFMAP,  $\mathbf{u}$ ,  $\mathbf{k}_e$ ,  $\mathbf{Ku}$ , *poro* (pore identifier)**Output:**  $\mathbf{Ku}$ 

```

1:  $e \leftarrow$  global thread index
2: if  $e \geq n_{elem}$  or image[ $e$ ] = poro then
3:   return
4: end if
5:  $matKey \leftarrow$  image[ $e$ ]
6: for  $i \leftarrow 0$  to 7 do
7:   for  $j \leftarrow 0$  to 7 do
8:     for  $k \leftarrow 0$  to 2 do
9:        $row \leftarrow$  DOFMAP[ $e, i$ ]  $\cdot 3 + k$ 
10:      for  $l \leftarrow 0$  to 2 do
11:         $col \leftarrow$  DOFMAP[ $e, j$ ]  $\cdot 3 + l$ 
12:         $localIdx \leftarrow$  GETLOCALSTIFFNESS( $matKey, i, k, j, l$ )
13:         $\mathbf{Ku}[row] \leftarrow \mathbf{Ku}[row] + \mathbf{k}_e[localIdx] \cdot \mathbf{u}[col]$  (atomic)
14:      end for
15:    end for
16:  end for
17: end for

```

---



---

**Algorithm 7** productKuElemwise\_kernel (Element-wise)

---

**Input:** image, NodesVector,  $\mathbf{u}$ ,  $\mathbf{k}_e$ , *poro* (pore identifier), *dofmap\_type***Output:**  $\mathbf{Ku}$ 

```

1:  $e \leftarrow$  global thread index
2: if  $e \geq n_{elem}$  or image[ $e$ ] = poro then
3:   return
4: end if
5:  $matKey \leftarrow$  image[ $e$ ]
6:  $localdof \leftarrow$  GETLOCALMAP(dofmap_type)  $\triangleright$  Local 8-node map (Direct/Active)
7: for  $i \leftarrow 0$  to 7 do
8:   for  $j \leftarrow 0$  to 7 do
9:     for  $k \leftarrow 0$  to 2 do
10:       $row \leftarrow$   $localdof[i] \cdot 3 + k$ 
11:      for  $l \leftarrow 0$  to 2 do
12:         $col \leftarrow$   $localdof[j] \cdot 3 + l$ 
13:         $localIdx \leftarrow$  GETLOCALSTIFFNESS( $matKey, i, k, j, l$ )
14:         $\mathbf{Ku}[row] \leftarrow \mathbf{Ku}[row] + \mathbf{k}_e[localIdx] \cdot \mathbf{u}[col]$  (atomic)
15:      end for
16:    end for
17:  end for
18: end for

```

---

### 4.5.3 Mass Kernels

This kernel computes the lumped mass vector from element contributions using the DOFMAP connectivity. The operation is equivalent to the diagonal mass construction introduced in Chapter 3 and provides the inverse-mass field required by the explicit time integrator.

---

**Algorithm 8** `lumpedMass_kernel` (DOFMAP-based)

---

**Input:** `image`, `mass`, `DOFMAP`, `poro` (pore identifier)

**Output:** `M`

```

1:  $e \leftarrow$  global thread index
2: if  $e \geq n_{elem}$  or image[ $e$ ] = poro then
3:   return
4: end if
5:  $contrib \leftarrow 0.125 \cdot \text{mass}[\text{image}[e]]$  ▷ Compute node weight (1/8)
6: for  $i \leftarrow 0$  to 7 do
7:    $node \leftarrow \text{DOFMAP}[e, i]$ 
8:    $\mathbf{M}[node] \leftarrow contrib$  (atomic)
9: end for
```

---

This Element-wise mass variant reconstructs node indices dynamically, avoiding explicit global map storage. As in the force kernel, this alternative trades additional indexing arithmetic for reduced global memory.

---

**Algorithm 9** `lumpedMassElemwise_kernel` (Element-wise)

---

**Input:** `image`, `mass`, `NodesVector`, `poro` (pore identifier), `dofmap_type`

**Output:** `M`

```

1:  $e \leftarrow$  global thread index
2: if  $e \geq n_{elem}$  or image[ $e$ ] = poro then
3:   return
4: end if
5:  $localdof \leftarrow \text{GETLOCALMAP}(dofmap\_type)$  ▷ Local 8-node map (Direct/Active)
6:  $contrib \leftarrow 0.125 \cdot \text{mass}[\text{image}[e]]$  ▷ Compute node weight (1/8)
7: for  $i \leftarrow 0$  to 7 do
8:    $node \leftarrow localdof[i]$ 
9:    $\mathbf{M}[node] \leftarrow contrib$  (atomic)
10: end for
```

---

This routine inverts the lumped mass vector in place and preserves zero entries to avoid invalid divisions. The same inversion rule is applied in both CUDA modes so that differences between modes are restricted to data-access strategy, not to numerical formulation.



---

**Algorithm 10** invertMass\_kernel (Mass inversion)

---

**Input:**  $\mathbf{M}$ **Output:**  $\mathbf{M}^{-1}$  in place

```

1:  $i \leftarrow$  global thread index
2: if  $i \geq n_{nodes}$  then
3:   return
4: end if
5: if  $\mathbf{M}[i] = 0$  then
6:    $\mathbf{M}[i] \leftarrow 0$ 
7: else
8:    $\mathbf{M}[i] \leftarrow 1/\mathbf{M}[i]$ 
9: end if

```

---

### 4.5.4 Time Marching Kernels

This displacement update corresponds to the explicit position step of the leapfrog integrator. Together with velocity update and source imposition, it forms the same time marching structure described in Chapter 3, where stability and accuracy are controlled through the chosen  $\Delta t$  and  $\Delta L$  (CFL and dispersion criteria).

---

**Algorithm 11** updateDisplacement\_kernel (displacement update)

---

**Input:**  $\mathbf{u}$ ,  $\mathbf{v}$ ,  $\Delta t$ **Output:** Updated  $\mathbf{u}$ 

```

1:  $i \leftarrow$  global thread index
2: if  $i \geq n_{dof}$  then
3:   return
4: end if
5:  $\mathbf{u}[i] \leftarrow \mathbf{u}[i] + \mathbf{v}[i]\Delta t$ 

```

---

This velocity update applies internal force acceleration weighted by inverse mass and time step. In practice, this kernel advances momentum using the current  $\mathbf{Ku}$  result and keeps the transient evolution consistent with the elastodynamic formulation used throughout the thesis.

---

**Algorithm 12** updateVelocity\_kernel (velocity update)

---

**Input:**  $\mathbf{v}$ ,  $\mathbf{M}^{-1}$ ,  $\mathbf{Ku}$ ,  $\Delta t$ **Output:** Updated  $\mathbf{v}$ 

```

1:  $i \leftarrow$  global thread index
2: if  $i \geq n_{dof}$  then
3:   return
4: end if
5:  $node \leftarrow \lfloor i/3 \rfloor$ 
6:  $\mathbf{v}[i] \leftarrow \mathbf{v}[i] - \mathbf{M}^{-1}[node] \cdot \mathbf{Ku}[i] \cdot \Delta t$ 

```

---

### 4.5.5 History Kernels

This kernel stores full sensor traces at each time index, enabling waveform level comparisons between solver modes. These histories are the primary observables for arrival-time analysis and cross-validation, as explored in the experimental chapter.

---

**Algorithm 13** `history_kernel` (Sensor history recording)

---

**Input:** `sensor1`, `sensor2`, `u`, `t`

**Output:** `S1`, `S2`

```

1:  $i \leftarrow$  global thread index
2: if  $i \geq n_{\text{sensor}}$  then
3:   return
4: end if
5:  $S_1[i, t] \leftarrow u[\text{sensor}_1[i]]$ 
6:  $S_2[i, t] \leftarrow u[\text{sensor}_2[i]]$ 

```

---

This reduced-history variant records one averaged value per sensor face and time step, substantially reducing output storage. It is the preferred option for large-scale studies where the objective is effective-velocity estimation from representative sensor curves rather than full face-resolution diagnostics.

---

**Algorithm 14** `history_average_kernel` (Average sensor history recording)

---

**Input:** `sensor1`, `sensor2`, `u`, `t`

**Output:** `S1`, `S2`

```

1: Initialize  $sum_1, sum_2 \leftarrow 0$ 
2: for  $i \leftarrow 0$  to  $n_{\text{facenodes}} - 1$  do
3:    $sum_1 += u[\text{sensor}_1[i]]$ 
4:    $sum_2 += u[\text{sensor}_2[i]]$ 
5: end for
6:  $S_1[t] \leftarrow sum_1 / n_{\text{facenodes}}$ 
7:  $S_2[t] \leftarrow sum_2 / n_{\text{facenodes}}$ 

```

---

### 4.5.6 Pulse Kernels

This pulse kernel imposes the prescribed Dirichlet excitation directly on the selected pulse DOFs. The injected signal follows the UPT modeling assumptions adopted in this thesis, allowing consistent comparison between numerical travel times and the effective-velocity interpretation discussed since Chapter 2.

---

**Algorithm 15** `applyPulse_kernel` (Dirichlet pulse injection at time  $t$ )

---

**Input:**  $\mathbf{u}_g$ ,  $\mathbf{u}$ ,  $timeIndex$ ,  $pulse$ **Output:** Updated  $\mathbf{u}$ 

```

1:  $pv \leftarrow$  global thread index
2: if  $pv \geq n_{pulse}$  then
3:   return
4: end if
5:  $\mathbf{u}[pulse[pv]] \leftarrow \mathbf{u}_g[timeIndex]$ 

```

---

### 4.5.7 Utility Kernels

This utility kernel clears device vectors between solver stages with minimal overhead. Although simple, this operation is important to avoid contamination between consecutive updates and to preserve reproducibility in long benchmark batches.

---

**Algorithm 16** `zero_vector_kernel` (vector reset)

---

**Input:**  $n_{size}$ ,  $\mathbf{vector}$ **Output:** Zeroed  $\mathbf{vector}$ 

```

1:  $i \leftarrow$  global thread index
2: if  $i < n_{size}$  then
3:    $\mathbf{vector}[i] \leftarrow 0$ 
4: end if

```

---

## 4.6 Summary of the Implemented Solvers

The current implementation provides **five solver modes**, selected by the runtime parameter `solver_type` (values 1 to 5, Table 6). Although the mathematical model is the same in all cases (explicit leapfrog integration for elastodynamic wave propagation), the computational strategy changes according to the target architecture and data-access pattern.

The main distinctions between modes are: (i) where the dominant operations are executed (CPU or GPU), (ii) how  $\mathbf{Ku}$  is computed (DOFMAP based or Element-wise), and (iii) how DOFs is mapped (Direct Node Mapping or Active Node Mapping).

### Mode 1: Serial CPU Solver

This is the reference implementation for validation and debugging. It builds a global DOF mapping, computes lumped mass, locates sensors, injects the pulse, and advances the

solution with the standard leapfrog routine. The matrix-vector action  $\mathbf{Ku}$  is evaluated through the DOF map, which provides a direct and transparent baseline for correctness checks.

### **Mode 2: OpenMP CPU Solver**

Mode 2 keeps the same numerical formulation as mode 1, but parallelizes the dominant loops (DOF map generation, mass assembly, pulse handling, history extraction, and leapfrog updates) using OpenMP. Therefore, differences relative to mode 1 are expected primarily in performance, while numerical outputs should remain consistent within floating-point tolerance.

### **Mode 3: CUDA GPU Solver**

In this mode, expensive stages are executed on the GPU. The implementation copies model constants to device memory, generates DOF mapping and sensor indexing on device, computes and inverts lumped mass on device, and performs the full leapfrog loop with GPU kernels. The design explicitly minimizes CPU–GPU transfers, returning mainly sensor histories (and optional snapshots) to the host.

### **Mode 4: CUDA GPU Element-wise Solver**

Mode 4 follows the same GPU-oriented philosophy as mode 3, but adopts an element-wise strategy to compute internal forces. This avoids storing a full global DOF map, which can reduce memory pressure for large models or for configurations where element-local access is advantageous. Sensor indexing is also handled with element-wise kernels.

### **Mode 5: Serial CPU Element-wise Solver**

Mode 5 is the CPU counterpart of the element-wise approach. It is useful as a deterministic reference when analyzing the behavior of element-wise kernels in mode 4, and as an alternative when GPU resources are unavailable. The formulation remains explicit leapfrog, but  $\mathbf{Ku}$  is assembled from element contributions without relying on the global DOF-map product used in modes 1 and 2.

In addition to the `solver_type`, three other settings affect all modes in terms of performance and memory:

Table 6: Implemented solver modes in the current implementation.

| ID | Name                         | Execution model                       | Main characteristic              |
|----|------------------------------|---------------------------------------|----------------------------------|
| 1  | Serial CPU                   | Single-thread C                       | Baseline DOFMAP implementation   |
| 2  | OpenMP CPU                   | Multi-thread shared-memory            | Parallel CPU version of mode 1   |
| 3  | CUDA GPU                     | GPU kernels + minimal host transfers  | DOFMAP generated and used on GPU |
| 4  | CUDA GPU<br>(Element-wise)   | GPU kernels + Element-wise operations | Avoids global DOFMAP storage     |
| 5  | Serial CPU<br>(Element-wise) | Single-thread C                       | CPU Element-wise reference       |

- **DOF mapping type** (Direct Node Mapping or Active Node Mapping), which changes indexing and memory layout.
- **Sensor configuration**, which defines how receiver data are sampled (single representative signal or full face acquisition) and directly impacts sensor array size, output volume, and post-processing cost.
- **History (snapshot) strategy**, which controls full-field snapshot storage at selected time steps and directly affects output size, I/O overhead, and post-processing cost.

In summary, the implemented set of solvers forms a coherent progression: a serial baseline (mode 1), CPU shared-memory acceleration (mode 2), full GPU acceleration (mode 3), and two Element-wise variants (modes 4 and 5) aimed at alternative memory-access and storage trade-offs. This structure supports both reproducibility studies and performance-oriented exploration on large-scale digital rocks images.

## 5 Numerical Results

This Section presents the numerical results of the implementation solver in a well-known model frequently employed in the literature and digital rocks. The first model comprises a simple homogeneous rod fixed at one end subjected to a Heaviside load. This model's analytical response enables a comparison with the numerical responses obtained from both 2D and 3D solvers. Subsequently, simulations were performed using the implemented solver on real images of samples acquired through  $\mu$ CT. The effective velocities calculation analysis involved three digital rocks: Berea Brown (BB), Berea Gray (BG), and Crab Orchard (CO). Furthermore, the effective velocities  $V_P$  and  $V_S$  obtained using the dynamic solver were compared with the static results acquired through the program [19]. Finally, the implementation is validated against the digital rock benchmark of Andr  et al. [14], with analyses for Fontainebleau, Berea, and Grosmont samples.

All numerical simulations reported in this chapter were executed on a single workstation. Table 7 summarizes the hardware and software configuration of the machine used throughout the experiments. The CPU was responsible for mesh generation, pre-processing, and I/O operations, while all time-stepping kernels were dispatched to the GPU using the CUDA programming model.

Table 7: Hardware and software configuration of the workstation used in the numerical experiments.

| Component           | Specification                              |
|---------------------|--------------------------------------------|
| Processor (CPU)     | Intel Xeon W-2295 (36-core, up to 4.6 GHz) |
| System memory       | 512 GB DDR4                                |
| Graphics card (GPU) | NVIDIA GeForce RTX 3080                    |
| GPU memory (VRAM)   | 12 GB GDDR6X                               |
| CUDA version        | 12.8                                       |
| Operating system    | Linux (Ubuntu 20.04)                       |
| Compiler            | GCC 9.4.0 / NVCC 12.2                      |

## 5.1 Finite Homogeneous Rod

To validate the implementations and accuracy of 2D and 3D FEMs in the time domain using Leap-Frog time integration, we chose a well-known model frequently employed in the literature for dynamic problems [101]. The model consists of a simple homogeneous rod fixed at one end subjected to a Heaviside load, as illustrated in Figure 20. This model has an analytical response that allows for a comparison with the numerical responses obtained from both 2D and 3D solvers.

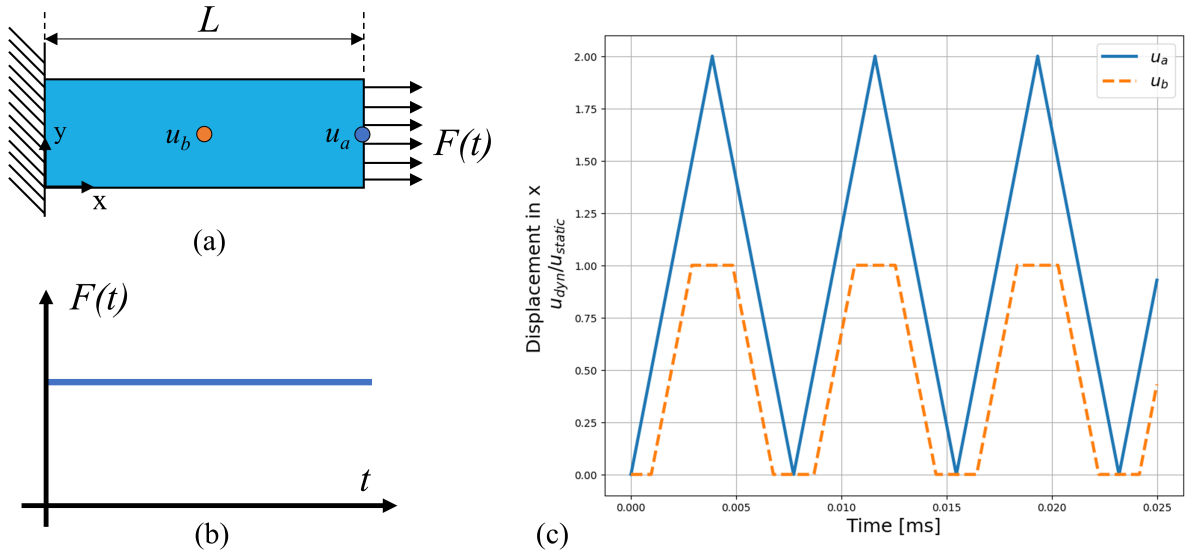


Figure 20: Wave propagation in a one-dimensional rod: (a) model, geometry, and forces; (b) Heaviside loading pattern; and (c) displacement ratio at end-node and mid-span.

In this section, the model's dimensions and properties adhere to the specifications

outlined in the work by [102]. Specifically, the material properties include a Young's modulus of  $E = 210$  GPa, a Poisson's ratio of  $\nu = 0$ , a mass density of  $\rho = 7850$  kg/m<sup>3</sup>, and  $V_P = 5172.19$  m/s. The square rod under consideration has a length of  $L = 10$  m and a width of 1 m, resulting in a cross-sectional area  $A = 1$  m<sup>2</sup>, subjected to a Heaviside load of  $P = 21$  kN. The load is defined as

$$F(t) = PH(t) = \begin{cases} P, & t \geq 0 \\ 0, & t < 0. \end{cases} \quad (5.1)$$

Although  $\nu = 0$  is not representative of a real engineering material, this value was adopted because the analytical reference used in this validation solves the one-dimensional wave equation, which contains no Poisson (lateral-contraction) term. The FEM model evaluated here is a general 2D/3D solid that would naturally develop lateral displacements for  $\nu \neq 0$ , an effect with no counterpart in the 1D analytical solution. Setting  $\nu = 0$ , as in the benchmark configuration of [102], avoids this mismatch, so that the errors reported in the convergence studies of Sections 5.1.1 and 5.1.2 reflect only the spatial and temporal discretization error under investigation, and not a geometric inconsistency between the 2D/3D model and the 1D analytical solution.

In Figure 20(c), the displacement curve over time at both the end  $u_a$  and the midpoint  $u_b$  of the bar is displayed. The displacement is represented as the ratio of the dynamic displacement  $u_{dyn}$  to the static displacement  $u_{static}$  at the end of the bar in the static model, calculated by,

$$u_{static} = \frac{PL}{EA}. \quad (5.2)$$

To validate the numerical implementation of the Leap-Frog time integration scheme and to determine optimal discretization parameters for the subsequent digital rock simulations, two comprehensive convergence studies are conducted on this homogeneous rod model. The first study analyzes the convergence with respect to spatial discretization ( $\Delta L$ ), while the second examines temporal discretization ( $\Delta t$ ). These convergence tests are essential for establishing the relationship between mesh refinement, time step size, and solution accuracy, as well as for identifying the optimal balance between computational cost and numerical precision.



### 5.1.1 Convergence of Space Discretization

The primary objective of the initial tests is to observe the behavior of the curve and the approximation of displacements as the model becomes increasingly refined. To investigate the convergence of space discretization, all tested models share the same length  $L$  but vary in mesh discretization (spatial), according to Table 8. These simulations maintain fixed values for  $\Delta t$ ,  $T_f$ , and the total number of time steps  $N_t$ . Given that each simulation features a specific  $\Delta L$  value, the respective CFL values were calculated using Equation 3.26.

Table 8: Adjustable mesh discretization with a constant time step  $\Delta t$ .

| Tests   | Mesh                     | $\Delta L$ [m] | $T_f$ [ $10^{-3}$ s] | $\Delta t$ [ $10^{-6}$ s] | $N_t$ | $V_P$ [m/s] | CFL    |
|---------|--------------------------|----------------|----------------------|---------------------------|-------|-------------|--------|
| Model 1 | $5 \times 5 \times 5$    | 2.00           | 25.0                 | 1.80                      | 13888 | 5172.19     | 0.0047 |
| Model 2 | $10 \times 5 \times 5$   | 1.00           | 25.0                 | 1.80                      | 13888 | 5172.19     | 0.0093 |
| Model 3 | $50 \times 5 \times 5$   | 0.20           | 25.0                 | 1.80                      | 13888 | 5172.19     | 0.0465 |
| Model 4 | $100 \times 5 \times 5$  | 0.10           | 25.0                 | 1.80                      | 13888 | 5172.19     | 0.0931 |
| Model 5 | $500 \times 5 \times 5$  | 0.20           | 25.0                 | 1.80                      | 13888 | 5172.19     | 0.4655 |
| Model 6 | $1000 \times 5 \times 5$ | 0.01           | 25.0                 | 1.80                      | 13888 | 5172.19     | 0.9310 |

It can be observed that most of the curves in Figure 21 are in good agreement with the analytic solution. The model with the lowest level of discretization produced the worst results, but this was expected since the lower level of discretization introduces larger numerical errors and approximations, leading to decreased accuracy in capturing the underlying behavior of the system. After analyzing the displacement results, it becomes evident that higher discretization leads to outcomes that closely align with the analytical solution.

A close examination of the curves in Figure 22 during the time interval of 0.019 to 0.020 ms, one can observe that even with a favorable  $\Delta t$ , models with  $n_x = 5$  and  $n_x = 10$  are more sensitive to the choice of lower discretization. Additionally, in Figure 23, on the mid-span, we can observe that lower discretization choices also exhibit sensitivity in displacement results. On the other hand, some models perform well with an equivalent number of time steps, but with varying processing times. It is important to note that larger models with increased degrees of freedom require higher memory consumption and more processing power. This highlights the trade-off between precision and computational

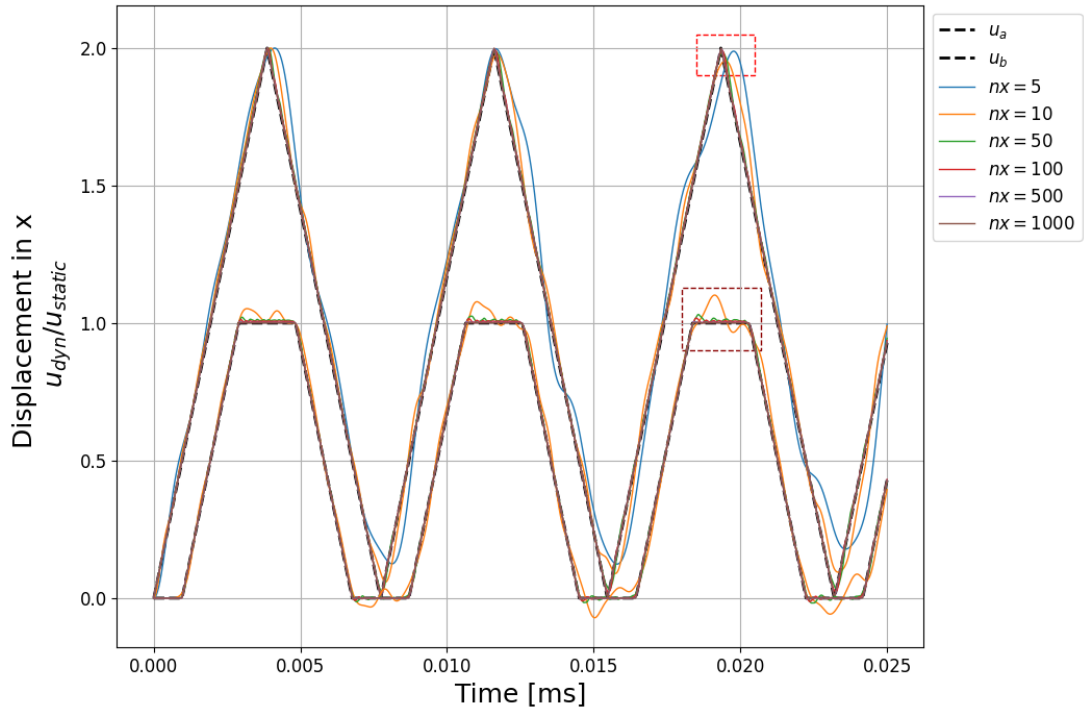


Figure 21: Numerical normalized displacement at the end node and mid-span in the finite homogeneous rod models with varying space discretization.

resources in the context of mesh grid variations and fixed-time step configurations.

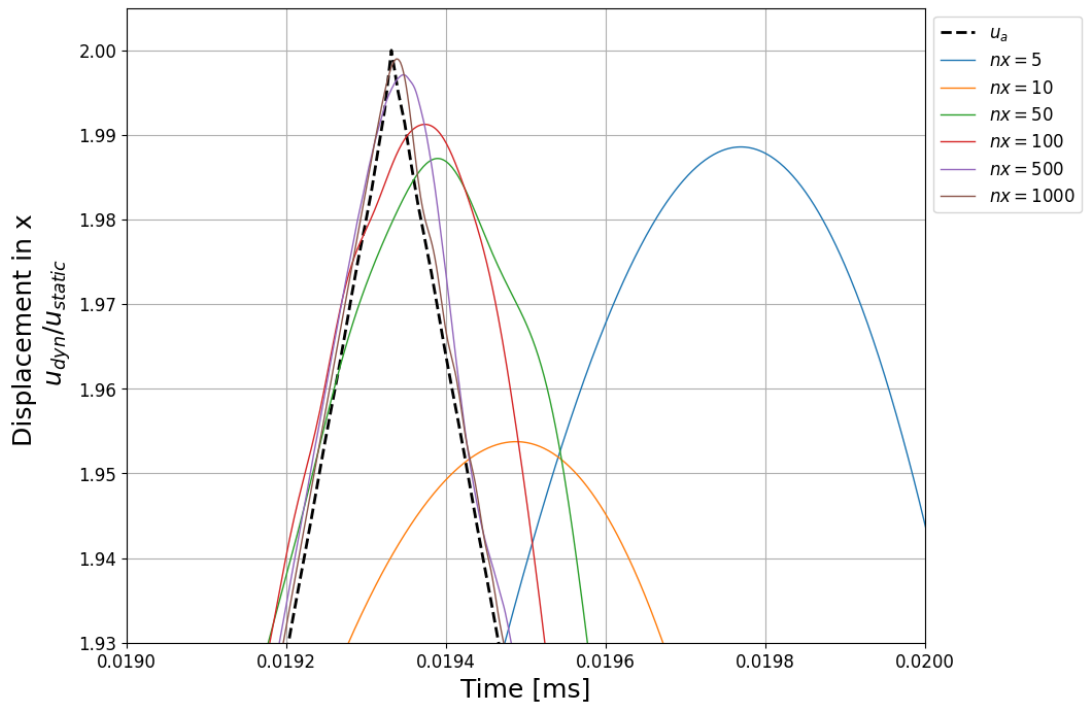


Figure 22: Zoomed of end node displacements and analytical solution in models with varying space discretization.

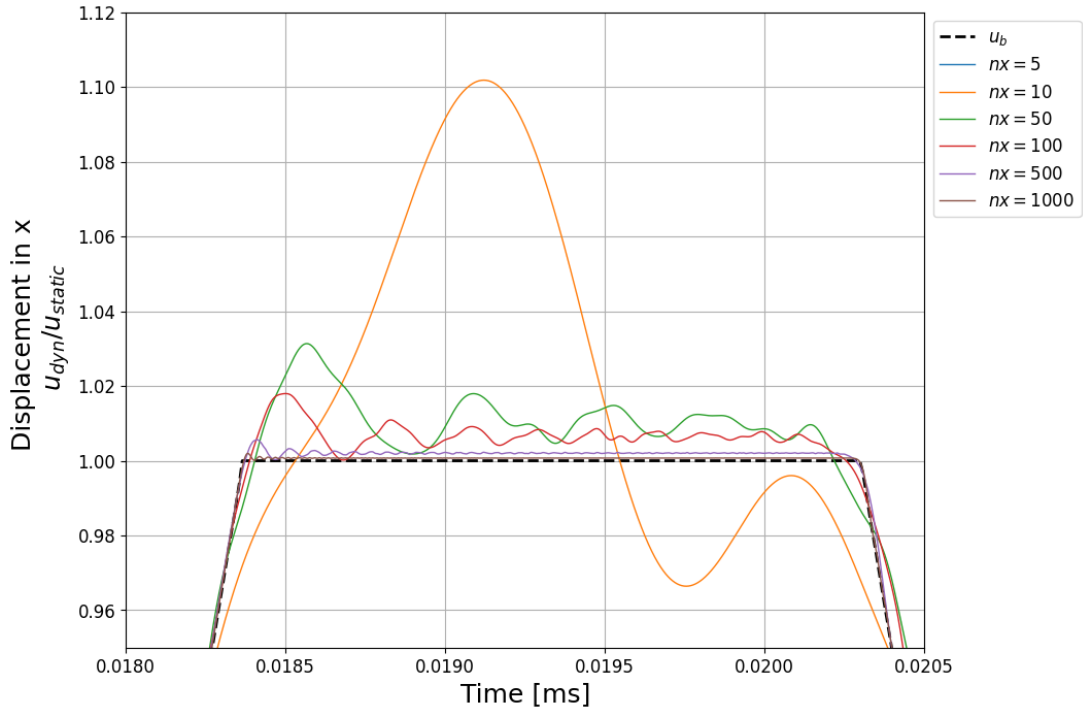


Figure 23: Zoomed of mid-span displacements and analytical solution in models with varying space discretization.

The relative error is commonly calculated by comparing the difference between the single predicted value and the analytical value. However, to compare the model across all time intervals, we employed the overall relative error  $\epsilon$ . This metric offers a more robust evaluation by encompassing the entire dataset, rather than focusing on isolated points at a specific time. Consequently, analytical displacement results were generated for each model for all time intervals, adhering to the specified mesh discretization. The mathematical formulation of the overall relative error for this problem is as follows

$$\epsilon = \frac{\int_0^L \int_0^{T_f} |d - u| dt dx}{\int_0^L \int_0^{T_f} u dt dx}, \quad (5.3)$$

where  $u$  is the analytical result and  $d$  is the numerical result.

Upon further examination of the curve in Figure 24, it becomes clear that increased spatial discretization correlates with the lowest overall relative error, indicating a higher level of precision in the results. On the other hand, the CFL factor, a dimensionless parameter linked to numerical stability, could offer a potential understanding of the model's behavior under the convergence of spatial discretization. However, within the context of these particular tests, relying solely on the CFL may not be adequate to determine the optimal analysis. This limitation arises from the trade-off between precision and

computational resources.

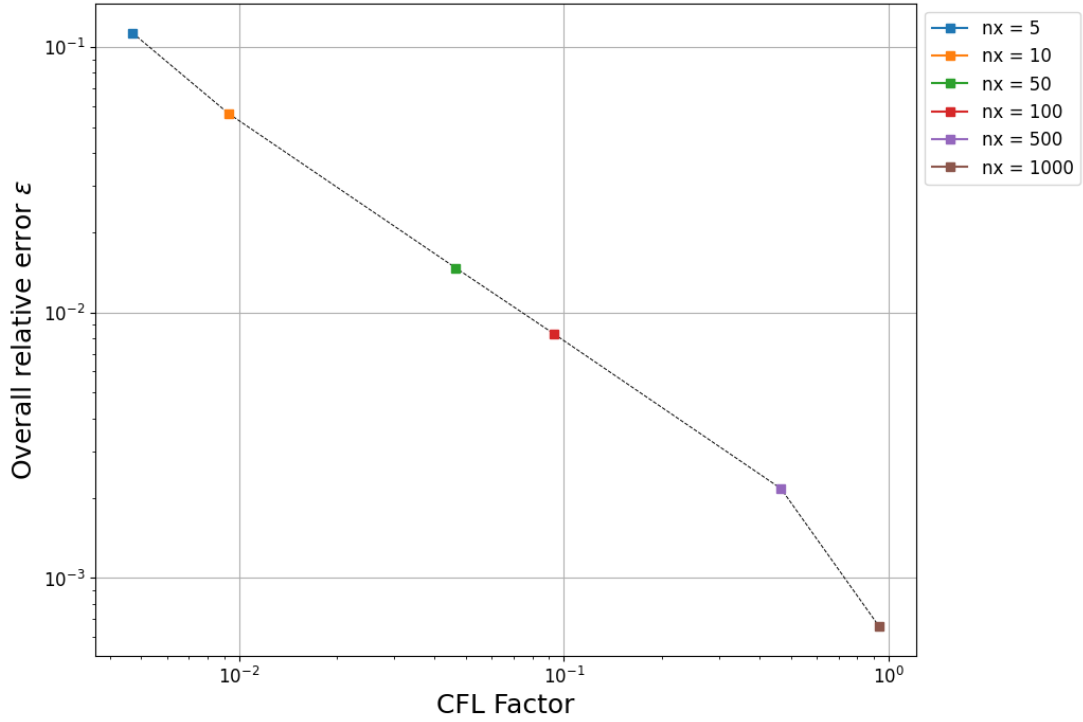


Figure 24: Overall relative error as a function of CFL factor for refined meshes.

### 5.1.2 Convergence of Time Discretization

To assess the convergence of time discretization, six simulations were conducted using a fixed mesh of  $100 \times 10 \times 10$  and a constant  $T_f$ . In these simulations, the varying parameter was  $\Delta t$ . Additionally, CFL factors were calculated based on Equation 3.26, and the corresponding simulation data is provided in Table 9.

Table 9: Adjustable time step  $\Delta t$  with a constant mesh discretization.

| Tests    | Mesh                      | $\Delta L$ [m] | $T_f$ [ $10^{-3}$ s] | $\Delta t$ [ $10^{-6}$ s] | $N_t$ | $V_P$ [m/s] | CFL    |
|----------|---------------------------|----------------|----------------------|---------------------------|-------|-------------|--------|
| Model 7  | $100 \times 10 \times 10$ | 0.10           | 25.0                 | 19.2                      | 1300  | 5172.19     | 0.9947 |
| Model 8  | $100 \times 10 \times 10$ | 0.10           | 25.0                 | 16.7                      | 1500  | 5172.19     | 0.8620 |
| Model 9  | $100 \times 10 \times 10$ | 0.10           | 25.0                 | 12.5                      | 2000  | 5172.19     | 0.6465 |
| Model 10 | $100 \times 10 \times 10$ | 0.10           | 25.0                 | 8.33                      | 3000  | 5172.19     | 0.4310 |
| Model 11 | $100 \times 10 \times 10$ | 0.10           | 25.0                 | 5.00                      | 5000  | 5172.19     | 0.2586 |
| Model 12 | $100 \times 10 \times 10$ | 0.10           | 25.0                 | 2.50                      | 10000 | 5172.19     | 0.1293 |

The chosen discretization for the time discretization tests exhibits a strong correspondence with the analytical results, as demonstrated in Subsection 5.1.1. This deliberate selection has yielded results that are visually indistinguishable from the analytical outcomes across the various models examined with different values of  $\Delta t$ . Specifically, when examining the curves presented in Figure 25, the distinction between the analytical and numerical outcomes is not readily apparent, whether considering the end node or mid-span. This observation already suggests that, for this particular discretization, the choice of the time step  $\Delta t$  holds less significance to the precision in approximating the results.

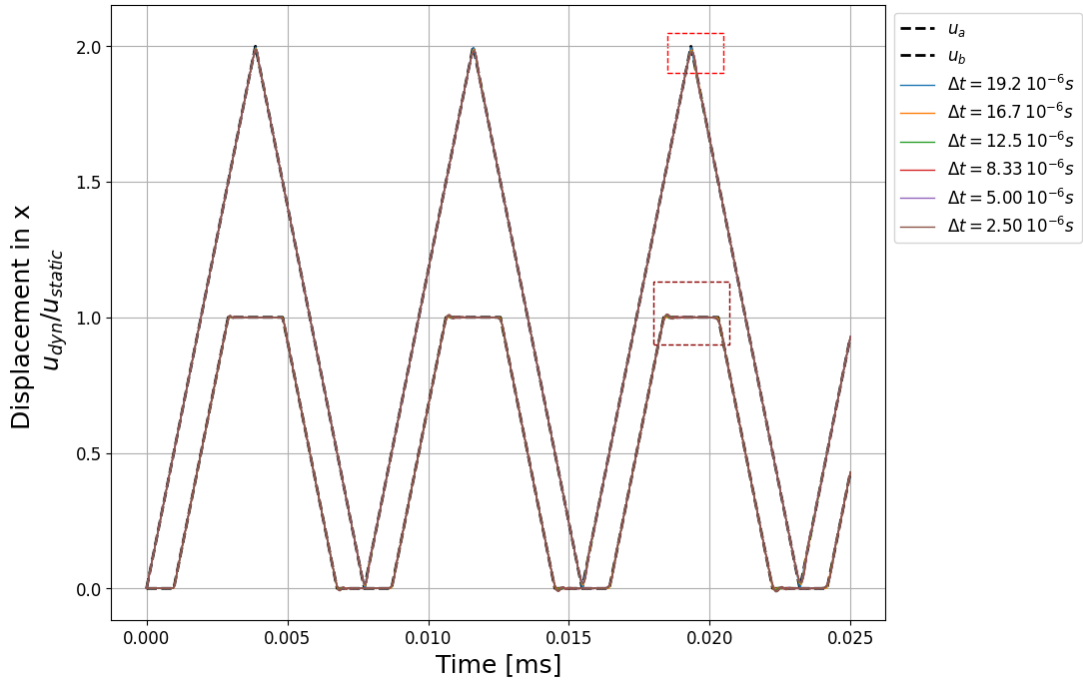


Figure 25: Numerical normalized displacement at the end node and mid-span in the finite homogeneous rod models with discretization.

A detailed inspection is warranted for the curves in Figures 26 and 27. Within both curves, it becomes apparent that reducing  $\Delta t$  does not necessarily result in increased accuracy. Instead, the reduction in  $\Delta t$  within this discretization appears to yield less precise and computationally time-consuming results.

To validate the difference between the numerical and analytical outcomes in models with varying time discretization, an error map was generated (Figure 28). This visualization employs a color map to represent the relative error between the analytical solution  $u$  and the numerical prediction  $d$  at each position in both time and space. All graphs share a uniform color scale, and the end node is positioned at  $x = 10.0$ . As the mesh discretization remains consistent across all models, the variation in CFL occurs exclusively due to changes in  $\Delta t$ . Furthermore, in the context of this discretization, the error map reveals a decrease

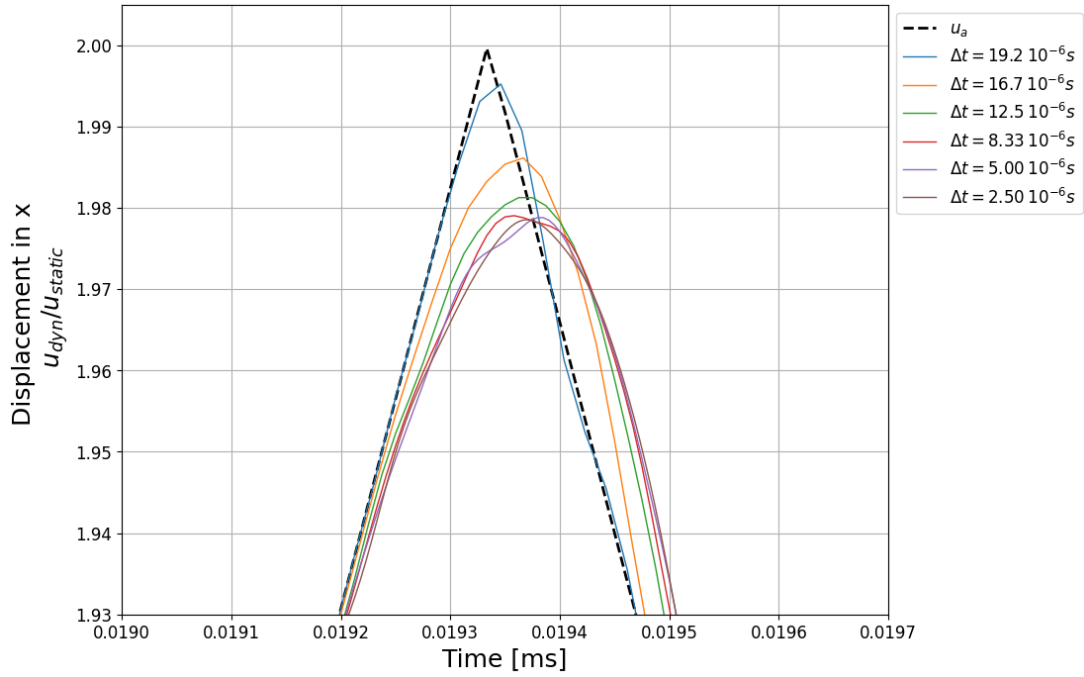


Figure 26: Zoomed of end node displacements and analytical solution in models with varying time discretization.

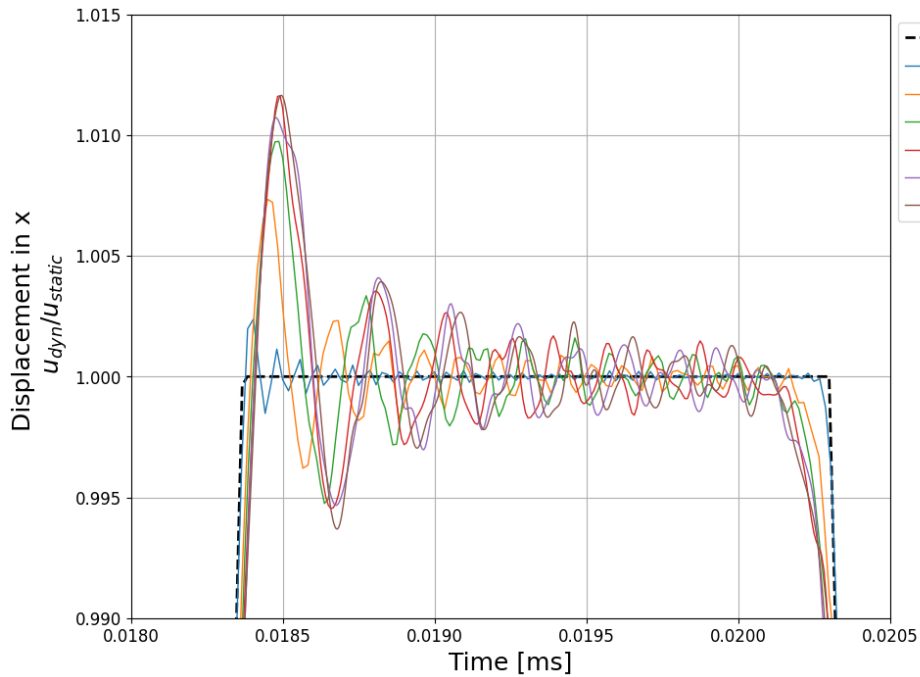


Figure 27: Zoomed of mid-span displacements and analytical solution in models with varying time discretization.

in numerical accuracy with a reduction in  $\Delta t$ .

In conclusion, two curves can contribute to drawing insights for models with different time discretization. These curves represent the overall relative error (Figure 29) and

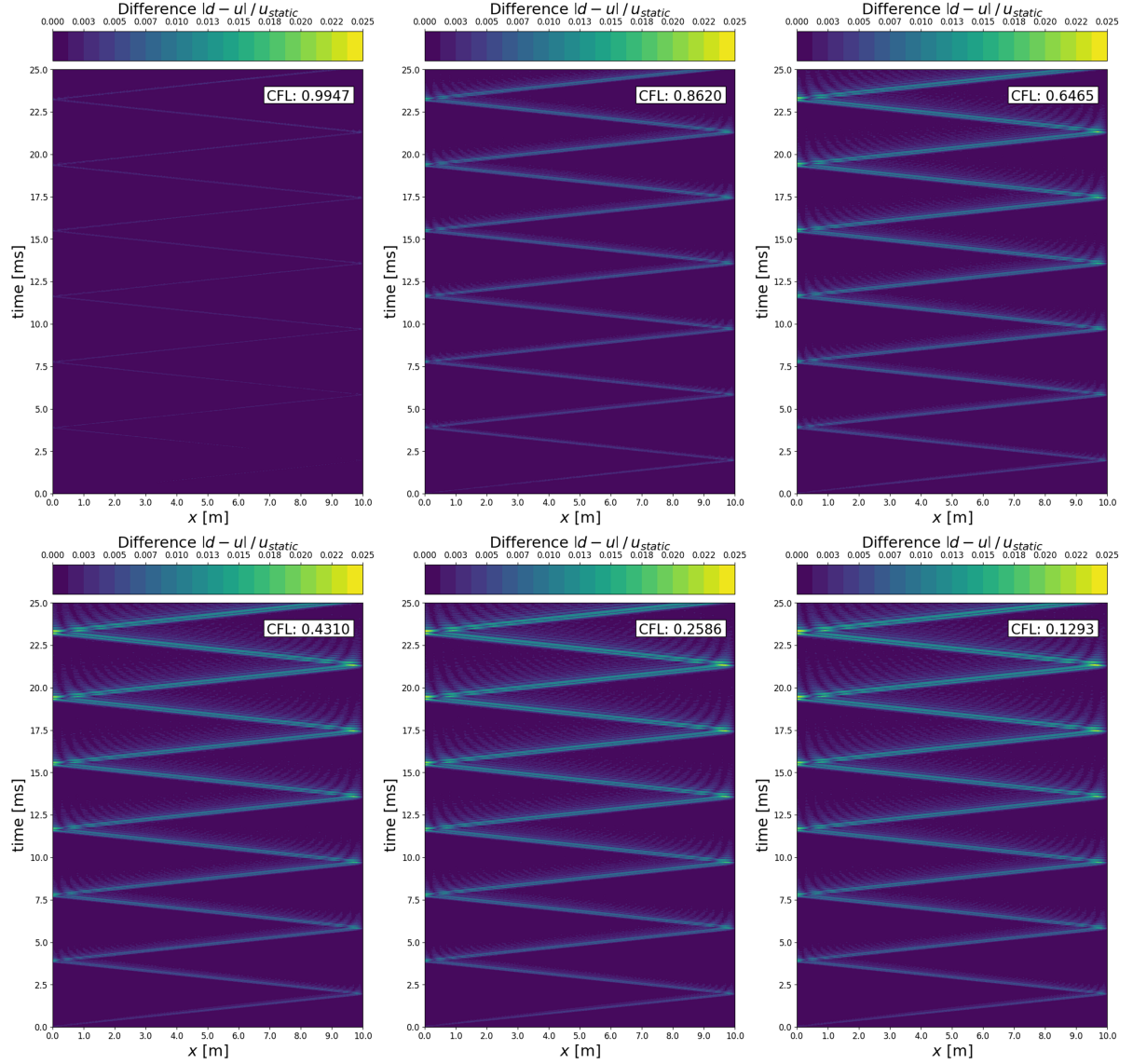


Figure 28: Error map for rod simulation using a  $100 \times 10 \times 10$  mesh, while varying the CFL factor through changes in time step  $\Delta t$ .

average time per simulation (Figure 30), both to the CFL factor. Analyzing these curves, one can infer that, for these models, the most efficient configurations exhibit a CFL factor ranging between 0.80 and 1.00, striking an optimal balance between processing time and simulation accuracy.

This behavior, in which accuracy improves as the CFL number approaches its stability limit rather than as  $\Delta t$  is decreased, is a known property of explicit leapfrog schemes for the wave equation, sometimes called a “magic time step” in the literature [103]. A centered, second-order time-marching scheme does not reproduce the exact phase velocity: it introduces its own dispersion error, which depends on the CFL number and adds to the spatial dispersion error discussed in Section 3.3.2 [76, 75]. Since these two errors have

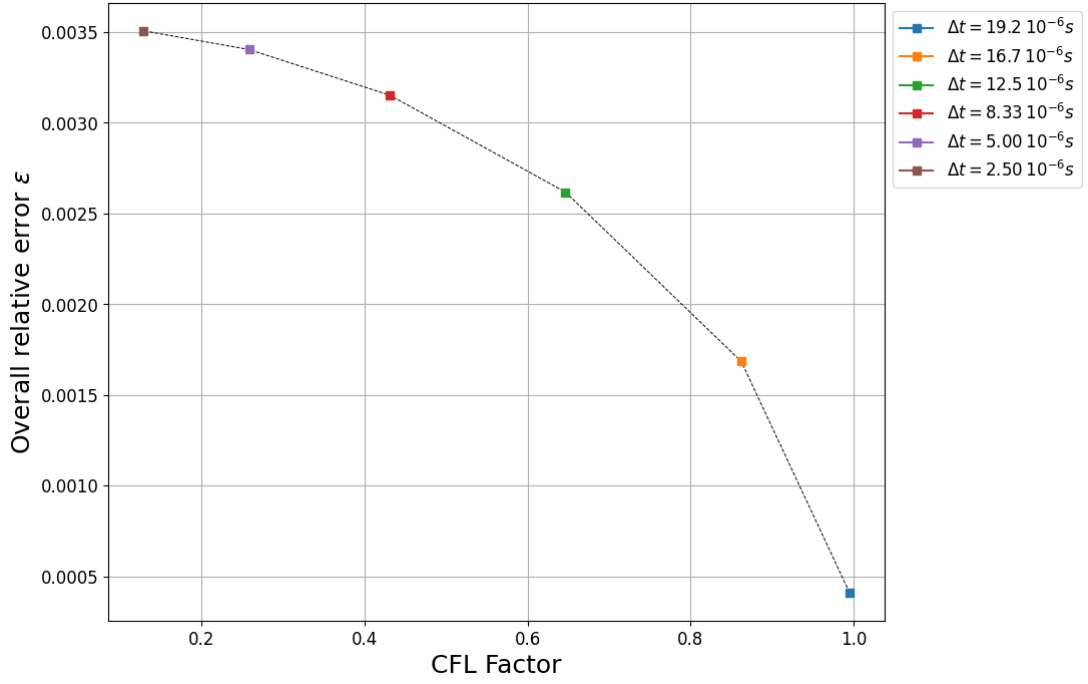


Figure 29: Overall relative error as a function of CFL factor for models with varying time discretization.

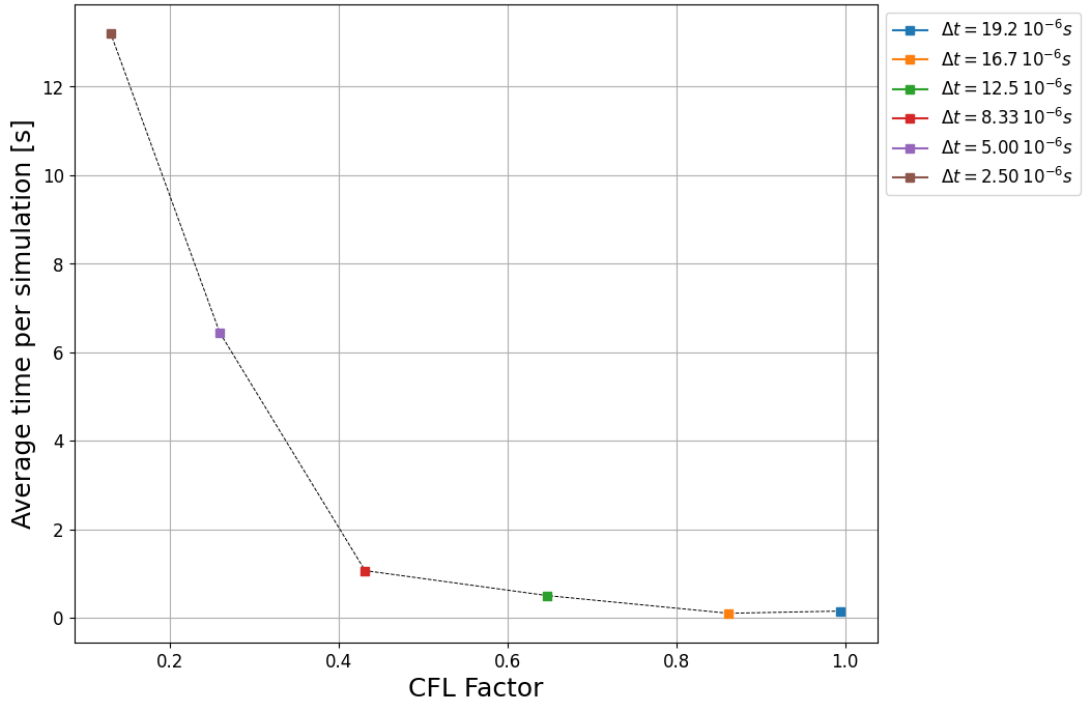


Figure 30: Average processing time per simulation with varying time discretization.

opposite signs, a  $\Delta t$  close to the critical value  $\Delta t_{cr}$  makes them partially cancel out, so the total error is smallest near the stability limit rather than at small time steps [104]. As the CFL number decreases, this cancellation weakens and the spatial dispersion error



remains largely uncompensated.

## 5.2 MicroCT 2D Images

Following the benchmark verification, simulations were conducted using the implemented solver on real images of samples acquired through  $\mu$ CT. The effective velocities  $V_P$  calculation analysis utilized three digital rocks (Figure 31): Berea Brown (BB), Berea Gray (BG), and Crab Orchard (CO). Additionally, the effective velocities  $V_P$  and  $V_S$  obtained using the dynamic solver were compared with the static results acquired through the CHFEM solver [19]. For both dynamic and static simulations, the solid phase represented by white pixels will be considered as quartz. The elastic properties attributed to quartz include  $V_P = 6004.30$  m/s,  $V_S = 4082.77$  m/s, and  $\rho = 2650$  kg/m<sup>3</sup> [2]. However, in the static simulation, due to solver limitations, the porous part represented by black pixels will be treated as a material with distinct properties:  $V_P = 0.028$  m/s,  $V_S = 0.000$  m/s, and  $\rho = 2650$  kg/m<sup>3</sup>. It is important to note that additional rock characteristics can be referenced in Table 10.

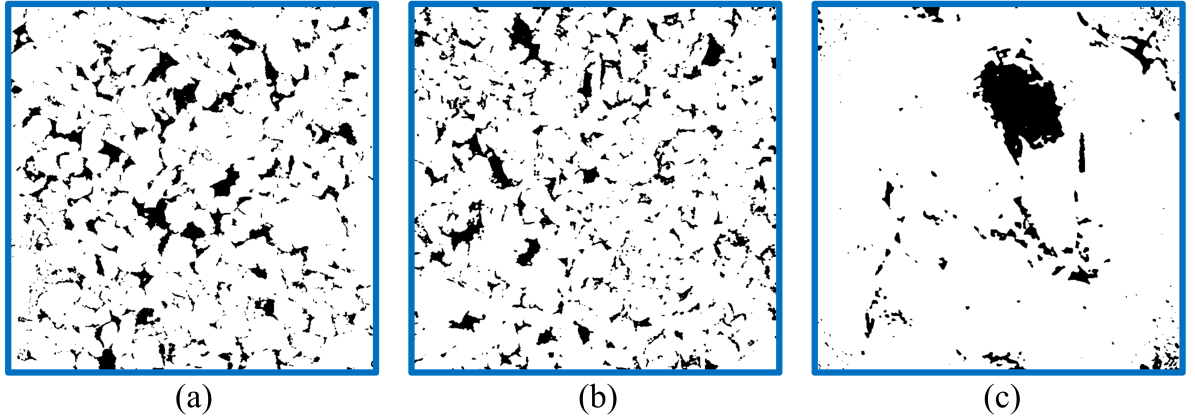


Figure 31: Binary two-dimensional slices of the digital rock dataset with pore space in black, and solid phase in white: (a) Berea Brown (BB), (b) Berea Gray (BG), and (c) Crab Orchard (CO).

Table 10: Characteristics of investigated rocks samples.

| Digital Core | Rock ID. | Type      | Mesh    | Porosity |
|--------------|----------|-----------|---------|----------|
| Berea Brown  | BB       | sandstone | 600×600 | 12.19 %  |
| Berea Gray   | BG       | sandstone | 600×600 | 11.77 %  |
| Crab Orchard | CO       | sandstone | 632×660 | 7.60 %   |

### 5.2.1 Gaussian Pulse and Buffer Zone

Laboratory experiments have long established the calibration dependency of the transmitted pulse for measuring effective velocities  $V_P$  and  $V_S$ . Additionally, the work by [105] emphasizes that, for numerical experiments, the choice of boundary conditions influences the calculation of effective velocities. In light of this, an initial set of 45 simulations was conducted, wherein both the buffer zone and the transmitted Gaussian pulse were varied to determine the optimal boundary conditions. Buffer zone values ranged from 50 to 400 pixels, and 9 Gaussian pulses, with frequencies spanning from 0.57 MHz to 49.71 MHz, were generated.

From the array of simulations, those employing buffer zones of 200 and 400 pixels were chosen to illustrate the observed outcomes. Figure 32 showcases the Gaussian pulses used in the simulation with a 200-pixel buffer zone,  $\Delta t = 3.182\text{e-}9$  s,  $\Delta L = 2.000\text{e-}5$  m, and  $\text{CFL} = 0.9552$ . Furthermore, Figure 33 presents the Gaussian pulses used in simulations with a 400-pixel buffer zone,  $\Delta t = 2.500\text{e-}9$  s,  $\Delta L = 2.000\text{e-}5$  m, and  $\text{CFL} = 0.7505$ .

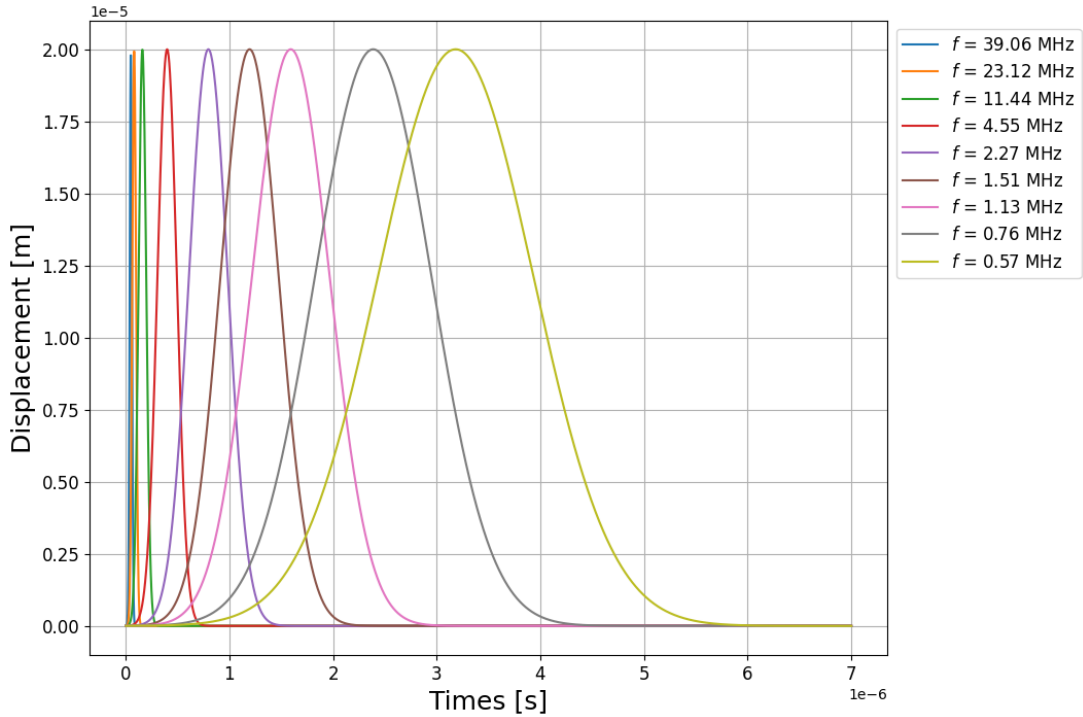


Figure 32: The Gaussian pulses and frequencies used in the model with the 200-pixel buffer zone, time step size  $\Delta t = 3.182\text{e-}9$  s, and spatial discretization size  $\Delta L = 2.00\text{e-}5$  m.

As demonstrated in Section 3.5, the calculation of the effective velocity relies on two sensors recording the average displacement throughout the UPT experiment. Figures 34 and 35 illustrate the measured displacements at sensors S1 and S2, respectively, for models

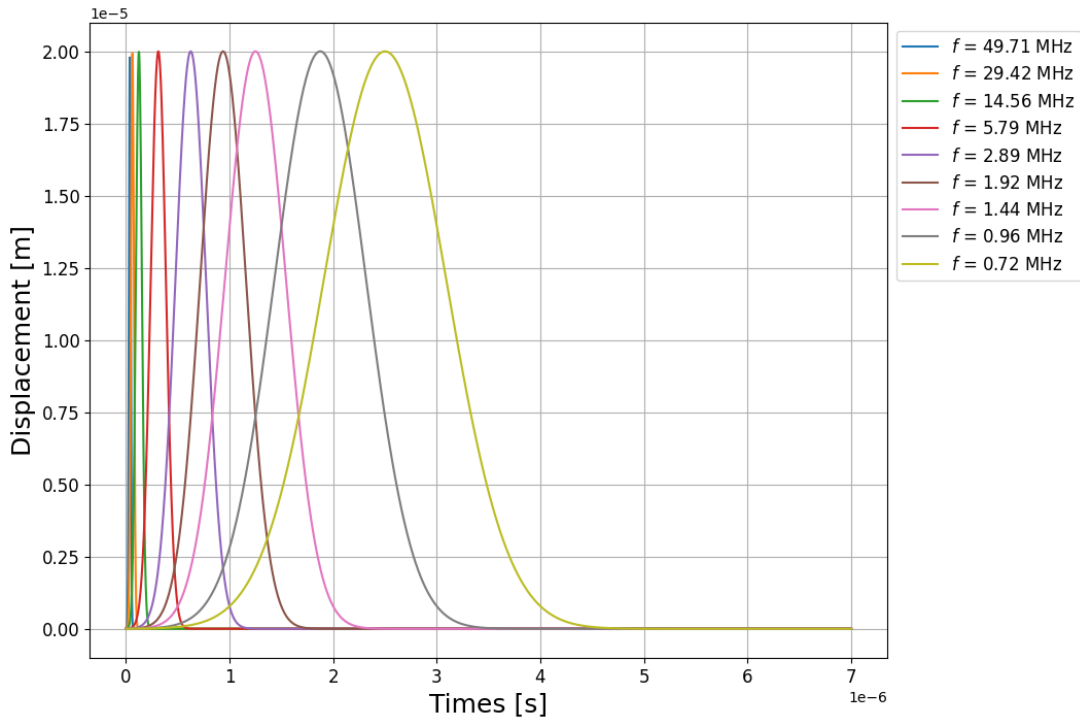


Figure 33: The Gaussian pulses and frequencies used in the model with the 400-pixel buffer zone, time step size  $\Delta t = 2.500\text{e-}9$  s, and spatial discretization size  $\Delta L = 2.00\text{e-}5$  m.

featuring 200 and 400 pixels in the buffer zone. Concerning the Gaussian pulses, a consistent observation across all simulations is that higher frequencies, and consequently shorter wavelengths, exhibit pronounced peaks at sensor S1, while providing minimal information at sensor S2. On the other hand, lower frequencies and longer wavelengths result in pronounced peaks at both sensors S1 and S2. Given this result, further investigations are needed to determine the relationship between ultrasonic wavelength (or excitation frequency) and the characteristic pore size of the digital rock.

In the model featuring a 200-pixel buffer zone (Figure 34), interesting patterns emerge for frequencies 1.13 MHz, 1.51 MHz, 2.27 MHz, and 4.55 MHz in the S2 sensor curve. Notably, two peaks are observable, with the second peak exhibiting a higher amplitude than the first. However, this pattern is absent in the S2 sensor curve simulation with a 400-pixel buffer zone (Figure 35). This difference occurs exclusively due to the disparity in the size of the buffer zone. Furthermore, another notable occurrence is the presence of minor peaks in the S1 sensor curve after the first peak, which are only visible in the analysis of the 200-pixel buffer zone and missing in the simulations of the 400-pixel buffer zone. These secondary curves, resulting from reflections and transmission, exhibit a diminishing trend as the buffer zone increases. It is important to note that these secondary curves can affect the calculation of effective velocities (as shown in Figure 34) for 1.13 MHz, 0.76

MHz, and 0.57 MHz frequencies. Consequently, future studies will explore alternative boundary conditions to mitigate these undesired reflections, such as the PML techniques [91, 92].

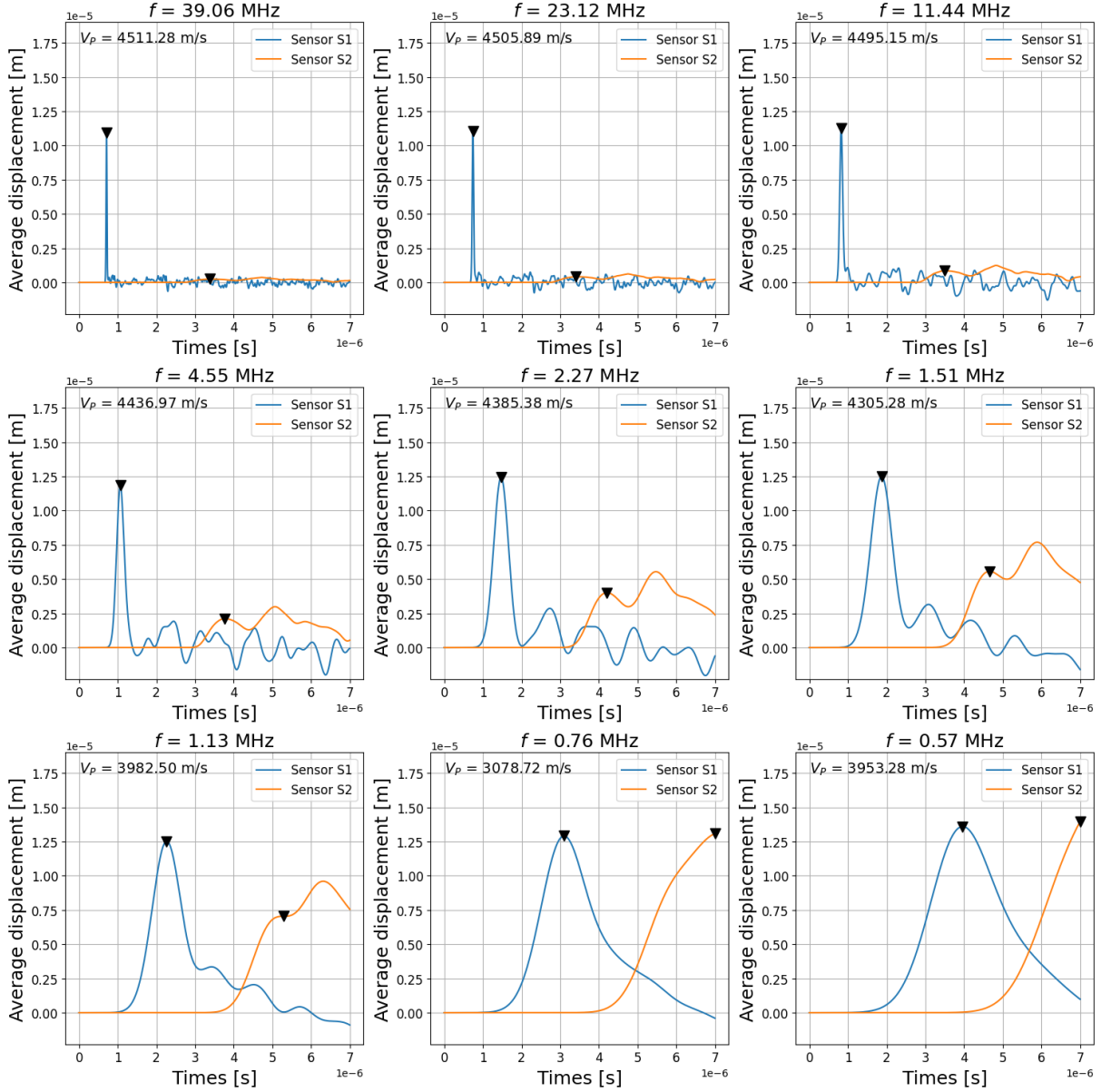


Figure 34: Effective velocities and average displacement in sensors S1 and S2 for various frequencies of Gaussian pulse using a 200-pixel buffer zone model.

To better illustrate the undesired effect generated by the buffer zone reduced size, we created Figure 36. In Figures 36(a) and 36(b), the average displacement vs. time curves for sensors S1 and S2 are presented, respectively, considering models with 200 and 400-pixel buffer zones at frequencies  $f = 1.51$  MHz and  $f = 1.44$  MHz. In these curves, it is also possible to observe the undesired effect on the average displacement of the S2 curve due to the reduced-size buffer zone.

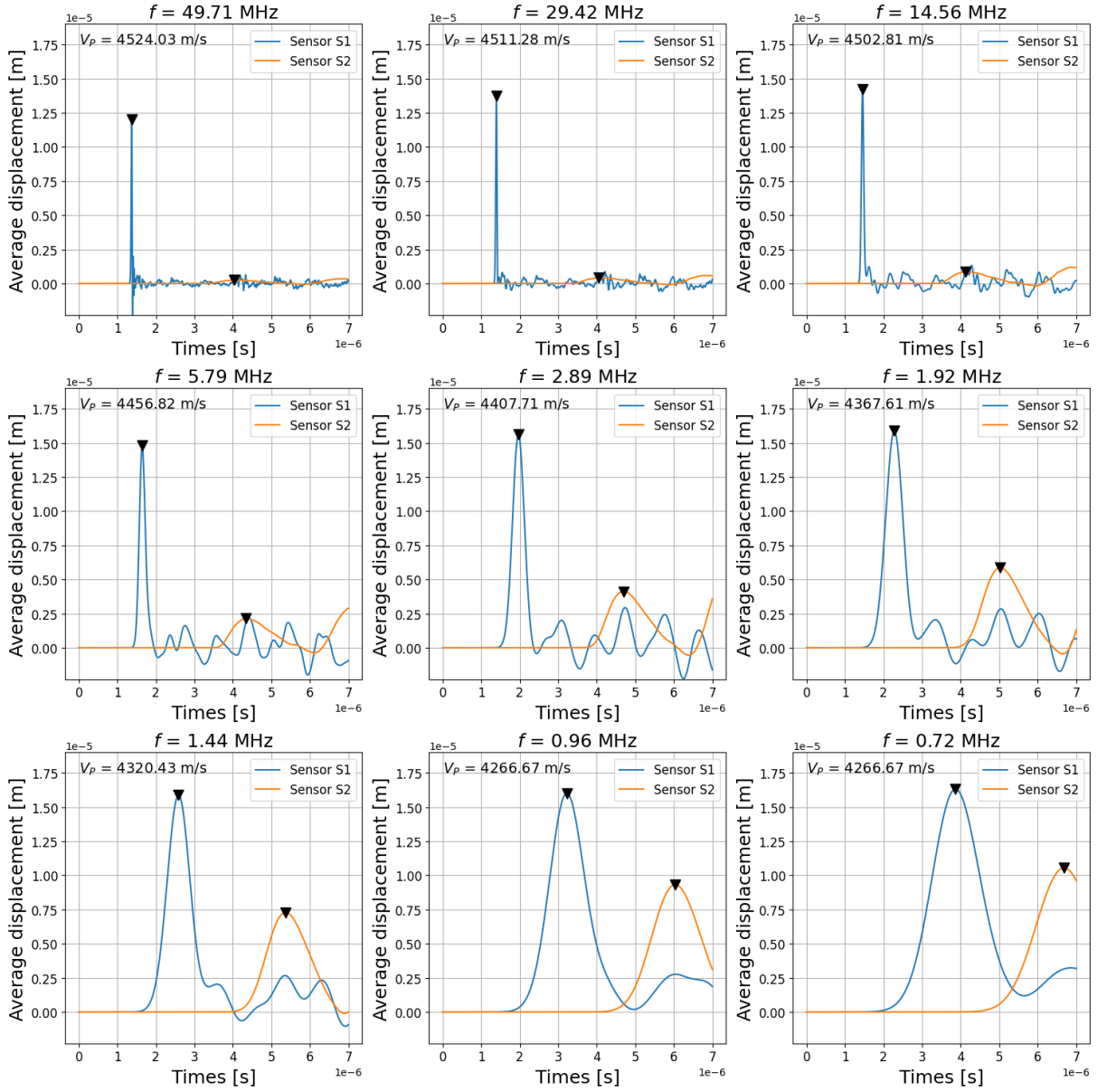


Figure 35: Effective velocities and average displacement in sensors S1 and S2 for various frequencies of Gaussian pulse using a 400-pixel buffer zone model.

For each model, three specific times  $w$  were selected to generate a displacement map, utilizing a color map for visualization to represent displacements across all models at a given time. The initial chosen time  $t_1$  corresponds to the peak of the first S1 sensor, as depicted in Figures 36(c) and 36(f). In these instances, the Gaussian pulse exhibits apparent similarities and behaves predictably. In Figures 36(d) and 36(g), representing intermediate time  $t_2$ , it is possible to observe reflected displacements in the left buffer zone on the free surface, generating secondary measurements in sensor S1. Finally, in Figure 36(e) time  $t_3$ , displacement values already reflected on the right free surface can be seen. These reflected waves, absent in Figure 36(h), give rise to the undesirable effect on

the average displacement of the S2 curve. Hence, as a potential preliminary adjustment to the model, the adoption of distinct sizes for the right and left buffer zones is considered, recognizing their differential contributions to the analysis.

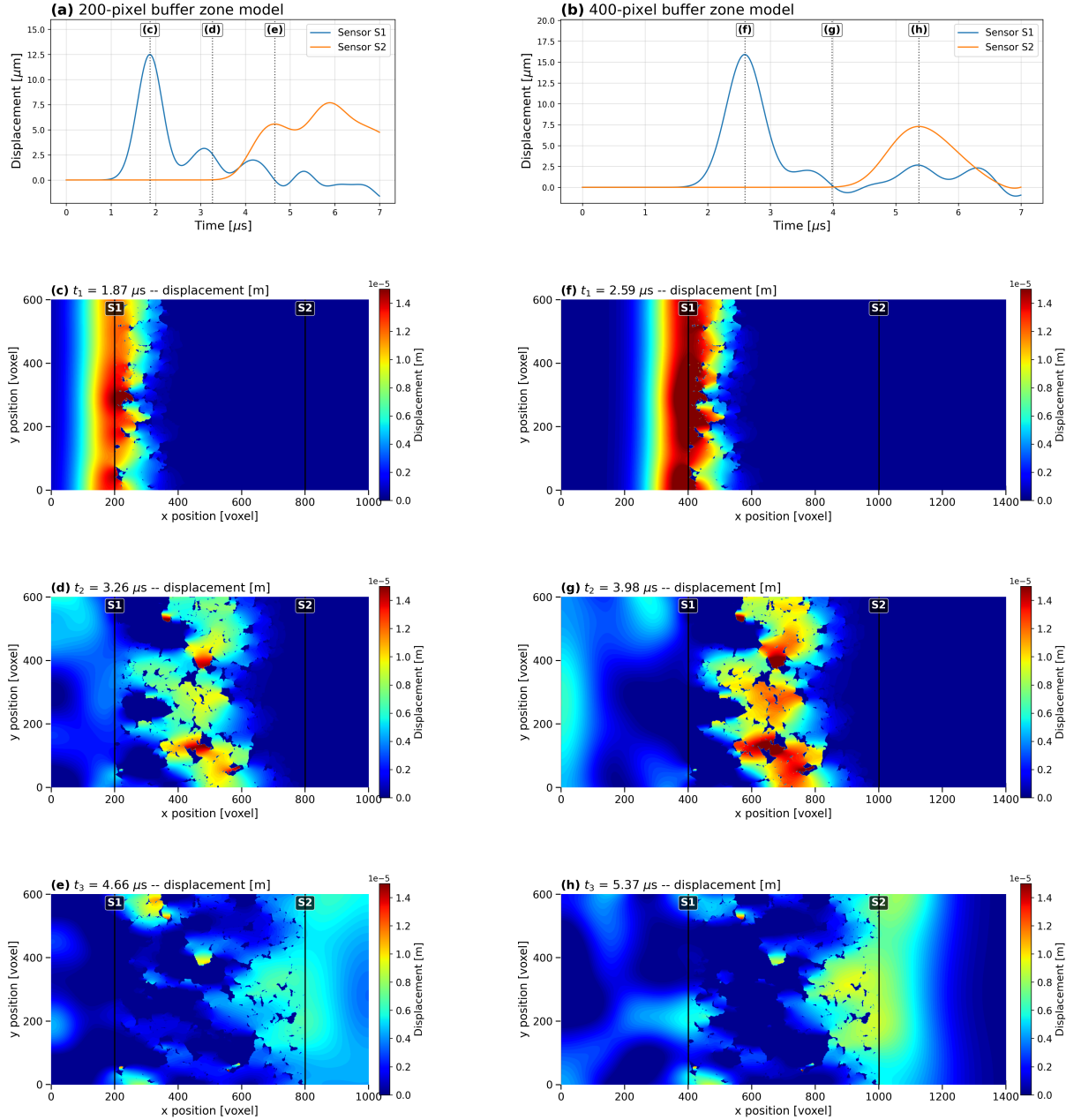


Figure 36: Displacement maps for 200 and 400-pixel buffer zone models at frequencies  $f = 1.51$  MHz and  $f = 1.44$  MHz showing: (a) average displacement in sensors S1 and S2 in the 200-pixel model; (b) average displacement in sensors S1 and S2 in the 400-pixel model; (c) displacement map for  $t_1$  in the 200-pixel model; (d) displacement map for  $t_2$  in the 200-pixel model; (e) displacement map for  $t_3$  in the 200-pixel model; (f) displacement map for  $t_1$  in the 400-pixel model; (g) displacement map for  $t_2$  in the 400-pixel model; and (h) displacement map for  $t_3$  in the 400-pixel model.

### 5.2.2 Effective Velocities of Digital Cores

In light of the Gaussian pulse frequencies and an optimized buffer zone, dynamic simulations were conducted to determine the effective velocities  $V_P$  and  $V_S$  of the 2D rock samples (Figure 31). For comparison, the effective properties of the same images were also calculated using the static program [19]. This program computes the effective property as a function of  $E$  and  $\nu$ . To ensure methodological consistency, these results were subsequently transformed into effective velocities using the conversion Table 27 provided in Appendix A. All simulations were performed using a uniform buffer zone of 400 pixels, with a temporal resolution of  $\Delta t = 3.182 \times 10^{-9}$  s, a spatial discretization of  $\Delta L = 2.000 \times 10^{-5}$  m, and a CFL number of 0.9552. The complete results for each simulation are summarized in Table 11.

Table 11: Effective velocities results and times metrics for analyzing rock samples.

| Digital Core | Static      |             |          | Dynamic     |             |          |
|--------------|-------------|-------------|----------|-------------|-------------|----------|
|              | $V_P$ [m/s] | $V_S$ [m/s] | Time [s] | $V_P$ [m/s] | $V_S$ [m/s] | Time [s] |
| BB           | 4109.00     | 2620.50     | 118.90   | 4266.67     | 2441.68     | 563.70   |
| BG           | 4176.52     | 2619.46     | 117.55   | 4341.09     | 2527.65     | 576.45   |
| CO           | 4906.61     | 3211.44     | 139.53   | 5175.03     | 2974.62     | 664.79   |

An examination of the variance in the velocity results reveals a discrepancy ranging between 4% and 8% when comparing the static and dynamic simulation outcomes. However, in the absence of experimental UPT data, it is not possible to determine which simulation approach exhibits better convergence toward the experimental response. It is expected that the transition to three-dimensional simulations will lead to a closer agreement between the effective velocity values obtained from both the static and dynamic approaches (see Section 5.4).

## 5.3 Performance and Speedup Analysis

### 5.3.1 Homogeneous Benchmark Model

To quantify the computational gains achieved by the parallel implementations, a performance benchmark was conducted using a homogeneous single-material model with a  $300 \times 100 \times 100$  voxel grid, comprising  $3 \times 10^6$  elements and approximately  $9.2 \times 10^6$

degrees of freedom. Five solver variants were evaluated: the serial CPU baseline, a serial CPU element-wise strategy, an OpenMP shared-memory parallel version, a CUDA GPU solver with a global DOFMAP (CUDA), and a CUDA GPU solver using an element-wise strategy (CUDA\_EW). The hardware configuration used in all experiments is described in Table 7.

The execution times and corresponding speedups relative to the serial baseline are reported in Table 12. The serial solver required 2,367.61 s to complete the simulation, while the OpenMP version reduced this to 1,201.93 s, yielding a speedup of  $1.97\times$ . The CUDA solver achieved 16.38 s ( $144.54\times$  speedup), and the CUDA\_EW variant reached 16.04 s ( $147.57\times$  speedup), the best overall result.

Table 12: Execution time, speedup, and memory consumption for the evaluated solver variants on the model  $300 \times 100 \times 100$  benchmark.

| Solver    | Time [s] | Speedup                           | Memory [MB]   |
|-----------|----------|-----------------------------------|---------------|
| Serial    | 2,367.61 | $1.00\times$                      | 219.65        |
| Serial_EW | 2,348.67 | $1.00\times$                      | 125.90        |
| OpenMP    | 1,203.78 | $1.97\times$                      | 219.65        |
| CUDA      | 16.38    | $144.54 \times$                   | 219.65        |
| CUDA_EW   | 16.04    | <b><math>147.57 \times</math></b> | <b>125.90</b> |

A particularly relevant aspect of the CUDA\_EW variant is its memory efficiency. By avoiding the explicit storage of the full DOFMAP and instead recomputing element-level connectivity on the fly, CUDA\_EW consumes approximately 125.9 MB, compared to 219.6 MB for the standard CUDA and OpenMP implementations. This characteristic is critical for large-scale models that would otherwise exceed available GPU memory.

The OpenMP implementation, while straightforward to deploy, delivers only a  $1.97\times$  improvement than the CUDA\_EW solver. This limited gain reflects the sequential nature of the leapfrog time-marching loop under shared-memory parallelism, and confirms that CPU-based parallelization is insufficient for large-scale time-domain wave propagation problems.

To illustrate the parallel implementation, Figures 37, 38, and 39 display the displacements of the  $x$ -,  $y$ -, and  $z$ -components, respectively, computed by the CUDA\_EW solver on the  $300 \times 100 \times 100$  benchmark model. Each figure presents four snapshots at time steps 200, 400, 600, and 800. The Gaussian pulse advancing from the inlet toward the outlet



along the  $x$ -axis. Sensors S1 and S2 are positioned to record the average displacement. The  $x$ -displacement field (Figure 37) captures the compressional wave front, while the  $y$ - and  $z$ -displacement fields (Figures 38 and 39) represent the transverse shear components.

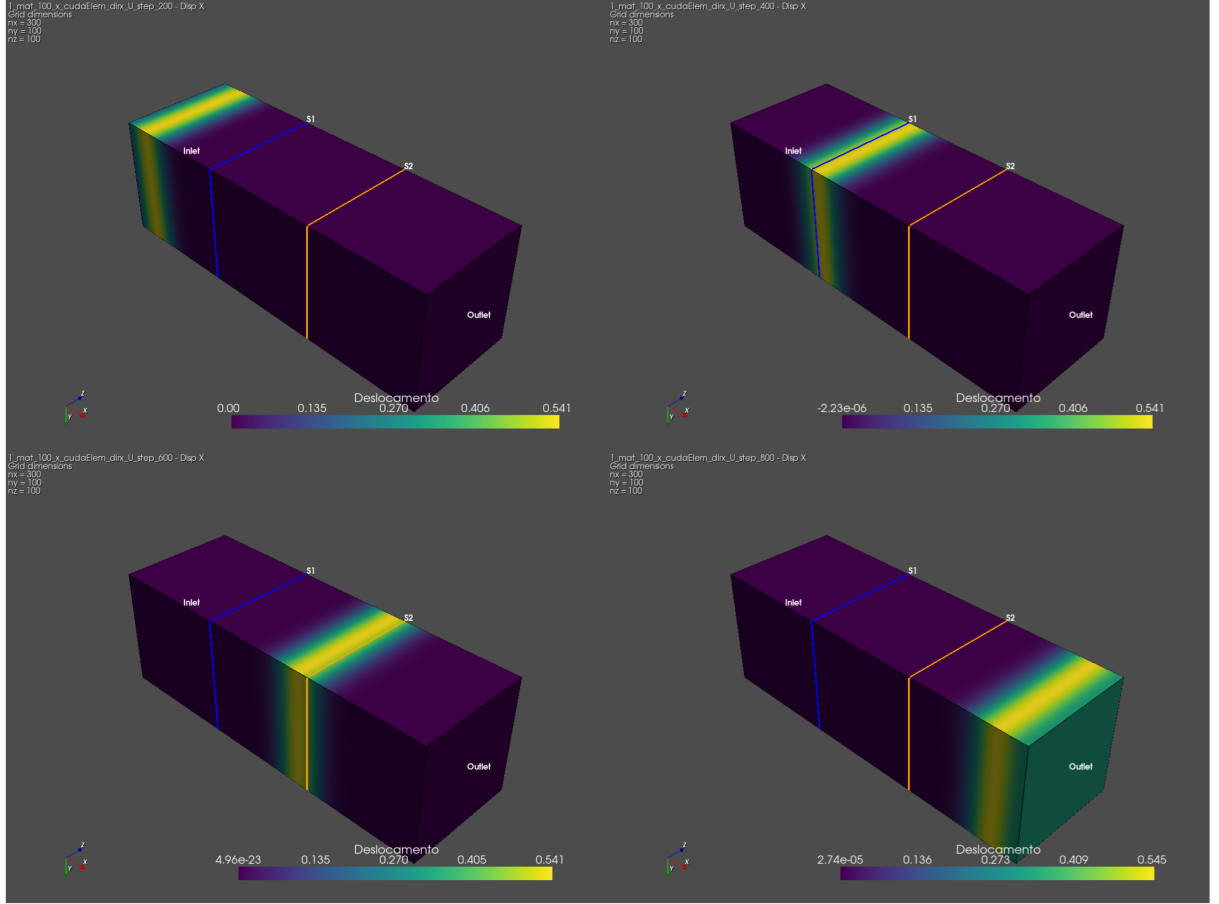


Figure 37: Displacement of the  $x$ -component for the homogeneous  $300 \times 100 \times 100$  benchmark model computed with the CUDA\_EW solver, showing snapshots at time steps 200, 400, 600, and 800. The color scale represents the magnitude of displacement. The inlet (source), sensors S1 and S2, and the outlet face are indicated.

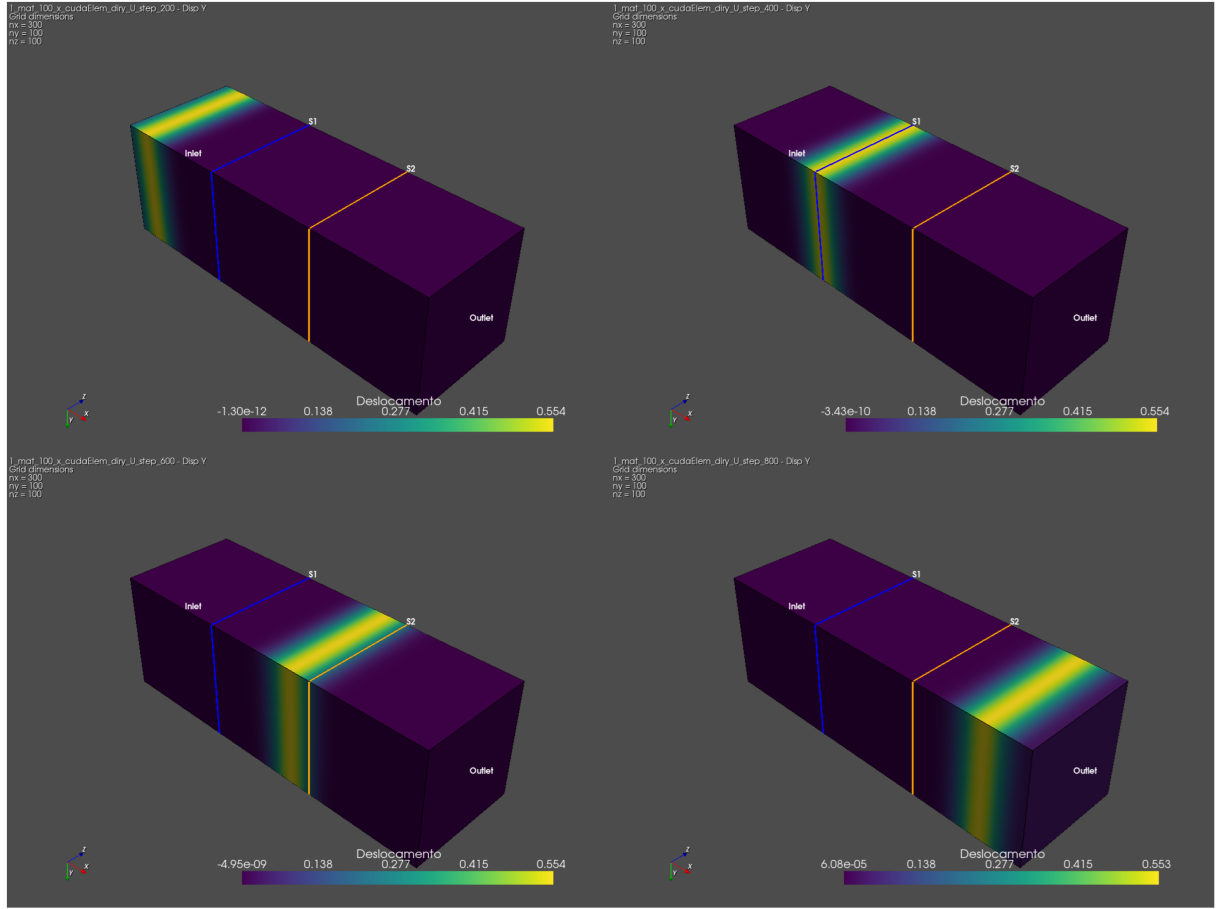


Figure 38: Displacement of the  $y$ -component (transverse shear) for the homogeneous  $300 \times 100 \times 100$  benchmark model computed with the CUDA\_EW solver at time steps 200, 400, 600, and 800. The wave propagates along the  $x$ -axis, and the  $y$ -displacement field characterizes the shear wave transverse to the propagation direction.

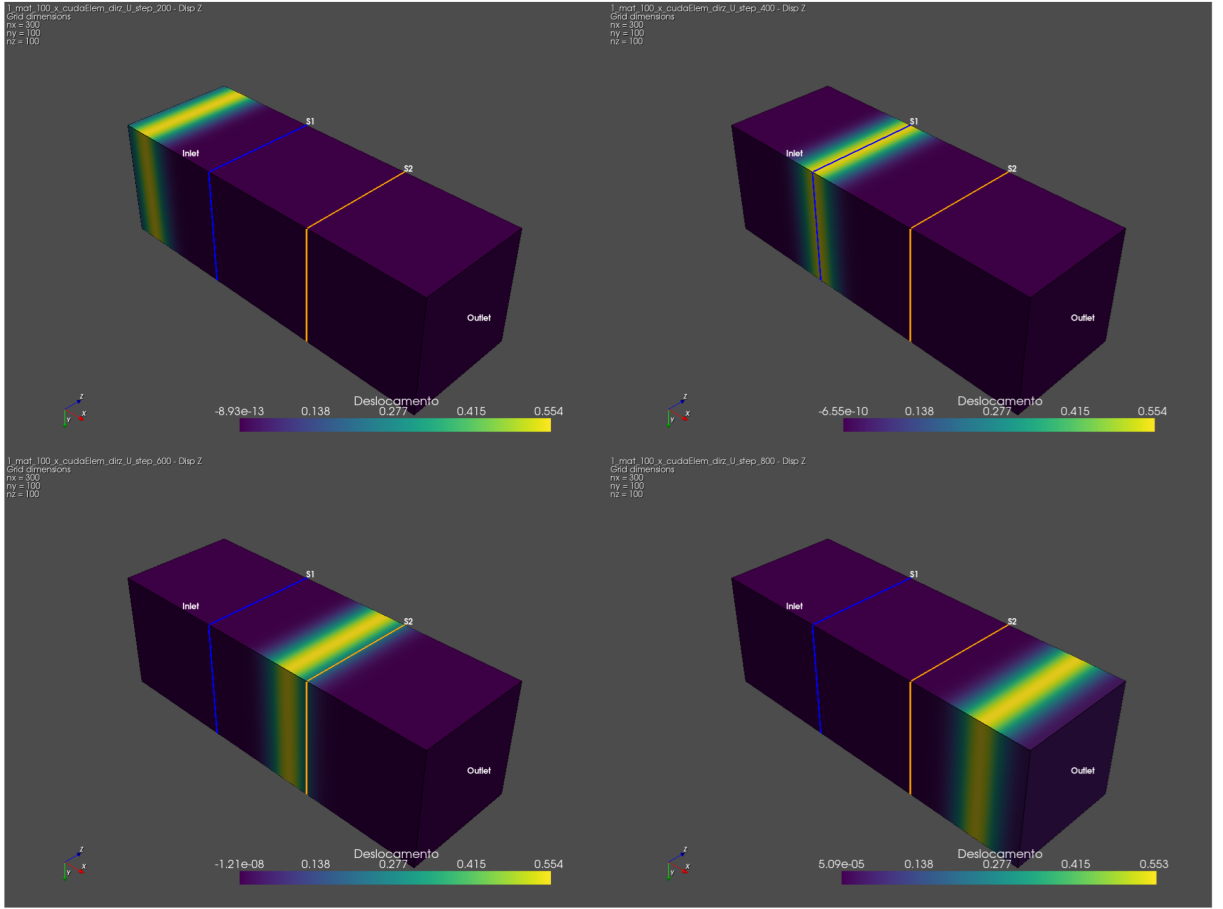


Figure 39: Displacement of the  $z$ -component (transverse shear) for the homogeneous  $300 \times 100 \times 100$  benchmark model computed with the CUDA\_EW solver at time steps 200, 400, 600, and 800. The  $z$ -displacement field is analogous to the  $y$ -component and confirms the expected symmetric shear wave response in both transverse directions.

### 5.3.2 Scalability on a Digital Rock: Fontainebleau Sandstone

While the benchmark above employed an idealized, homogeneous voxel grid, an additional scalability test was performed directly on segmented  $\mu$ CT sub-volumes of the Fontainebleau sandstone (Section 5.4.1), using cubic sub-volumes of  $50^3$ ,  $100^3$ ,  $150^3$ ,  $200^3$ ,  $250^3$ , and the full  $288 \times 288 \times 300$  voxels, all at the native voxel resolution of  $7.5 \mu\text{m}$ . Unlike the homogeneous case, this test exercises the real, heterogeneous pore-solid geometry of the digital rock, giving a more representative estimate of solver performance for the large-scale simulations discussed later in this chapter.

Running the serial and OpenMP-based solvers for the thousands of time steps required by a converged velocity measurement is not practical at these scales; for the largest sub-volumes, a single full-length run could take several days. For this reason, each solver/mesh

configuration was instead executed for a short, fixed window of only 20 time steps, and the comparison metric adopted was the average wall-clock time per time step (Table 13) logged internally by the solver, rather than the total simulation time. Because this metric does not depend on how many steps were actually executed, it allows a fair, like-for-like comparison between very fast (GPU) and very slow (serial/OpenMP) implementations without requiring any of them to run to completion, while still allowing the cost of a full-length simulation to be estimated by extrapolation.

Table 13: Average wall-clock time per time step for the five solver variants applied to Fontainebleau sandstone sub-volumes of increasing size.

| Sub-volume                  | Serial [ms] | Serial_EW [ms] | OpenMP [ms] | CUDA [ms] | CUDA_EW [ms] |
|-----------------------------|-------------|----------------|-------------|-----------|--------------|
| $50^3$                      | 155.14      | 174.378        | 171.301     | 8.368     | 8.211        |
| $100^3$                     | 2426.036    | 2176.576       | 1707.052    | 58.263    | 55.895       |
| $150^3$                     | 5900.507    | 5049.835       | 5018.592    | 149.737   | 147.053      |
| $200^3$                     | 18561.38    | 17754.116      | 11803.271   | 458.105   | 448.316      |
| $250^3$                     | 30366.549   | 39785.602      | 26300.404   | 894.789   | 875.368      |
| $288 \times 288 \times 300$ | 53778.688   | 54468.809      | 34770.699   | 3760.526  | 3612.421     |

## 5.4 Validation against Digital Rock Benchmarks

In DRP, numerical homogenization combined with  $\mu$ CT images allows for the non-destructive evaluation of the physical behavior of complex media. In this context, the validation of new dynamic solvers, including the ones provided by this thesis, for large-scale rocks requires the use of reliable rock benchmarks. This highlights the work of Andrä et al. [65, 14], which provides reference datasets based on samples of Berea sandstone, Fontainebleau sandstone, and Grosmont carbonate rocks.

While the first part of the study by Andrä et al. [65] focused on image acquisition and segmentation to delineate solid and pore phases, the second part [14] computed macroscopic properties including absolute permeability, electrical resistivity, and elastic moduli. For the elastic moduli analyses, the authors applied a diverse range of numerical algorithms. The suite of evaluated methods included solvers based on the FEM [106] (denoted as FEM-F and FEM-C for Fortran and C++ implementations), the Dynamic Pulse Propagation Finite-Difference solver [107] (FDM-dynamic) from ETH Zürich, and an integral formulation solver [108] based on the Lippmann-Schwinger (L-S) equation using

the Fast Fourier Transform (FFT). The variability and consistency of the results reported by these different algorithms in the literature have established a “gold standard” in the literature for benchmarking new computational tools.

Before comparing velocity values, it is important to contextualize the expected level of agreement. As documented by Andrä et al. [14], the variability in elastic moduli across the different methods ranges from 3% to 18% of their midrange values, even when applied to the same segmented image. Part of this spread is traceable not to algorithmic deficiencies but to the image segmentation process: Andrä et al. [65] found that segmentation-induced porosity variations span from  $\approx 6.4\%$  of midrange for Berea sandstone to as much as 31% for Grosmont carbonate. Consequently, observed discrepancies between the present solver and the reference methods should be interpreted in this context: differences within the method-variability range of Andrä et al. [14] are consistent with a well-functioning implementation.

In this regard, the central objective of this section is the validation of the computational solver developed during this research. To ensure a methodologically sound assessment, the validation is conducted using the exact same microtomographic images made available and consolidated in the benchmarks by Andrä et al. [14]. The primary focus is a direct comparison of the resulting elastic property values. To demonstrate the accuracy and performance of the dynamic CUDA-based solver, the simulation results focus on estimating compressional ( $V_P$ ) and shear ( $V_S$ ) wave propagation velocities. We compare these results not only against the extensive reference values established by Andrä et al. [14] methods but also against results from the Computational Homogenization with the image-based Finite Element Method (CHFEM) by Lopes et al. [19].

The necessity for this validation arises from the specific architecture of the numerical model proposed here. Employing the FEM to simulate elastic wave propagation, the algorithm was designed in CUDA to leverage the massively parallel architectures of GPUs. Since digital rock models often reach millions of degrees of freedom and rapidly exhaust conventional RAM limits, this validation will demonstrate that the new implementation is a viable and efficient option for large-scale analysis.

The static reference employed throughout this section is the CHFEM solver, which was originally developed in the context of image-based numerical homogenization for elastic properties [109] and later extended for large-scale GPU computing [19]. CHFEM adopts an assembly-free formulation of the FEM, in which each voxel of the  $\mu$ CT image is directly mapped to a trilinear hexahedral element. The system of linear equations arising from the

discretization of the static elasticity problem is solved iteratively using the Preconditioned Conjugate Gradient (PCG) method, combined with a NbN stencil strategy that avoids the explicit construction of the global stiffness matrix. This design substantially reduces memory consumption, enabling analyses of models with hundreds of millions of degrees of freedom on consumer-grade GPU hardware. Effective macroscopic elastic moduli ( $K$  and  $G$ ) are recovered by post-processing the energy-minimizing displacement field under prescribed homogeneous boundary conditions. The CHFEM solver has been validated against analytical solutions and independent numerical benchmarks [109, 19], and is used here as a reference static tool to complement the dynamic CUDA-based analyses. For the static and dynamic comparison we will use the workflow described in Figure 40.

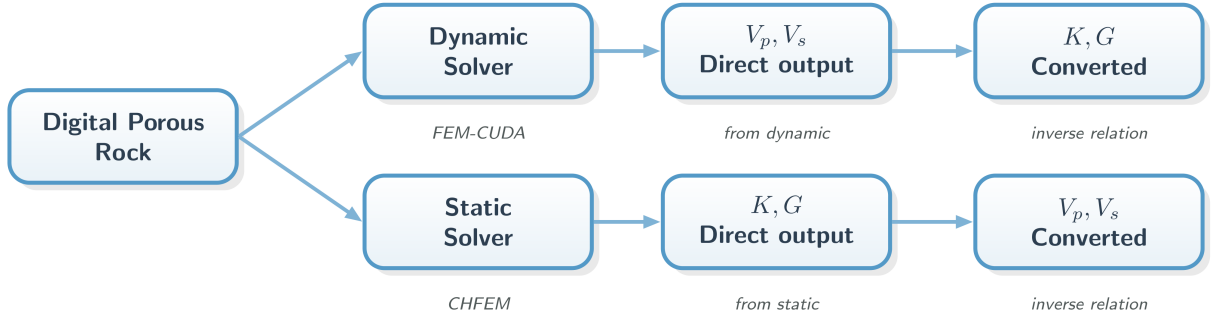


Figure 40: Workflow of the digital rock benchmarks, illustrating the parallel computation paths: the dynamic solver yielding  $V_P, V_S$  with subsequent conversion to  $K, G$ , and the static solver yielding  $K, G$  with subsequent conversion to  $V_P, V_S$ .

#### 5.4.1 Fontainebleau Sandstone

Fontainebleau sandstone is frequently modeled and adopted as a reference material in reservoir characterization studies due to its intrinsically cleaner and monodisperse structure [110]. This porous medium is ideal for evaluating numerical algorithms in their first phases, as it primarily consists of well-cemented quartz grains, a mineral characterized by distinct physical features (homogeneous at the sample scale [111]), including a density of  $2.65 \text{ g/cm}^3$ , a bulk modulus of  $37.0 \text{ GPa}$ , and a shear modulus of  $44.0 \text{ GPa}$  [2, 112]. The rock dataset by Andr  et al. [65] was provided as a segmented binary volume, measuring  $288 \times 288 \times 300$  voxels ( $x \times y \times z$ ). Each voxel represents a physical size of  $7.5 \text{ }\mu\text{m}$  and has an experimental porosity of  $14.7\%$  (Table 2 of Andr  et al. [14]). Unlike the Berea and Grosmont samples in the same benchmark the Fontainebleau dataset is provided as a single segmented image and a single segmentation algorithm.

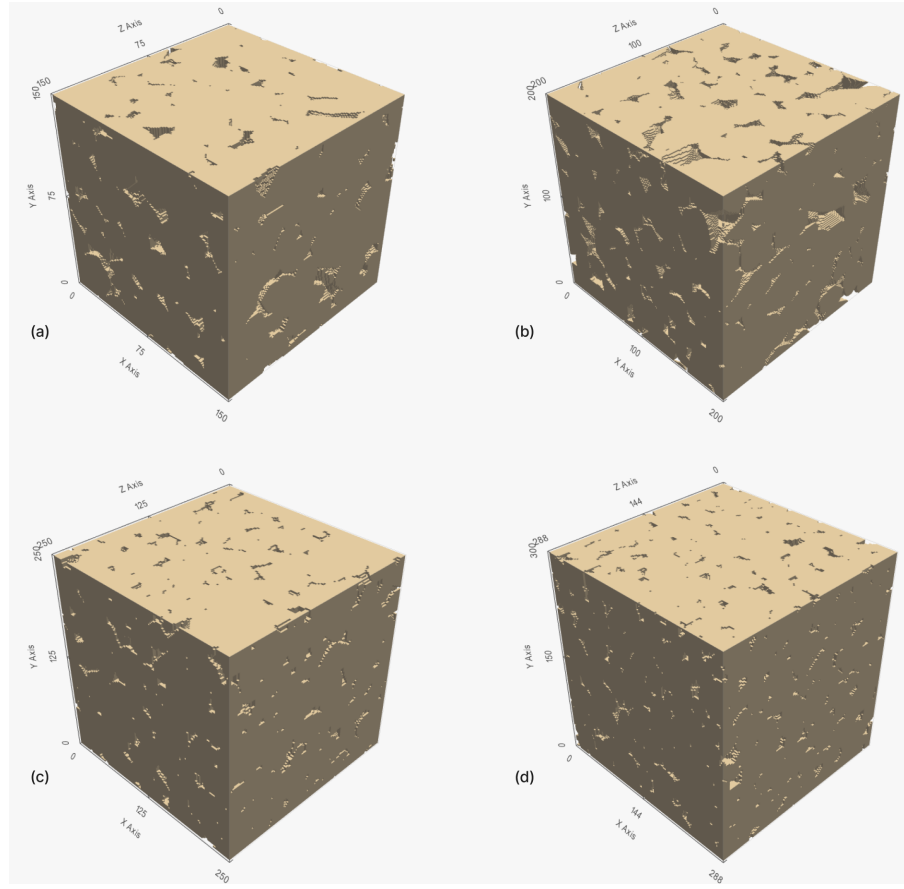


Figure 41: 3D visualizations of the Fontainebleau sandstone voxel-based domains used in the numerical simulations, showing the grain and pore network distribution at different scales: (a)  $150^3$  voxels; (b)  $200^3$  voxels; (c)  $250^3$  voxels; and (d)  $288 \times 288 \times 300$  voxels (original sample). The axes indicate spatial dimensions in voxel units, highlighting the increase in total degrees of freedom across the varied sub-volumes.

In this work, the original image of the Fontainebleau sandstone was subdivided to generate three new subsamples with dimensions of  $250^3$ ,  $200^3$ , and  $150^3$  voxels, as illustrated in Figure 41. This multi-scale strategy serves three concurrent purposes: (i) it quantifies how the effective properties converge toward RVE values as the domain size increases; and (ii) it allows a systematic study of the computational cost in terms of degrees of freedom using the dynamic FEM-CUDA solver; and (iii) it compares the agreement of static and dynamic results at different scales.

Table 14: Geometric and computational properties of the Fontainebleau samples.

| Samples                     | Porosity ( $\phi$ ) | Element size | Solid DOFs         | Total DOFs         |
|-----------------------------|---------------------|--------------|--------------------|--------------------|
| $150 \times 150 \times 150$ | 0.1454              | $7.50 \mu m$ | $2.28 \times 10^7$ | $2.40 \times 10^7$ |
| $200 \times 200 \times 200$ | 0.1471              | $7.50 \mu m$ | $7.00 \times 10^7$ | $7.28 \times 10^7$ |
| $250 \times 250 \times 250$ | 0.1466              | $7.50 \mu m$ | $1.37 \times 10^8$ | $1.42 \times 10^8$ |
| $288 \times 288 \times 300$ | 0.1467              | $7.50 \mu m$ | $1.98 \times 10^8$ | $2.06 \times 10^8$ |

The geometric and computational properties of all four samples are summarized in Table 14. The intrinsic characteristics of these models, including porosities, element dimensions, and the DOF. The porosity varies by less than 0.2% points across all sub-volumes, confirming that the rock microstructure is sufficiently homogeneous at this scale for the extracted sub-volumes to be representative of the parent image. The computational properties presented in Table 14 reflect the impact of the node mapping strategies discussed in Chapter 3. The Total DOFs column represents the upper limit of the model's degrees of freedom, calculated via DNM, where all nodes of the regular grid (including pore regions) are instantiated. In contrast, the Solid DOFs column details the effective size of the DOFs when the ANM algorithm, described in Section 3.4.3, is applied.

Static numerical simulations were conducted for each of the four available samples, allowing for the computation of effective macroscopic properties, notably the bulk modulus ( $K$ ) and shear modulus ( $G$ ), as well as wave velocities ( $V_P$  and  $V_S$ ). In a subsequent stage focused on execution in massively parallel environments (CUDA-based code), six analyses were performed for each of the four samples ( $V_P$  and  $V_S$ ), adopting different pulse lengths for propagation. Later, the velocity results were converted to bulk and shear moduli using the relationships in Table 27.

#### 5.4.1.1 Static Results (CHFEM)

Static numerical simulations were performed using the CHFEM solver. The computed effective bulk modulus ( $K$ ) and shear modulus ( $G$ ) are listed per loading direction in Table 15.



Table 15: Static elastic properties ( $K, G$ ) calculated for the Fontainebleau samples in different directions.

| Samples                     | Bulk Modulus ( $K$ ) |       |       | Shear Modulus ( $G$ ) |          |          |
|-----------------------------|----------------------|-------|-------|-----------------------|----------|----------|
|                             | $K_x$                | $K_y$ | $K_z$ | $G_{xy}$              | $G_{xz}$ | $G_{yz}$ |
| $150 \times 150 \times 150$ | 23.86                | 24.90 | 23.91 | 26.72                 | 26.17    | 26.64    |
| $200 \times 200 \times 200$ | 23.93                | 24.43 | 24.01 | 26.44                 | 26.15    | 26.32    |
| $250 \times 250 \times 250$ | 24.15                | 24.21 | 24.07 | 26.38                 | 26.38    | 26.38    |
| $288 \times 288 \times 300$ | 24.06                | 24.20 | 24.18 | 26.43                 | 26.40    | 26.31    |

*Note: Values expressed in [GPa].*

For the static benchmark runs with CHFEM, the computed values for Fontainebleau fall in the ranges  $K \in [23.86, 24.43]$  GPa,  $G \in [26.15, 26.72]$  GPa,  $V_P \in [4536.7, 5138.4]$  m/s, and  $V_S \in [2921.7, 3435.1]$  m/s. The spread in directional values is small in all cases, indicating near-isotropic macroscopic behavior consistent with the homogeneous character of Fontainebleau sandstone [14]. The directional anisotropy decreases as the domain size increases: for the  $150^3$  sub-volume the range  $K_x - K_y$  spans 1.04 GPa, while for the full image this spread is reduced to 0.14 GPa, reflecting convergence toward an isotropic effective medium with increasing sample size. For the largest sample, with dimensions  $288 \times 288 \times 300$ , one complete CHFEM execution, comprising the initial guess stage and the six loading cases used to obtain the homogenized properties, required approximately 412.0 s. The corresponding peak GPU memory requirement was 2.542 GB, as reported in CHFEM's log.

#### 5.4.1.2 Dynamic Results (FEM-CUDA)

For the dynamic CUDA-based analyses, the directional wave velocities were extracted from the sensors outputs for all tested scales. For each sample direction, three simulations are required to determine the compressional and shear wave velocities. The results of the performed simulations are presented in Table 16. For the largest sample ( $288 \times 288 \times 300$ ), the adopted pulse frequencies were 0.80 MHz for  $V_P$  and 0.57 MHz for  $V_S$ . The dynamic simulations indicate a clear directional pattern with  $V_{P,x}$  consistently larger than the transverse components, while  $V_{S,y}$  and  $V_{S,z}$  remain close to each other. Representative values are: for  $150^3$ ,  $(V_{P,x}, V_{S,y}, V_{S,z}) = (5172.4, 3422.9, 3395.4)$  m/s; for  $200^3$ ,  $(5046.7, 3377.1, 3358.2)$  m/s; for  $250^3$ ,  $(5058.5, 3373.3, 3368.3)$  m/s; and for  $288 \times 288 \times 300$ ,  $(5037.9, 3363.2, 3363.2)$  m/s. This trend supports a near-isotropic transverse

response with mild anisotropy in the propagation axis.

Table 16: Velocity properties ( $V_P, V_S$ ) for the Fontainebleau samples along the  $x, y$ , and  $z$  axes using dynamic CUDA-based analyses.

| Samples                     | x-axis    |           |           | y-axis    |           |           | z-axis    |           |           |
|-----------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
|                             | $V_{P,x}$ | $V_{S,y}$ | $V_{S,z}$ | $V_{P,y}$ | $V_{S,x}$ | $V_{S,z}$ | $V_{P,z}$ | $V_{S,x}$ | $V_{S,y}$ |
| $150 \times 150 \times 150$ | 5172.4    | 3422.9    | 3395.4    | 5208.3    | 3443.9    | 3388.6    | 5257.0    | 3363.2    | 3372.4    |
| $200 \times 200 \times 200$ | 5046.7    | 3377.1    | 3358.2    | 5070.4    | 3383.5    | 3358.2    | 5094.3    | 3383.5    | 3389.8    |
| $250 \times 250 \times 250$ | 5058.5    | 3373.3    | 3368.3    | 5049.4    | 3363.2    | 3368.3    | 5067.6    | 3373.3    | 3373.3    |
| $288 \times 288 \times 300$ | 5037.9    | 3363.2    | 3363.2    | 5056.2    | 3370.2    | 3370.2    | 5052.6    | 3363.2    | 3372.4    |

*Note: Velocities in [m/s].*

#### 5.4.1.3 RVE Convergence

The four sub-volumes of the Fontainebleau dataset also serve as a convergence study for the RVE size. Because the  $150^3$  sub-sample contains only  $\approx 14.5\%$  porosity while the full  $288 \times 288 \times 300$  sample yields  $14.67\%$ , the porosity variation across sub-volumes is small for this material. A more stringent test of RVE convergence is provided by tracking how  $V_P$  and  $V_S$  change as the sub-volume grows from  $150^3$  to  $288 \times 288 \times 300$  voxels. Convergence of the effective velocities with increasing sample size provides confidence that the chosen sub-volume adequately represents the macroscopic behavior of the rock, consistent with the methodology recommended by Andrä et al. [14]. As the domain increases from  $150^3$  to the original  $288 \times 288 \times 300$  sample, the estimated  $V_{P,x}$  converges to approximately  $5.0 \times 10^3$  m/s and the shear components stabilize around  $3.36 \times 10^3$  m/s. This behavior indicates that boundary and finite-size effects become less relevant for larger digital cores.

#### 5.4.1.4 Comparison with the Andrä et al. Benchmark

The Table 17 presents the effective bulk and shear moduli obtained in this work alongside the reference values reported by Andrä et al. [14] for the full-resolution  $288 \times 288 \times 300$  Fontainebleau image. The benchmark [14] includes four methods: the Lippmann-Schwinger solver (L-S), two finite-element implementations (FEM-C and FEM-F), and the dynamic finite-difference pulse propagation method (FDM-dynamic). Our dynamic FEM-CUDA result ( $K = 23.3$  GPa,  $G = 25.6$  GPa) lies within the midrange established by Andrä ( $K_M = 23.4$  GPa,  $G_M = 26.3$  GPa) and agrees particularly well with the FDM-dynamic approach ( $K = 23.6$  GPa,  $G = 23.5$  GPa) for bulk moduli. The static CHFEM result

( $K \approx 24.2$  GPa,  $G \approx 26.3$  GPa) is consistent with the FEM-C and FEM-F values reported in the benchmark.

Table 17: Fontainebleau: comparison of Bulk ( $K$ ) and Shear ( $G$ ) moduli between this work the Andrä et al. [14] report, and static solver [19]. Andrä values correspond to the  $288 \times 288 \times 300$  full-scale model with porosity  $\phi = 0.147$ .

| Method                       | $K$ [GPa] | $G$ [GPa] | Source      |
|------------------------------|-----------|-----------|-------------|
| CHFEM static (this work)     | 24.2      | 26.3      | [19, 109]   |
| FEM-CUDA dynamic (this work) | 23.5      | 25.6      | this thesis |
| L-S                          | 22.6      | 29.1      | [14]        |
| FEM-C                        | 24.1      | 26.6      | [14]        |
| FEM-F                        | 24.3      | 26.6      | [14]        |
| FDM-dynamic                  | 23.6      | 23.5      | [14]        |
| Midrange (Andrä)             | 23.4      | 26.3      | [14]        |

Notably, the L-S method consistently predicts higher shear moduli ( $G = 29.1$  GPa) compared to all other solvers. This result is trend documented by Andrä et al. [14] across all three benchmark samples (Fontainebleau, Berea, and Grosmont) and attributed to the Fourier-transform-based formulation and its implicit PBC. On the other hand, computationally obtained moduli for Fontainebleau are consistently stiffer than laboratory ultrasonic measurements [112], a bias that Andrä et al. [14] attribute to the under representation of thin compliant porosity (e.g., microcracks at grain contacts) in micro-CT images at this resolution. Overall, the results produced by both the static CHFEM and the dynamic FEM-CUDA solvers fall within the range of benchmark values, validating the implementation against the reference dataset.

### 5.4.2 Berea Sandstones

Berea sandstone is one of the most widely used reference materials in DRP, valued for its well-characterized pore network and moderate heterogeneity [14]. The rock dataset by Andrä et al. [65] was provided as a segmented binary volume with voxel size  $0.74 \mu\text{m}$ . Unlike Fontainebleau, Berea is available in multiple segmented versions, and the subvolumes analyzed here present porosities ranging from 0.1733 to 0.1840. Four cubic sub-volumes of sizes  $250^3$ ,  $300^3$ ,  $350^3$ , and  $400^3$  voxels were extracted from the benchmark dataset, as illustrated in Figure 42, following the same multi-scale strategy adopted for Fontainebleau.

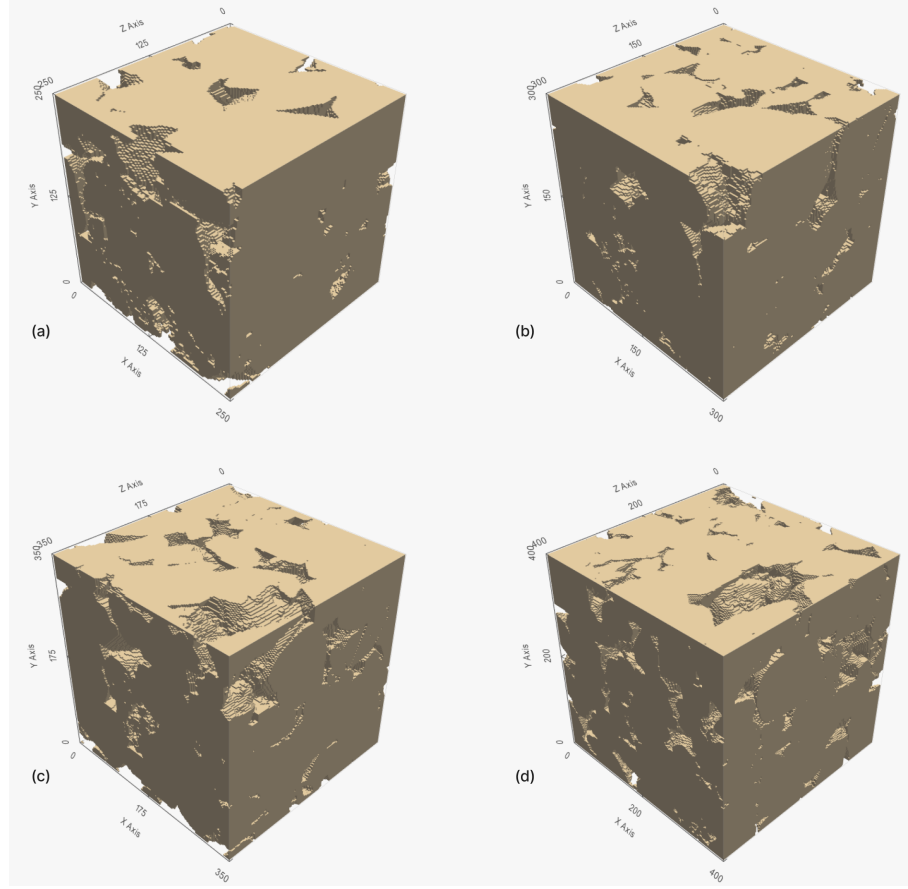


Figure 42: 3D visualizations of the Berea sandstone voxel-based domains used in the benchmark analyses, illustrating the pore-structure heterogeneity across four model scales: (a)  $250^3$  voxels; (b)  $300^3$  voxels; (c)  $350^3$  voxels; and (d)  $400^3$  voxels. The visualizations emphasize the volumetric expansion of the sample used to evaluate the stability of effective elastic properties as a function of the domain size.

The geometric and computational properties of all four samples are summarized in Table 18. The porosity varies slightly across sub-volumes, reflecting the mesoscopic heterogeneity [113] characteristic of Berea sandstone. The Solid DOFs and Total DOFs columns follow the same node-mapping conventions described for Fontainebleau in Section 5.4.1.

Table 18: Geometric and computational properties of the Berea samples.

| Samples                     | Porosity ( $\phi$ ) | Element size | Solid DOFs         | Total DOFs         |
|-----------------------------|---------------------|--------------|--------------------|--------------------|
| $250 \times 250 \times 250$ | 0.1737              | $0.74 \mu m$ | $9.61 \times 10^7$ | $1.04 \times 10^8$ |
| $300 \times 300 \times 300$ | 0.1733              | $0.74 \mu m$ | $1.50 \times 10^8$ | $1.63 \times 10^8$ |
| $350 \times 350 \times 350$ | 0.1803              | $0.74 \mu m$ | $2.18 \times 10^8$ | $2.41 \times 10^8$ |
| $400 \times 400 \times 400$ | 0.1840              | $0.74 \mu m$ | $3.53 \times 10^8$ | $3.86 \times 10^8$ |

Static and dynamic numerical simulations were conducted for each of the four sub-volumes. In the dynamic phase, six analyses were performed per sample to extract  $V_P$  and  $V_S$ , which were subsequently converted to  $K$  and  $G$  using the relationships in Table 27.

#### 5.4.2.1 Static Results (CHFEM)

Static numerical simulations were performed using the CHFEM solver. The computed effective bulk modulus ( $K$ ) and shear modulus ( $G$ ) per loading direction are listed in Table 19.

Table 19: Static elastic properties ( $K, G$ ) calculated for the Berea samples in different directions.

| Samples                     | Bulk Modulus ( $K$ ) |       |       | Shear Modulus ( $G$ ) |          |          |
|-----------------------------|----------------------|-------|-------|-----------------------|----------|----------|
|                             | $K_x$                | $K_y$ | $K_z$ | $G_{xy}$              | $G_{xz}$ | $G_{yz}$ |
| $250 \times 250 \times 250$ | 20.84                | 19.95 | 22.02 | 22.62                 | 22.93    | 21.63    |
| $300 \times 300 \times 300$ | 21.36                | 19.89 | 21.97 | 22.61                 | 23.05    | 21.98    |
| $350 \times 350 \times 350$ | 20.88                | 19.93 | 20.49 | 21.65                 | 21.70    | 21.75    |
| $400 \times 400 \times 400$ | 20.55                | 20.25 | 19.75 | 21.53                 | 21.39    | 21.64    |

*Note: Values expressed in [GPa].*

For the static benchmark runs with CHFEM, the computed values for Berea fall in the ranges  $K \in [19.89, 32.93]$  GPa and  $G \in [21.39, 26.35]$  GPa across the tested sub-volumes. Compared with Fontainebleau, the directional spread in  $K$  is more pronounced, reflecting the stronger mesoscopic heterogeneity [113] of Berea sandstone. The directional anisotropy shows a mild decreasing trend as the domain size increases from  $250^3$  to  $400^3$ , consistent with averaging of heterogeneous pore structures over larger representative volumes.

### 5.4.2.2 Dynamic Results (FEM-CUDA)

For the dynamic CUDA-based analyses, the directional wave velocities were extracted from the sensor outputs for all tested scales. Three simulations per direction were required to obtain compressional and shear wave velocities. The results are listed in Table 20. For the largest sample ( $400^3$ ), the adopted pulse frequencies were 7.72 MHz for  $V_P$  and 5.63 MHz for  $V_S$ .

Table 20: Velocity properties ( $V_P, V_S$ ) for the Berea samples along the  $x, y$ , and  $z$  axes.

| Samples                     | x-axis    |           |           | y-axis    |           |           | z-axis    |           |           |
|-----------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
|                             | $V_{P,x}$ | $V_{S,y}$ | $V_{S,z}$ | $V_{P,y}$ | $V_{S,x}$ | $V_{S,z}$ | $V_{P,z}$ | $V_{S,x}$ | $V_{S,y}$ |
| $250 \times 250 \times 250$ | 5109.3    | 3188.8    | 3293.8    | 5131.4    | 3251.0    | 3259.5    | 5044.4    | 3205.1    | 3242.5    |
| $300 \times 300 \times 300$ | 5026.8    | 3212.5    | 3268.0    | 5040.3    | 3193.1    | 3236.0    | 4921.3    | 3185.4    | 3220.3    |
| $350 \times 350 \times 350$ | 5005.0    | 3155.3    | 3171.4    | 4965.2    | 3204.0    | 3171.4    | 4892.4    | 3158.5    | 3181.1    |
| $400 \times 400 \times 400$ | 4899.7    | 3162.6    | 3165.6    | 4889.2    | 3202.0    | 3153.6    | 4827.3    | 3165.6    | 3189.8    |

*Note: Velocities in [m/s].*

The dynamic results indicate that  $V_{P,x}$  decreases monotonically with model size, from approximately 5109 m/s ( $250^3$ ) to 4900 m/s ( $400^3$ ), while the shear components  $V_{S,y}$  and  $V_{S,z}$  remain in a narrower band near  $3.1\text{--}3.3 \times 10^3$  m/s. Representative velocity tuples are: for  $250^3$ ,  $(V_{P,x}, V_{S,y}, V_{S,z}) = (5109.3, 3188.8, 3293.8)$  m/s; and for  $400^3$ ,  $(4899.7, 3162.6, 3165.6)$  m/s. Again, this behavior is consistent with the expected results as the representative volume increases, and contrasts with the near-isotropic response observed for Fontainebleau. The combination of static and dynamic analyses indicates that Berea is more anisotropic than Fontainebleau in the tested domains. The observed spread in directional velocities is relevant for pulse-based inversion workflows and reinforces the need for multi-directional simulation campaigns when estimating equivalent elastic properties.

### 5.4.2.3 RVE Convergence

The four sub-volumes of the Berea dataset serve as a convergence study for the RVE size. Unlike Fontainebleau, the porosity of Berea sub-volumes varies from 0.1733 ( $300^3$ ) to 0.1840 ( $400^3$ ), indicating that local heterogeneity is more pronounced in this lithology. As the domain increases from  $250^3$  to  $400^3$ , both  $V_{P,x}$  and the shear components show a converging trend, suggesting that the  $400^3$  sub-volume approaches the RVE for this

material. Directional contrasts in  $V_P$  remain relevant across all scales, consistent with the findings documented for Berea in the literature [14].

#### 5.4.2.4 Comparison with the Andrä et al. Benchmark

Table 21 compares the elastic moduli obtained in this work against the Andrä benchmark for a  $400 \times 400 \times 400$  Berea subsample with porosity  $\phi = 0.184$ . Our dynamic FEM-CUDA result ( $K = 21.3$  GPa,  $G = 21.8$  GPa) falls within the range of the Andrä reference values and is in reasonable agreement with the midrange ( $K_M = 19.4$  GPa,  $G_M = 22.1$  GPa). The modest overestimate in  $K$  is consistent with the use of finer voxel discretization in this study compared with the coarsened image employed by the FDM-dynamic solver in Andrä et al. [14]. The static CHFEM result ( $K \approx 19.8$  GPa,  $G \approx 21.6$  GPa) agrees closely with the FEM-C and FDM-dynamic values reported in the benchmark, falling within the method variability range documented by Andrä et al. [14].

Table 21: Berea: comparison of Bulk ( $K$ ) and Shear ( $G$ ) moduli between this work, the Andrä et al. [14] report, and static solver [19]. Andrä values correspond to a  $400^3$  subsample with porosity  $\phi = 0.184$ .

| Method                                 | $K$ [GPa] | $G$ [GPa] | Source      |
|----------------------------------------|-----------|-----------|-------------|
| CHFEM static (this work, $400^3$ )     | 19.8      | 21.6      | [19, 109]   |
| FEM-CUDA dynamic (this work, $400^3$ ) | 21.3      | 21.8      | this thesis |
| L-S                                    | 19.2      | 23.6      | [14]        |
| FEM-C                                  | 19.9      | 20.5      | [14]        |
| FEM-F                                  | 20.1      | 20.7      | [14]        |
| FDM-dynamic                            | 18.6      | 20.6      | [14]        |
| Midrange (Andrä)                       | 19.4      | 22.1      | [14]        |

#### 5.4.3 Grosmont Carbonate

Grosmont carbonate is structurally distinct from the sandstone datasets and exhibits higher porosity levels, ranging from 0.2469 to 0.2651 across the tested sub-volumes. This carbonate benchmark, provided by Andrä et al. [65] as a segmented binary volume with voxel size  $2.02 \mu\text{m}$ , is particularly relevant for applications involving subsurface storage and reservoir-scale geomechanics. Four cubic sub-volumes of sizes  $250^3$ ,  $300^3$ ,  $350^3$ , and  $400^3$

voxels were analyzed, as illustrated in Figure 43, following the same multi-scale strategy adopted for Fontainebleau and Berea.

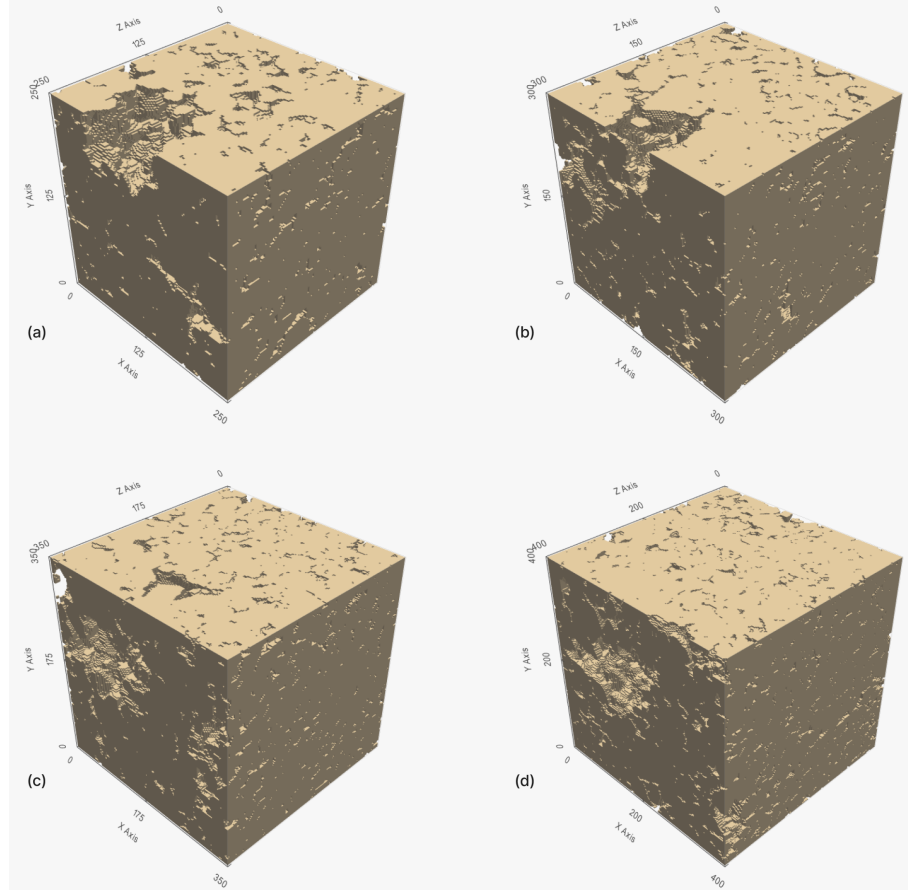


Figure 43: 3D visualizations of the Grosmont carbonate voxel-based domains used in the benchmark analyses, showing the complex vuggy porosity [114] and solid matrix distribution at different scales: (a)  $250^3$  voxels; (b)  $300^3$  voxels; (c)  $350^3$  voxels; and (d)  $400^3$  voxels. The models highlight the significant increase in computational complexity and total degrees of freedom required for the characterization of highly heterogeneous carbonate samples.

The geometric and computational properties of all four samples are summarized in Table 22. The porosity decreases from 0.2651 ( $250^3$ ) to 0.2469 ( $400^3$ ), reflecting the vuggy and heterogeneous pore network typical of carbonate rocks [114]. This larger porosity variability across sub-volumes, compared with Fontainebleau and Berea, is consistent with the segmentation sensitivity reported by Andrä et al. [65] for Grosmont.

Static and dynamic numerical simulations were conducted for each of the four sub-volumes. In the dynamic phase, six analyses were performed per sample to extract  $V_P$  and  $V_S$ , which were subsequently converted to  $K$  and  $G$  using the relationships in Table 27.



Table 22: Geometric and computational properties of the Grosmont samples.

| Samples                     | Porosity ( $\phi$ ) | Element size | Solid DOFs         | Total DOFs         |
|-----------------------------|---------------------|--------------|--------------------|--------------------|
| $250 \times 250 \times 250$ | 0.2651              | $2.02 \mu m$ | $9.75 \times 10^7$ | $1.04 \times 10^8$ |
| $300 \times 300 \times 300$ | 0.2558              | $2.02 \mu m$ | $1.52 \times 10^8$ | $1.63 \times 10^8$ |
| $350 \times 350 \times 350$ | 0.2514              | $2.02 \mu m$ | $2.60 \times 10^8$ | $2.78 \times 10^8$ |
| $400 \times 400 \times 400$ | 0.2469              | $2.02 \mu m$ | $3.59 \times 10^8$ | $3.86 \times 10^8$ |

#### 5.4.3.1 Static Results (CHFEM)

Static numerical simulations were performed using the CHFEM solver. The computed effective bulk modulus ( $K$ ) and shear modulus ( $G$ ) per loading direction are listed in Table 23.

Table 23: Static elastic properties ( $K, G$ ) calculated for the Grosmont samples in different directions.

| Samples                     | Bulk Modulus ( $K$ ) |       |       | Shear Modulus ( $G$ ) |          |          |
|-----------------------------|----------------------|-------|-------|-----------------------|----------|----------|
|                             | $K_x$                | $K_y$ | $K_z$ | $G_{xy}$              | $G_{xz}$ | $G_{yz}$ |
| $250 \times 250 \times 250$ | 24.38                | 15.58 | 25.52 | 11.83                 | 13.64    | 11.85    |
| $300 \times 300 \times 300$ | 24.62                | 16.78 | 25.77 | 12.29                 | 13.88    | 12.29    |
| $350 \times 350 \times 350$ | 23.57                | 17.66 | 25.70 | 12.54                 | 13.75    | 12.43    |
| $400 \times 400 \times 400$ | 23.80                | 18.25 | 26.11 | 12.84                 | 13.87    | 12.68    |

*Note: Values expressed in [GPa].*

For the static benchmark runs with CHFEM, the computed values for Grosmont fall in the ranges  $K \in [13.41, 26.11]$  GPa and  $G \in [10.45, 13.88]$  GPa across the tested sub-volumes. These ranges are substantially lower than those obtained for Fontainebleau and Berea, reflecting the higher porosity and the mechanically weaker carbonate matrix. The directional spread in  $K$  is considerably larger than in either sandstone indicating strong directional sensitivity consistent with the heterogeneous vuggy pore structure of Grosmont.

#### 5.4.3.2 Dynamic Results (FEM-CUDA)

For the dynamic CUDA-based analyses, the directional wave velocities were extracted from the sensor outputs for all tested scales. Three simulations per direction were required to

obtain compressional and shear wave velocities. The results are listed in Table 24. For the largest sample  $400^3$ , the adopted pulse frequencies were 2.91 MHz for  $V_P$  and 1.90 MHz for  $V_S$ .

Table 24: Velocity properties ( $V_P, V_S$ ) for the Grosmont samples along the  $x, y$ , and  $z$  axes.

| Samples                     | x-axis    |           |           | y-axis    |           |           | z-axis    |           |           |
|-----------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
|                             | $V_{P,x}$ | $V_{S,y}$ | $V_{S,z}$ | $V_{P,y}$ | $V_{S,x}$ | $V_{S,z}$ | $V_{P,z}$ | $V_{S,x}$ | $V_{S,y}$ |
| $250 \times 250 \times 250$ | 4639.9    | 2660.7    | 2507.0    | 4113.2    | 2498.0    | 2498.0    | 4697.5    | 2650.6    | 2509.3    |
| $300 \times 300 \times 300$ | 4668.4    | 2675.2    | 2531.1    | 4169.0    | 2515.1    | 2508.0    | 4724.1    | 2663.3    | 2534.7    |
| $350 \times 350 \times 350$ | 4679.1    | 2678.3    | 2556.4    | 4229.6    | 2542.7    | 2527.9    | 4742.5    | 2672.3    | 2553.6    |
| $400 \times 400 \times 400$ | 4687.1    | 2678.9    | 2561.7    | 4256.0    | 2553.2    | 2531.6    | 4754.8    | 2672.3    | 2554.4    |

*Note: Velocities in [m/s].*

The dynamic results show clear separation between compressional and shear components, with  $V_P$  values in the range 4113–4755 m/s and  $V_S$  values between 2498 and 2679 m/s. Representative velocity tuples are: for  $250^3$ ,  $(V_{P,x}, V_{S,y}, V_{S,z}) = (4639.9, 2660.7, 2507.0)$  m/s; and for  $400^3$ ,  $(4687.1, 2678.9, 2561.7)$  m/s. The gradual increase of all three components with model size suggests convergence toward representative carbonate elastic behavior. Compared with Berea and Fontainebleau, Grosmont presents lower shear rigidity and higher sensitivity to directional effects, which is expected for heterogeneous carbonates.

#### 5.4.3.3 RVE Convergence

The four sub-volumes of the Grosmont dataset also serve as a convergence study for the RVE size. The porosity decreases monotonically from 0.2651 ( $250^3$ ) to 0.2469 ( $400^3$ ), a variation of approximately 1.8 percentage points, which is significantly larger than that observed for Fontainebleau ( $< 0.2$  pp) and Berea ( $\sim 1.1$  pp). This larger porosity gradient reflects the vuggy, heterogeneous character of the carbonate microstructure and implies that a  $400^3$  sub-volume may not yet constitute a fully representative volume for this lithology. Nevertheless, both  $V_P$  and  $V_S$  show a monotonically increasing trend with domain size, suggesting gradual convergence. These results reinforce the importance of benchmark-specific calibration when transferring elastic parameters between lithologies.

#### 5.4.3.4 Comparison with the Andrä et al. Benchmark

Table 25 compares the elastic moduli obtained in this work against the Andrä carbonate benchmark for a  $400 \times 400 \times 400$  sub-volume with porosity  $\phi = 0.247$ . Our dynamic

FEM-CUDA result ( $K = 27.6$  GPa,  $G = 13.9$  GPa) shows good agreement with the L-S value ( $G = 14.0$  GPa) and the benchmark midrange ( $G_M = 13.1$  GPa) for the shear modulus. The bulk modulus, however, is higher than all Andrä reference values, which is consistent with the lower porosity of our sub-volumes ( $\phi \approx 0.247\text{--}0.265$ ) relative to those employed by the FEM-F solver in Andrä et al. [14]. The higher  $K$  values obtained for the Grosmont carbonate further highlight the sensitivity of elastic moduli to local pore-network heterogeneity in carbonate rocks.

Table 25: Grosmont: comparison of Bulk ( $K$ ) and Shear ( $G$ ) moduli between this work, the Andrä et al. [14] report, and static solver [19]. Andrä values correspond to a  $400^3$  subsample with porosity  $\phi = 0.247$ .

| Method                                 | $K$ [GPa] | $G$ [GPa] | Source      |
|----------------------------------------|-----------|-----------|-------------|
| CHFEM static (this work, $400^3$ )     | 26.1      | 12.7      | [19, 109]   |
| FEM-CUDA dynamic (this work, $400^3$ ) | 27.6      | 13.9      | this thesis |
| L-S                                    | 24.8      | 14.0      | [14]        |
| FEM-C                                  | 22.7      | 12.2      | [14]        |
| FEM-F                                  | 22.5      | 12.6      | [14]        |
| FDM-dynamic                            | 17.3      | 12.3      | [14]        |
| Midrange (Andrä)                       | 21.0      | 13.1      | [14]        |

#### 5.4.4 Computational Performance: Static versus Dynamic Execution Times

In addition to the elastic property comparison, the computational cost of the static CHFEM and dynamic FEM-CUDA solvers was measured for every sub-volume of the three benchmark rocks. It is important to note that the CHFEM solver does not solve the six load cases directly at the target sample resolution. An initial-guess stage is first performed on two coarser, uniformly downsampled representations of the sample, typically at one-quarter and one-half of the full mesh resolution (e.g.,  $100^3$  and  $200^3$  elements for a  $400^3$  sub-volume, or  $12^3$  and  $25^3$  elements for a  $50^3$  sub-volume). At each of these coarser levels, the PCG solver is run independently for the six loading directions ( $X$ ,  $Y$ ,  $Z$ , and the  $XY$ ,  $XZ$ ,  $YZ$  shear cases) until the prescribed residual tolerance is reached, and the resulting displacement fields are used to construct the initial guess for the subsequent, finer level. The reported static time therefore includes the two initial-guess stages plus the six full-resolution load cases. The dynamic time corresponds to the total wall-clock time required to run the complete set of directional simulations ( $V_P$ ,  $V_{S1}$ ,  $V_{S2}$ , for the

three axes) needed to obtain the homogenized velocities for each sample. The results are summarized in Table 26.

Table 26: Comparison of static (CHFEM) and dynamic (FEM-CUDA) total execution times for the Fontainebleau, Berea, and Grosmont benchmark samples. The static time corresponds to the largest value measured across the tested execution environments.

| Rock          | Sample size                 | Static time [s] | Dynamic time [s] |
|---------------|-----------------------------|-----------------|------------------|
| Fontainebleau | $150^3$                     | 106.27          | 698.65           |
|               | $200^3$                     | 273.43          | 2256.50          |
|               | $250^3$                     | 713.76          | 4233.05          |
|               | $288 \times 288 \times 300$ | 1279.15         | 6692.20          |
| Berea         | $250^3$                     | 1013.15         | 2259.29          |
|               | $300^3$                     | 2040.36         | 3499.99          |
|               | $350^3$                     | 4859.06         | 6781.78          |
|               | $400^3$                     | 2131.79         | 10833.39         |
| Grosmont      | $250^3$                     | 587.60          | 2903.63          |
|               | $300^3$                     | 1148.20         | 5644.52          |
|               | $350^3$                     | 1967.43         | 9881.61          |
|               | $400^3$                     | 3169.74         | 18494.14         |

# 6 Conclusions and Future Work

## 6.1 Conclusions

In conclusion, this work represents an important step forward in the development of an efficient time-domain image-based finite element solver in GPU for conducting UPT tests on large rock images obtained via  $\mu$ CT. The research was motivated by two complementary gaps in the DRP literature: (i) the absence of GPU-accelerated dynamic solvers capable of handling models with more than  $10^8$  degrees of freedom on domestic GPUs hardware; and (ii) the lack of a systematic comparison between static and dynamic numerical approaches applied to the same segmented image under Periodic Boundary Conditions. Both gaps have been addressed by this work, whose main contributions are summarized below.

1. A complete elastodynamic FEM formulation to image-based porous-media models was presented in Chapter 3. The framework combines (a) the Navier equations of motion for a linear elastic solid, (b) the assembly-free EbE strategy to avoid storage of the global stiffness matrix, (c) a lumped diagonal mass matrix whose inversion reduces to an element-wise reciprocal, and (d) the explicit Leap-Frog (Verlet) time integrator. Two degree-of-freedom mapping strategies were introduced and compared: Direct Node Mapping (DNM), which reconstructs connectivity analytically from each element index without any additional storage; and Active Node Mapping (ANM), which eliminates inactive pore-region DOFs and therefore reduces the system size by an amount proportional to the model porosity.
2. We developed and described five solver modes that share the same mathematical model but differ in architecture and data-access strategy: (1) serial C baseline with global DOFMAP (mode 1); (2) OpenMP shared-memory parallel CPU (mode 2); (3) CUDA GPU with precomputed DOFMAP (mode 3); (4) CUDA GPU with element-wise on-the-fly indexing that avoids storing the full connectivity map (mode 4); and (5) serial CPU element-wise reference (mode 5). All GPU main stages

are executed on-device, so the only host-device transfers occur at initialization and finalization, avoiding the PCIe bottleneck that would arise if displacement vectors were returned to the host at each time step. The element-wise CUDA kernel (mode 4) is, to the authors' knowledge, the first implementation of the assembly-free EbE product for dynamic image-based FEM on domestic GPU.

3. Benchmark validation on three-dimensional rock images. Section 5.4 validated the dynamic FEM-CUDA solver against the community benchmark of Andrä et al. [14] using the same segmented  $\mu$ CT images. Across all three lithologies, the FEM-CUDA results fall within the inter-method variability range (3%–18% of midrange moduli) documented by Andrä et al. [14]: Fontainebleau sandstone, Berea sandstone and Grosmont carbonate.

Taken together, these results demonstrate that the FEM-CUDA solver produces physically consistent elastic-property estimates across lithologies ranging from sandstones to complex vuggy carbonates.

## 6.2 Future Work

The ongoing phase of this research aims to achieve the following objectives:

1. Complete GPU performance benchmarking by report simulations on models exceeding  $10^8$  DOFs. The NVIDIA Nsight profiling should be used to identify and eliminate remaining performance bottlenecks (e.g., atomic contention in the EbE kernel, memory-access patterns in the history kernels).
2. Conduct 3D digital rock simulations utilizing the dynamic FEM-CUDA solver and subsequently compare the outcomes with new experimental laboratory UPT measurements performed on the corresponding physical core plugs. This comparison with new sandstone and carbonate samples will provide definitive evidence of the solver's accuracy in realistic scenarios.
3. Extend the NbN stencil strategy of Lopes et al. [19] to the dynamic setting. The NbN approach eliminates the element-based loop entirely by computing the stiffness-displacement product at the node level, which may reduce atomic-operation contention and improve memory-access coalescing compared with the current EbE kernel. Beyond these performance gains, the node-level granularity of NbN also makes it

straightforward to assign a specialized update kernel to each node type, since every node is processed independently rather than through a shared element loop. This property is directly relevant to the pore-fluid-saturation extension where solid-matrix and fluid-filled pore nodes are governed by distinct constitutive physics.

4. Evaluate the computational efficiency by comparing simulation times on real rock samples using both static and dynamic FEM-CUDA solver, with implementations on both CPU and GPU platforms. This benchmarking will quantify the speedup achieved by GPU parallelism and identify remaining computational bottlenecks.
5. Reduce memory in large-scale simulations by avoiding explicit storage of the mass matrix  $\mathbf{M}$  and computing its contribution on-the-fly during time integration. This matrix-free strategy is expected to improve scalability on GPU architectures and enable larger models within available device memory.
6. Reduce the required buffer-zone thickness by implementing attenuation in the boundary treatment (absorbing buffer zone), minimizing spurious reflections while preserving waveform quality at the sensors [89]. This improvement is expected to lower computational cost by decreasing the number of grid points outside the region of interest. One possibility would be the use of Perfectly Matched Layers (PML) [92, 91], a thin artificial layer that progressively attenuates outgoing waves through a coordinate stretching, or Boundary Element Methods (BEM), which reproduce the semi-infinite radiation condition exactly via the elastodynamic fundamental solution [115, 116]. BEM eliminates the buffer zone entirely, but at the cost of a full, non-symmetric system matrix and singular boundary integrals.
7. Implement automatic scaling of physical quantities (displacement, time, material moduli) to improve numerical conditioning and reduce round-off errors in single-precision arithmetic, following the guidelines of Bathe [69] and Cook et al. [72]. This is particularly important for rocks with extreme property contrasts.
8. Incorporate pore-fluid effects into the wave-propagation model to enable simulation of fluid-saturated rocks under the Gassmann framework. A key implementation requirement is the discretization of the viscous skin depth with at least three grid points per wavelength, as established by Saenger et al. [107]. This extension will allow the computation of frequency-dependent velocity dispersion and attenuation at the pore scale, paving the way for direct comparison with laboratory measurements on saturated samples.

## 6.3 Final Remarks

The Ultrasonic Pulse Transmission framework combining image-based FEM, the Leap-Frog time integrator, assembly-free Element-by-Element parallelism, and GPU acceleration via CUDA developed in this thesis represents a meaningful step toward making dynamic Digital Rock Physics accessible on domestic GPUs hardware. The validation against the Andrä benchmark confirms that the FEM-CUDA solver is physically credible and consistent with peer-reviewed reference methods. The comparison between static and dynamic approaches on the same digital images provides a strong relationship between the two computational paradigms in the context of image-based rock-physics simulation. The identified limitations defined during this research provide a clear roadmap for the continuation of the GPU performance benchmark using experimental UPT. The software infrastructure developed during this thesis, including the five-solver suite and the graphical user interface (Figure 16). The GUI is designed to support both academic research and practical applications in oil and gas exploration.



# REFERENCES

- [1] *Ultrasonic Pulsed-Beam Transmission And Reflection Methods For Measuring Rock Properties - Some Theoretical And Experimental Results*. Vol. All Days. SPWLA Annual Logging Symposium. SPWLA. May 1965, SPWLA-1965–A.
- [2] Gary Mavko, Tapan Mukerji, and Jack Dvorkin. *The rock physics handbook*. Cambridge University press, 2020.
- [3] Dean S. Oliver et al. “4D seismic history matching”. In: *Journal of Petroleum Science and Engineering* 207 (2021), p. 109119. ISSN: 0920-4105. DOI: 10.1016/j.petrol.2021.109119.
- [4] ASTM. *Annual Book of ASTM Standards, 1993 Vol. 04.08: Soil & Rock; Dimension Stone; Geosynthetics*. Philadelphia: American Society for Testing & Materials, 1984.
- [5] ISRM. *Suggested Methods for the Quantitative Description of Discontinuities in Rock Masses. Rock Characterisation, Testing and Monitoring*. Oxford: Pergamon Press, 1981.
- [6] Brian P Flannery et al. “Three-Dimensional X-Ray Microtomography”. In: *Science* 237.4821 (1987), pp. 1439–1444. ISSN: 0036-8075. DOI: 10.1126/science.237.4821.1439.
- [7] Erik H. Saenger. “Numerical methods to determine effective elastic properties”. In: *International Journal of Engineering Science* 46.6 (2008). Special Issue: Micromechanics of Materials, pp. 598–605. ISSN: 0020-7225. DOI: 10.1016/j.ijengsci.2008.01.005.
- [8] Erik H. Saenger, Oliver S. Krüger, and Serge A. Shapiro. “Effective elastic properties of randomly fractured soils: 3D numerical experiments”. In: *Geophysical Prospecting* 52.3 (2004), pp. 183–195. ISSN: 1365-2478. DOI: 10.1111/j.1365-2478.2004.00407.x.
- [9] Erik H. Saenger, Oliver S. Krüger, and Serge A. Shapiro. “Numerical considerations of fluid effects on wave propagation: Influence of the tortuosity”. In: *Geophysical Research Letters* 31.21 (2004). ISSN: 0094-8276. DOI: 10.1029/2004GL020970.

- [10] Erik H. Saenger et al. “Finite-difference modeling of wave propagation on microscale: a snapshot of the work in progress”. In: *Geophysics* 72.5 (2007), SM293–SM300. ISSN: 0016-8033. DOI: 10.1190/1.2753552.
- [11] E. H. Saenger, S. A. Shapiro, and Y. Keehm. “Seismic effects of viscous Biot-coupling: Finite difference simulations on micro-scale”. In: *Geophysical Research Letters* 32.14 (2005). ISSN: 0094-8276. DOI: 10.1029/2005GL023222.
- [12] Erik H. Saenger, Oliver S. Krüger, and Serge A. Shapiro. “Effective elastic properties of fractured rocks: Dynamic vs. static considerations”. In: *SEG Technical Program Expanded Abstracts 2006*. Society of Exploration Geophysicists, 2006, pp. 1948–1952. DOI: 10.1190/1.2369912.
- [13] Wei Zhu and Rui Shan. “Digital core based transmitted ultrasonic wave simulation and velocity accuracy analysis”. In: *Applied Geophysics* 13.2 (2016), pp. 375–381. ISSN: 1993-0658. DOI: 10.1007/s11770-016-0550-4.
- [14] Heiko Andrä et al. “Digital rock physics benchmarks—part II: Computing effective properties”. In: *Computers & Geosciences* 50 (2013), pp. 33–43. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2012.09.008.
- [15] Peter Arbenz et al. “A scalable multi-level preconditioner for matrix-free  $\mu$ -finite element analysis of human bone structures”. In: *International Journal for Numerical Methods in Engineering* 73.7 (2008), pp. 927–947. DOI: 10.1002/nme.2101.
- [16] Rafael S. Vianna et al. “Computing Effective Permeability of Porous Media with FEM and Micro-CT: An Educational Approach”. In: *Fluids* 5.1 (2020). ISSN: 2311-5521. DOI: 10.3390/fluids5010016.
- [17] Pedro Cortez Fetter Lopes et al. “A GPU implementation of the PCG method for large-scale image-based finite element analysis in heterogeneous periodic media”. In: *Computer Methods in Applied Mechanics and Engineering* 399 (2022), p. 115276. ISSN: 0045-7825. DOI: 10.1016/j.cma.2022.115276.
- [18] Luiz F Bez et al. “An Image-Based Explicit Matrix-Free FEM Implementation with Lumped Mass for NMR Simulations”. In: *Transport in Porous Media* 147.1 (2023), pp. 35–57. ISSN: 1573-1634. DOI: 10.1007/s11242-022-01894-1.
- [19] Pedro C.F. Lopes et al. “Simulation toolkit for digital material characterization of large image-based microstructures”. In: *Computational Materials Science* 219 (2023), p. 112021. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2023.112021.

- [20] Lucas Prata Ferreira et al. “An image-based numerical homogenization strategy for the characterization of viscoelastic composites”. In: *International Journal of Solids and Structures* 267 (2023), p. 112142. ISSN: 0020-7683. DOI: 10.1016/j.ijsolstr.2023.112142.
- [21] Pedro Cortez Fetter Lopes et al. “Enabling FEM-based absolute permeability estimation in giga-voxel porous media with a single GPU”. In: *Computer Methods in Applied Mechanics and Engineering* 434 (2025), p. 117559.
- [22] Titly Farhana Faisal et al. “Direct scale comparison of numerical linear elastic moduli with acoustic experiments for carbonate rock X-ray CT scanned at multi-resolutions”. In: *Journal of Petroleum Science and Engineering* 152 (2017), pp. 653–663. ISSN: 0920-4105. DOI: 10.1016/j.petrol.2017.01.025.
- [23] Federico Bucher, Fabio I Zyserman, and Leonardo B Monachesi. “Seismic wave propagation in coupled fluid and porous media: A finite element approach”. In: *Geophysical Prospecting* 72.8 (2024), pp. 2919–2941. DOI: <https://doi.org/10.1111/1365-2478.13562>.
- [24] Bruna Campos and Robert Gracie. “A Dynamic Three-Field Finite Element Model for Wave Propagation in Linear Elastic Porous Media”. In: *International Journal for Numerical and Analytical Methods in Geomechanics* 49.4 (2025), pp. 1139–1157. DOI: <https://doi.org/10.1002/nag.3916>.
- [25] Tsuyoshi Ichimura et al. “Fast and power efficient GPU-based explicit elastic wave propagation analysis by low-ordered orthogonal voxel finite element with INT8 Tensor Cores”. In: *Journal of Computational Science* 91 (2025), p. 102659. ISSN: 1877-7503. DOI: <https://doi.org/10.1016/j.jocs.2025.102659>.
- [26] Gregor P. Eberli et al. “Factors controlling elastic properties in carbonate sediments and rocks”. In: *The Leading Edge* 22.7 (2003), pp. 654–660. ISSN: 1070-485X. DOI: 10.1190/1.1599691.
- [27] Z. Hashin. “Analysis of Composite Materials—A Survey”. In: *Journal of Applied Mechanics* 50.3 (Sept. 1983), pp. 481–505. ISSN: 0021-8936. DOI: 10.1115/1.3167081.
- [28] S. Torquato. “Random Heterogeneous Media: Microstructure and Improved Bounds on Effective Properties”. In: *Applied Mechanics Reviews* 44.2 (Feb. 1991), pp. 37–76. ISSN: 0003-6900. DOI: 10.1115/1.3119494.

- [29] M.S. King. “Static and dynamic elastic properties of rocks from the Canadian shield”. In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 20.5 (1983), pp. 237–241. ISSN: 0148-9062. DOI: 10.1016/0148-9062(83)90004-9.
- [30] K. Alexandrov B. Belikov and T. Rysova. *Uprugie svoïstva porodoobrazuyushchikh mineralov i gornyykh porod*. Nauka, 1970.
- [31] Amos Nur and Zhijing Wang. *Seismic and acoustic velocities in reservoir rocks*. 10. Society of Exploration Geophysicists, 1989.
- [32] Vicente Brotons et al. “Improved correlation between the static and dynamic elastic modulus of different types of rocks”. In: *Materials and structures* 49 (2016), pp. 3021–3037. ISSN: 1871-6873. DOI: 10.1617/s11527-015-0702-7.
- [33] E.A. Eissa and A. Kazi. “Relation between static and dynamic Young’s moduli of rocks”. In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 25.6 (1988), pp. 479–482. ISSN: 0148-9062. DOI: 10.1016/0148-9062(88)90987-4.
- [34] D. M. McCann and D. C. Entwisle. “Determination of Young’s modulus of the rock mass from geophysical well logs”. In: *Geological Society, London, Special Publications* 65.1 (1992), pp. 317–325. DOI: 10.1144/GSL.SP.1992.065.01.24.
- [35] B Christaras, F Auger, and E Mosse. “Determination of the moduli of elasticity of rocks. Comparison of the ultrasonic velocity and mechanical resonance frequency methods with direct static methods”. In: *Materials and Structures* 27 (1994), pp. 222–228. ISSN: 1871-6873. DOI: 10.1007/BF02473036.
- [36] Vicente Brotons et al. “Relationship between static and dynamic elastic modulus of calcarenite heated at different temperatures: the San Julián’s stone”. In: *Bulletin of Engineering Geology and the Environment* 73 (2014), pp. 791–799. ISSN: 1435-9537. DOI: 10.1007/s10064-014-0583-y.
- [37] Lewis L Lacy. “Dynamic rock mechanics testing for optimized fracture designs”. In: *SPE annual technical conference and exhibition*. OnePetro. 1997.
- [38] Ali Reza Najibi et al. “Empirical relations between strength and static and dynamic elastic properties of Asmari and Sarvak limestones, two main oil reservoirs in Iran”. In: *Journal of Petroleum Science and Engineering* 126 (2015), pp. 78–82. ISSN: 0920-4105. DOI: 10.1016/j.petrol.2014.12.010.

- [39] W.L. b Heerden. “General relations between static and dynamic moduli of rocks”. In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 24.6 (1987), pp. 381–385. ISSN: 0148-9062. DOI: 10.1016/0148-9062(87)92262-5.
- [40] *Calibrated Wireline Mechanical Rock Properties Model for Predicting and Preventing Wellbore Collapse and Sanding*. Vol. All Days. SPE European Formation Damage Conference and Exhibition. May 2003, SPE-82236-MS. DOI: 10.2118/82236-MS.
- [41] Seyed Morteza Davarpanah, Peter Ván, and Balázs Vásárhelyi. “Investigation of the relationship between dynamic and static deformation moduli of rocks”. In: *Geomechanics and Geophysics for Geo-Energy and Geo-Resources* 6.1 (2020), p. 29. ISSN: 2363-8427. DOI: 10.1007/s40948-020-00155-z.
- [42] F Rummel and WL Van Heerden. “Suggested methods for determining sound velocity”. In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 15.2 (1978), pp. 53–58. ISSN: 0148-9062. DOI: 10.1016/0148-9062(78)91678-9.
- [43] Martin Gordon Culshaw. “Ulusay, R (ed.), 2015. The ISRM suggested methods for rock characterization, testing and monitoring: 2007–2014”. In: *Bulletin of Engineering Geology and the Environment* 74.4 (Nov. 2015), pp. 1499–1500. ISSN: 1435-9537. DOI: 10.1007/s10064-015-0780-3.
- [44] Francis Birch. “The velocity of compressional waves in rocks to 10 kilobars: 1.” In: *Journal of Geophysical Research (1896-1977)* 65.4 (1960), pp. 1083–1102. DOI: 10.1029/JZ065i004p01083.
- [45] D. U. Deere and R. P. Miller. *Engineering classification and index properties for intact rock*. Springfield: National Technical Information Service, 1966.
- [46] Francis Birch. “The velocity of compressional waves in rocks to 10 kilobars: 2.” In: *Journal of Geophysical Research (1896-1977)* 66.7 (1961), pp. 2199–2224. DOI: 10.1029/JZ066i007p02199.
- [47] Masayasu Inoue and Michito Ohomi. “Relation between uniaxial compressive strength and elastic wave velocity of soft rock”. In: vol. All Days. ISRM International Symposium. ISRM. Sept. 1981, ISRM-IS-1981–002.
- [48] W. C. Van Buskirk, S. C. Cowin, and R. N. Ward. “Ultrasonic Measurement of Orthotropic Elastic Constants of Bovine Femoral Bone”. In: *Journal of Biomechanical Engineering* 103.2 (May 1981), pp. 67–72. ISSN: 0148-0731. DOI: 10.1115/1.3138262.

- [49] Guilhem Barruol and Hartmut Kern. “Seismic anisotropy and shear-wave splitting in lower-crustal and upper-mantle rocks from the Ivrea Zone—experimental and calculated data”. In: *Physics of the Earth and Planetary Interiors* 95.3-4 (1996), pp. 175–194. ISSN: 0031-9201. DOI: 10.1016/0031-9201(95)03124-3.
- [50] H Kern. “P- and S-wave anisotropy and shear-wave splitting at pressure and temperature in possible mantle rocks and their relation to the rock fabric”. In: *Physics of the Earth and Planetary Interiors* 78.3-4 (1993), pp. 245–256. ISSN: 0031-9201. DOI: 10.1016/0031-9201(93)90159-7.
- [51] Vladislav Babuška and Zdeněk Pros. “Velocity anisotropy in granodiorite and quartzite due to the distribution of microcracks”. In: *Geophysical Journal International* 76.1 (Jan. 1984), pp. 121–127. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.1984.tb05027.x.
- [52] P Gaviglio. “Longitudinal waves propagation in a limestone: the relationship between velocity and density”. In: *Rock Mechanics and Rock Engineering* 22.4 (1989), pp. 299–306. ISSN: 1434-453X. DOI: 10.1007/BF01262285.
- [53] Sair Kahraman. “Evaluation of simple methods for assessing the uniaxial compressive strength of rock”. In: *International Journal of Rock Mechanics and Mining Sciences* 38.7 (2001), pp. 981–994. ISSN: 1365-1609. DOI: 10.1016/S1365-1609(01)00039-9.
- [54] E Yasar and Y Erdogan. “Correlating sound velocity with the density, compressive strength and Young’s modulus of carbonate rocks”. In: *International Journal of Rock Mechanics and Mining Sciences* 41.5 (2004), pp. 871–875. ISSN: 1365-1609. DOI: 10.1016/j.ijrmms.2004.01.012.
- [55] Sair Kahraman. “Estimating the direct P-wave velocity value of intact rock from indirect laboratory measurements”. In: *International Journal of Rock Mechanics and Mining Sciences* 39.1 (2002), pp. 101–104. ISSN: 1365-1609. DOI: 10.1016/S1365-1609(02)00005-9.
- [56] Celal Karpuz and A. Gunhan Pasamehmetoglu. “Field characterisation of weathered Ankara andesites”. In: *Engineering Geology* 46.1 (1997), pp. 1–17. ISSN: 0013-7952. DOI: 10.1016/S0013-7952(96)00002-6.
- [57] Z. A. Moradian and M. Behnia. “Predicting the Uniaxial Compressive Strength and Static Young’s Modulus of Intact Sedimentary Rocks Using the Ultrasonic Test”. In: *International Journal of Geomechanics* 9.1 (2009), pp. 14–19. DOI: 10.1061/(ASCE)1532-3641(2009)9:1(14).

- [58] Lucas Pimienta et al. “Pressure-dependent elastic and transport properties of porous and permeable rocks: Microstructural control”. In: *Journal of Geophysical Research: Solid Earth* 122.11 (2017), pp. 8952–8968. ISSN: 2169-9313. DOI: 10.1002/2017JB014464.
- [59] JCM Teo et al. “Determination of bone trabeculae modulus—an ultrasonic scanning and microct ( $\mu$ CT) imaging combination approach”. In: *Experimental mechanics* 46 (2006), pp. 453–461. ISSN: 1741-2765. DOI: 10.1007/s11340-006-8444-7.
- [60] Hongyan Yu et al. “X-ray micro-computed tomography and ultrasonic velocity analysis of fractured shale as a function of effective stress”. In: *Marine and Petroleum Geology* 110 (2019), pp. 472–482. ISSN: 0264-8172. DOI: 10.1016/j.marpetgeo.2019.07.015.
- [61] Junwei Ma et al. “Experimental Investigation of the Physical Properties and Microstructure of Slate under Wetting and Drying Cycles Using Micro-CT and Ultrasonic Wave Velocity Tests”. In: *Sensors* 20.17 (2020). ISSN: 1424-8220. DOI: 10.3390/s20174853.
- [62] Jiabin Liang et al. “High-Precision Tracking of Sandstone Deformation From Micro-CT Images”. In: *Journal of Geophysical Research: Solid Earth* 126.9 (2021), e2021JB022283. ISSN: 2169-9313. DOI: 10.1029/2021JB022283.
- [63] William D Carlson, Cambria Denison, and Richard A Ketcham. “High-resolution X-ray computed tomography as a tool for visualization and quantitative analysis of igneous textures in three dimensions”. In: *Visual Geosciences* 4 (2000), pp. 1–14. ISSN: 1610-2924. DOI: 10.1007/s10069-999-0003-9.
- [64] V. Cnudde et al. “Recent progress in X-ray CT as a geosciences tool”. In: *Applied Geochemistry* 21.5 (2006). Frontiers in Analytical Geochemistry—An IGC 2004 Perspective, pp. 826–832. ISSN: 0883-2927. DOI: 10.1016/j.apgeochem.2006.02.010.
- [65] Heiko Andrä et al. “Digital rock physics benchmarks—Part I: Imaging and segmentation”. In: *Computers & Geosciences* 50 (2013). Benchmark problems, datasets and methodologies for the computational geosciences, pp. 25–32. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2012.09.005.
- [66] A.P. Roberts and E.J. Garboczi. “Elastic properties of model random three-dimensional open-cell solids”. In: *Journal of the Mechanics and Physics of Solids* 50.1 (2002), pp. 33–55. ISSN: 0022-5096. DOI: 10.1016/S0022-5096(01)00056-4.

- [67] Christoph H. Arns et al. “Computation of linear elastic properties from microtomographic images: Methodology and agreement between theory and experiment”. In: *GEOPHYSICS* 67.5 (2002), pp. 1396–1405. DOI: 10.1190/1.1512785.
- [68] Erik H. Saenger and Thomas Bohlen. “Finite-difference modeling of viscoelastic and anisotropic wave propagation using the rotated staggered grid”. In: *GEOPHYSICS* 69.2 (Jan. 2004), pp. 583–591. ISSN: 0016-8033. DOI: 10.1190/1.1707078.
- [69] Klaus-Jürgen Bathe. *Finite Element Procedures*. Ed. Prentice Hall, 1996.
- [70] O.C. Zienkiewicz, R.L. Taylor, and S. Govindjee. *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann, 2024. ISBN: 9780443160455. URL: <https://books.google.com.br/books?id=ojwTEQAAQBAJ>.
- [71] Roland W Lewis and Bernard A Schrefler. *The finite element method in the static and dynamic deformation and consolidation of porous media*. John Wiley & Sons, 1999.
- [72] Robert D. Cook et al. *Concepts and Applications of Finite Element Analysis*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.
- [73] Pedro CF Lopes et al. “A vectorized assembly-free FEM solver for image-based numerical homogenization”. In: *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 44.8 (2022), p. 343.
- [74] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2003.
- [75] Robert Mullen and Ted Belytschko. “Dispersion analysis properties of finite element semidiscretizations of the two-dimensional wave equation”. In: *International Journal for Numerical Methods in Engineering* 18.1 (1982), pp. 11–29. DOI: 10.1002/nme.1620180103.
- [76] Kurt J. Marfurt. “Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations”. In: *Geophysics* 49.5 (1984), pp. 533–549. DOI: 10.1190/1.1441689.
- [77] Loup Verlet. “Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules”. In: *Physical Review* 159.1 (1 July 1967), pp. 98–103. ISSN: 0031-899X. DOI: 10.1103/PhysRev.159.98.



- [78] Catherine O’Sullivan and Jonathan D. Bray. “Selecting a suitable time step for discrete element simulations that use the central difference time integration scheme”. In: *Engineering Computations (Swansea, Wales)* 21.2-4 (2004), pp. 278–303. ISSN: 0264-4401. DOI: 10.1108/02644400410519794.
- [79] Richard Courant, Kurt Friedrichs, and Hans Lewy. “Über die partiellen Differenzengleichungen der mathematischen Physik”. In: *Mathematische annalen* 100.1 (1928), pp. 32–74. ISSN: 1432-1807. DOI: 10.1007/BF01448839.
- [80] Thomas Bohlen and Erik H. Saenger. “Accuracy of heterogeneous staggered-grid finite-difference modeling of Rayleigh waves”. In: *GEOPHYSICS* 71.4 (Aug. 2006), T109–T115. ISSN: 0016-8033. DOI: 10.1190/1.2213051. URL: <https://doi.org/10.1190/1.2213051>.
- [81] C.T. Sun and R.S. Vaidya. “Prediction of composite properties from a representative volume element”. In: *Composites Science and Technology* 56.2 (1996), pp. 171–179. ISSN: 0266-3538. DOI: 10.1016/0266-3538(95)00141-7.
- [82] T. Kanit et al. “Determination of the size of the representative volume element for random composites: statistical and numerical approach”. In: *International Journal of Solids and Structures* 40.13 (2003), pp. 3647–3679. ISSN: 0020-7683. DOI: 10.1016/S0020-7683(03)00143-4.
- [83] Kenjiro Terada et al. “Simulation of the multi-scale convergence in computational homogenization approaches”. In: *International Journal of Solids and Structures* 37.16 (2000), pp. 2285–2311. ISSN: 0020-7683. DOI: 10.1016/S0020-7683(98)00341-2.
- [84] F. Larsson et al. “Computational homogenization based on a weak format of micro-periodicity for RVE-problems”. In: *Computer Methods in Applied Mechanics and Engineering* 200.1 (2011), pp. 11–26. ISSN: 0045-7825. DOI: 10.1016/j.cma.2010.06.023.
- [85] V.-D. Nguyen et al. “Imposing periodic boundary condition on arbitrary meshes by polynomial interpolation”. In: *Computational Materials Science* 55 (2012), pp. 390–406. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2011.10.017.
- [86] M. G. D. Geers, V. G. Kouznetsova, and W. A. M. Brekelmans. “Multi-scale computational homogenization: Trends and challenges”. In: *Journal of Computational and Applied Mathematics* 234.7 (2010), pp. 2175–2182. DOI: 10.1016/j.cam.2009.08.077.

- [87] E.H. Saenger et al. “Digital material laboratory: Wave propagation effects in open-cell aluminium foams”. In: *International Journal of Engineering Science* 58 (2012), pp. 115–123. ISSN: 0020-7225. DOI: 10.1016/j.ijengsci.2012.03.030.
- [88] Wei Zhu and Rui Shan. “Digital core based transmitted ultrasonic wave simulation and velocity accuracy analysis”. In: *Applied Geophysics* 13 (2 June 2016), pp. 375–181. ISSN: 1993-0658. DOI: 10.1007/s11770-016-0550-4.
- [89] Shlomo Ta’asan and Douglas Nark. “An absorbing buffer zone technique for acoustic wave propagation”. In: *33rd Aerospace Sciences Meeting and Exhibit*. 1995, p. 164. DOI: 10.2514/6.1995-164.
- [90] Doron Kishoni and Shlomo Ta’asan. “Non-reflecting regions for finite difference methods in modeling of elastic wave propagation in plates”. In: *Review of Progress in Quantitative Nondestructive Evaluation* 13 (1994), pp. 195–195. ISSN: 0743-0760.
- [91] Arash Fathi, Babak Poursartip, and Loukas F. Kallivokas. “Time-domain hybrid formulations for wave simulations in three-dimensional PML-truncated heterogeneous media”. In: *International Journal for Numerical Methods in Engineering* 101.3 (2015), pp. 165–198. ISSN: 0029-5981. DOI: 10.1002/nme.4780.
- [92] Jean-Pierre Berenger. “A perfectly matched layer for the absorption of electromagnetic waves”. In: *Journal of Computational Physics* 114.2 (1994), pp. 185–200. ISSN: 0021-9991. DOI: 10.1006/jcph.1994.1159.
- [93] Christopher Bauer et al. “Gaussian pulses”. In: *Journal of Magnetic Resonance (1969)* 58.3 (1984), pp. 442–457. ISSN: 0022-2364. DOI: 10.1016/0022-2364(84)90148-3.
- [94] Peter Huthwaite. “Accelerated finite element elastodynamic simulations using the GPU”. In: *Journal of Computational Physics* 257 (2014), pp. 687–707. ISSN: 0021-9991. DOI: <https://doi.org/10.1016/j.jcp.2013.10.017>. URL: <https://www.sciencedirect.com/science/article/pii/S0021999113006931>.
- [95] John D Owens et al. “A survey of general-purpose computation on graphics hardware”. In: *Computer graphics forum* 26.1 (2007), pp. 80–113. DOI: 10.1111/j.1467-8659.2007.01012.x.
- [96] Design Guide. “Cuda c++ programming guide”. In: *NVIDIA, April* (2026). URL: <https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#features-and-technical-specifications>.

- [97] Jeff Bezanson et al. “Julia: A fresh approach to numerical computing”. In: *SIAM Review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671.
- [98] Thomas J.R. Hughes, Itzhak Levit, and James Winget. “An element-by-element solution algorithm for problems of structural and solid mechanics”. In: *Computer Methods in Applied Mechanics and Engineering* 36.2 (1983), pp. 241–254. ISSN: 0045-7825. DOI: 10.1016/0045-7825(83)90115-9.
- [99] Jocelyne Erhel, Alice Traynard, and Marina Vidrascu. “An element-by-element preconditioned conjugate gradient method implemented on a vector computer”. In: *Parallel Computing* 17.9 (1991), pp. 1051–1065. ISSN: 0167-8191. DOI: 10.1016/S0167-8191(05)80050-X.
- [100] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge university press, 2010.
- [101] William Weaver Jr, Stephen P Timoshenko, and Donovan Harold Young. *Vibration problems in engineering*. John Wiley & Sons, 1991.
- [102] Guilherme Barros et al. “DEM-BEM coupling in time domain for one-dimensional wave propagation”. In: *Engineering Analysis with Boundary Elements* 135 (2022), pp. 26–37. ISSN: 0955-7997. DOI: 10.1016/j.enganabound.2021.10.017.
- [103] Lloyd N. Trefethen. “Group velocity in finite difference schemes”. In: *SIAM Review* 24.2 (1982), pp. 113–136. DOI: 10.1137/1024038.
- [104] Mark A Dablain. “The application of high-order differencing to the scalar wave equation”. In: *Geophysics* 51.1 (1986), pp. 54–66. DOI: 10.1190/1.1442040.
- [105] Claudio Madonna, Bjarne S.G. Almqvist, and Erik H. Saenger. “Digital rock physics: Numerical prediction of pressure-dependent ultrasonic velocities using micro-CT imaging”. In: *Geophysical Journal International* 189 (3 2012), pp. 1475–1482. ISSN: 0956540X. DOI: 10.1111/j.1365-246X.2012.05437.x.
- [106] Edward Garboczi. *Finite Element and Finite Difference Programs for Computing the Linear Electric and Elastic Properties of Digital Images of Random Materials*. en. Dec. 1998. URL: [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=860168](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=860168).
- [107] Erik H Saenger et al. “Digital rock physics: Effect of fluid viscosity on effective elastic properties”. In: *Journal of Applied Geophysics* 74.4 (2011), pp. 236–241.

- [108] Hervé Moulinec and Pierre Suquet. “A numerical method for computing the overall response of nonlinear composites with complex microstructure”. In: *Computer methods in applied mechanics and engineering* 157.1-2 (1998), pp. 69–94.
- [109] Pedro Cortez Fetter Lopes. “Massively Parallel Implementations of the Preconditioned Conjugate Gradient Method Applied to Image-based Numerical Homogenization with the Assembly-free Finite Element Method”. MA thesis. Niteroi: The school of the thesis, 2021.
- [110] Shankar Subramanian et al. “Laboratory-based seismic attenuation in Fontainebleau sandstone: Evidence of squirt flow”. In: *Journal of Geophysical Research: Solid Earth* 120.11 (2015), pp. 7526–7535. DOI: <https://doi.org/10.1002/2015JB012290>. eprint: <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2015JB012290>. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015JB012290>.
- [111] Faisal Al Saadi, K Wolf, and CV Kruijsdijk. “Characterization of Fontainebleau sandstone: Quartz overgrowth and its impact on pore-throat framework”. In: *Journal of Petroleum & Environmental Biotechnology* 8.3 (2017), pp. 1–12.
- [112] Thierry Bourbie and Bernard Zinszner. “Hydraulic and acoustic properties as a function of porosity in Fontainebleau sandstone”. In: *Journal of Geophysical Research: Solid Earth* 90.B13 (1985), pp. 11524–11532.
- [113] Samuel Chapman et al. “Seismic attenuation in partially saturated Berea sandstone submitted to a range of confining pressures”. In: *Journal of Geophysical Research: Solid Earth* 121.3 (2016), pp. 1664–1676.
- [114] F Jerry Lucia. “Rock-fabric/ petrophysical classification of carbonate pore space for reservoir characterization”. In: *AAPG bulletin* 79.9 (1995), pp. 1275–1300.
- [115] Jose Dominguez. *Boundary elements in dynamics*. Wit Press, 1993.
- [116] Dimitri E. Beskos. “Boundary Element Methods in Dynamic Analysis: Part II (1986-1996)”. In: *Applied Mechanics Reviews* 50.3 (Mar. 1997), pp. 149–197. ISSN: 0003-6900. DOI: [10.1115/1.3101695](https://doi.org/10.1115/1.3101695). URL: <https://doi.org/10.1115/1.3101695>.

## APPENDIX A - Homogeneous Isotropic Materials

For isotropic linear elasticity, the constitutive Hooke's law (Equation 3.2) is fully defined by any two independent elastic moduli. In DRP, a common set of parameters includes Young's modulus  $E$ , Poisson's ratio  $\nu$ , bulk modulus  $K$ , shear modulus  $\mu$ , and the first Lamé parameter  $\lambda$  [2]. Table 27 summarizes the conversion formulas among these constants.

Table 27: Conversion formulas for homogeneous isotropic linear elastic material properties, adapted from Mavko, Mukerji, and Dvorkin [2].

|                    | $E$                                     | $\nu$                            | $K$                           | $\lambda$                      | $\mu$                          |
|--------------------|-----------------------------------------|----------------------------------|-------------------------------|--------------------------------|--------------------------------|
| $(E, \nu)$         | -                                       | -                                | $\frac{E}{3(1-2\nu)}$         | $\frac{E\nu}{(1+\nu)(1-2\nu)}$ | $\frac{E}{2(1+\nu)}$           |
| $(E, K)$           | -                                       | $\frac{3K-E}{6K}$                | -                             | $\frac{3K(3K-E)}{9K-E}$        | $\frac{3KE}{9K-E}$             |
| $(E, \mu)$         | -                                       | $\frac{E}{2\mu} - 1$             | $\frac{E\mu}{3(3\mu-E)}$      | $\frac{\mu(E-2\mu)}{3\mu-E}$   | -                              |
| $(\lambda, \mu)$   | $\mu \frac{3\lambda+2\mu}{\lambda+\mu}$ | $\frac{\lambda}{2(\lambda+\mu)}$ | $\lambda + \frac{2\mu}{3}$    | -                              | -                              |
| $(\lambda, \nu)^*$ | $\frac{\lambda(1+\nu)(1-2\nu)}{\nu}$    | -                                | $\frac{\lambda(1+\nu)}{3\nu}$ | -                              | $\frac{\lambda(1-2\nu)}{2\nu}$ |

\* **Note:** Cannot be used when  $\nu = 0 \Leftrightarrow \lambda = 0$

For convenience, three widely used identities are

$$K = \lambda + \frac{2}{3}\mu, \quad E = \mu \frac{3\lambda + 2\mu}{\lambda + \mu}, \quad \nu = \frac{\lambda}{2(\lambda + \mu)}. \quad (\text{A.1})$$

For mechanically stable isotropic media, the admissible range is

$$\mu > 0, \quad K > 0 \iff -1 < \nu < \frac{1}{2}. \quad (\text{A.2})$$

## APPENDIX B - Analytical Solutions for Local Image-Based 2D and 3D Finite Element Matrices

The stiffness matrices of the image-based 2D (Figure 44) and 3D (Figure 45) used in this work were analytically calculated using MATLAB's symbolic math in the study by [109]. The local stiffness matrices for finite elements assume that linear shape functions and isotropic materials. Consequently, the local stiffness matrix can be expressed as:

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}^T \mathbf{C} \mathbf{B} d\Omega, \quad (\text{B.1})$$

where  $K_e$  is the stiffness matrix of element  $e$ ,  $\Omega_e$  is the volume of element  $e$ ,  $\mathbf{B} = \partial \mathbf{N}$  is the strain-displacement matrix, and  $\mathbf{C}$  is the elasticity matrix for a homogeneous isotropic material. For 2D elasticity wave propagation,

$$\begin{aligned} c_1 &= \frac{1}{2} - \frac{2\nu}{3} & c_4 &= \frac{1}{8} \\ c_2 &= \frac{1}{8} - \frac{\nu}{2} & c_5 &= -\frac{1}{4} + \frac{\nu}{3}, \\ c_3 &= \frac{\nu}{6} & c_6 &= -\frac{1}{4} + \frac{\nu}{6} \end{aligned} \quad (\text{B.2})$$

$$K_e = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} c_1 & c_4 & c_6 & -c_2 & c_5 & -c_4 & c_3 & c_2 \\ c_4 & c_1 & c_2 & c_3 & -c_4 & c_5 & -c_2 & c_6 \\ c_6 & c_2 & c_1 & -c_4 & c_3 & -c_2 & c_5 & c_4 \\ -c_2 & c_3 & -c_4 & c_1 & c_2 & c_6 & c_4 & c_5 \\ c_5 & -c_4 & c_3 & c_2 & c_1 & c_4 & c_6 & -c_2 \\ -c_4 & c_5 & -c_2 & c_6 & c_4 & c_1 & c_2 & c_3 \\ c_3 & -c_2 & c_5 & c_4 & c_6 & c_2 & c_1 & -c_4 \\ c_2 & c_6 & c_4 & c_5 & -c_2 & c_3 & -c_4 & c_1 \end{bmatrix}, \quad (\text{B.3})$$

where  $c_1, c_2, c_3, c_4, c_5$ , and  $c_6$  are the coefficients from integral. For 3D elasticity wave propagation, the coefficients ( $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9$ , and  $c_{10}$ ) and the matrix are,

$$\begin{aligned}
 c_1 &= \frac{2}{9} - \frac{\nu}{3} & c_6 &= -\frac{1}{48} \\
 c_2 &= \frac{1}{24} - \frac{\nu}{6} & c_7 &= \frac{1}{18} - \frac{\nu}{12} \\
 c_3 &= -\frac{1}{18} & c_8 &= \frac{1}{36} + \frac{\nu}{12} \\
 c_4 &= -\frac{1}{24} & c_9 &= \frac{1}{48} - \frac{\nu}{12} \\
 c_5 &= -\frac{1}{36} & c_{10} &= \frac{5}{72} + \frac{\nu}{12}
 \end{aligned} \tag{B.4}$$





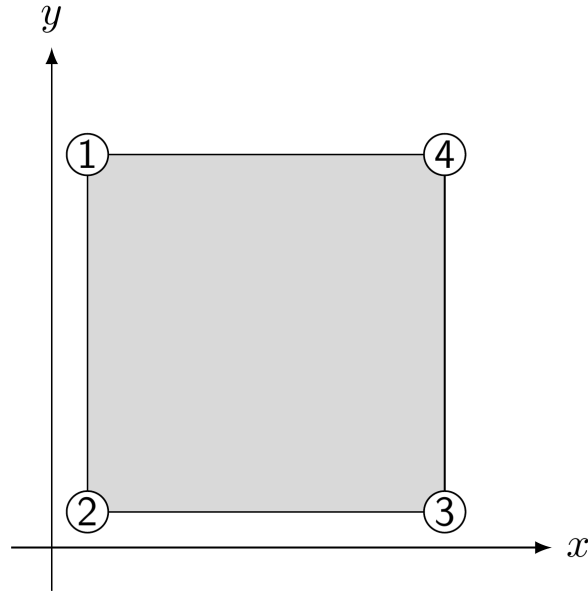


Figure 44: Four-node quadrilateral finite element (pixel) used in 2D image-based analysis. The element is defined by its corner nodes and linear shape functions for strain-displacement computations.

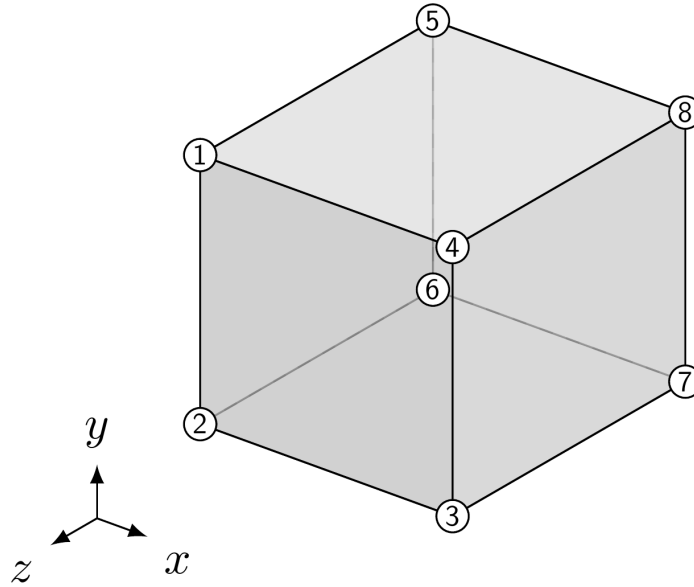


Figure 45: Eight-node hexahedral finite element (voxel) used in 3D image-based analysis. The element is defined by its eight corner nodes and linear shape functions for strain-displacement computations in three-dimensional elasticity problems.

## APPENDIX C - Gaussian Pulse

In this thesis, the source term used in the UPT simulations is a Gaussian pulse in time. This choice is motivated by its smoothness, compact energy concentration around a central instant, and low numerical dispersion when used in explicit time domain solvers [93]. The Gaussian curve, often referred to as the normal distribution, is a type of continuous probability distribution widely used in numerical simulations due to its beneficial properties. Additionally, since the curve is governed by two parameters, the mean and standard deviation, adjusting for the process of determining the optimal wavelength for each rock sample becomes user-friendly to manipulate. Mathematically, it is represented as:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (\text{C.1})$$

where  $\sigma$  is the standard deviation,  $\mu$  is the mean, and  $x$  is the variable. The Figure 46 shows the Gaussian curve as a function of maximum amplitude and standard deviation.

Another important property of the Gaussian curve used in this study is the Full Width at Half Maximum (FWHM), as shown in Figure 47. The FWHM is employed to determine the wavelength and frequency of the pulses used in rock samples. It's worth noting that the units of pulse width will be the same as those of the standard deviation, sigma. For a Gaussian pulse, the relation between FWHM and standard deviation is

$$\text{FWHM} = 2\sigma\sqrt{2\ln 2}. \quad (\text{C.2})$$

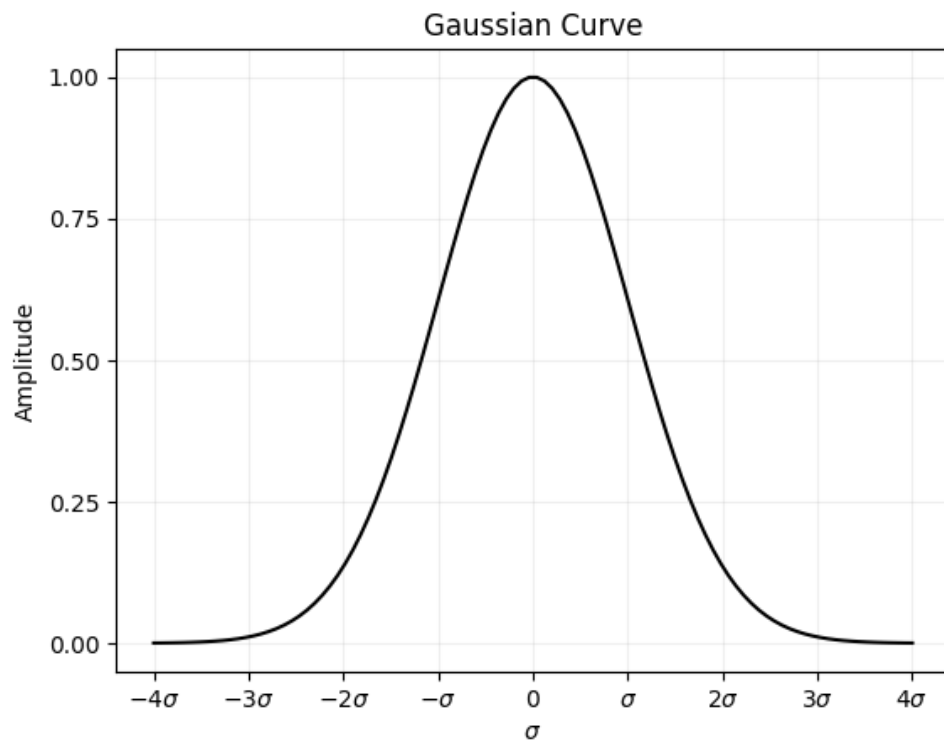


Figure 46: Gaussian pulse used as source-time function in the UPT simulations.

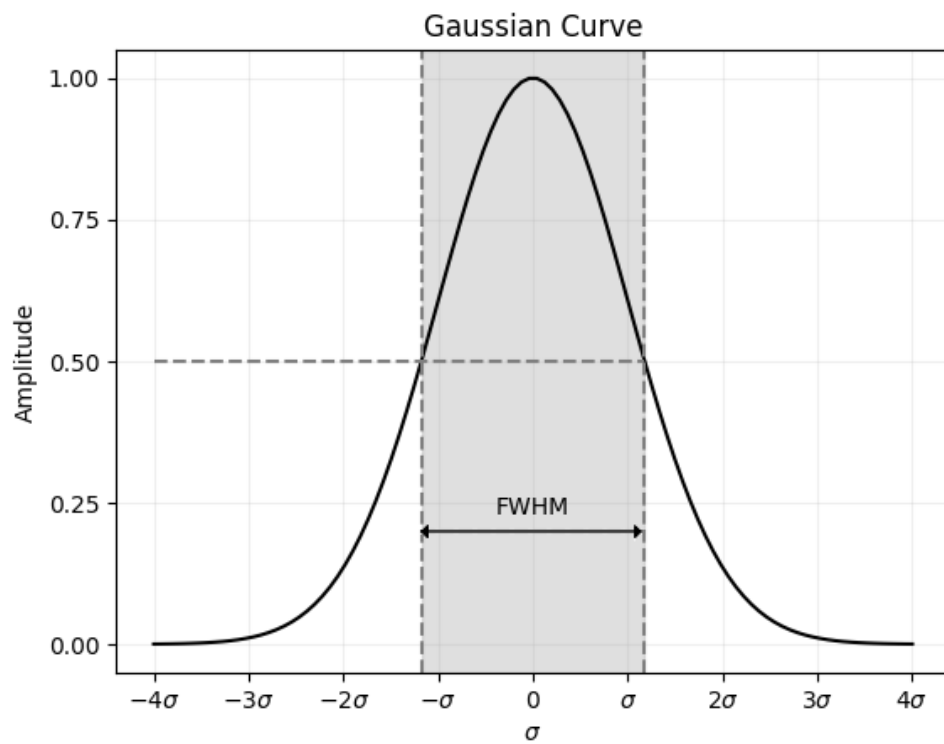


Figure 47: Relation between temporal standard deviation  $\sigma$  and full width at half maximum (FWHM).