Wave Equation: Space and Time Discretization, Energy Conservation, Convergence and Scalability

Bonalumi, Buttini, Nedina

February 22, 2026

Contents

Chapter 1

Space Discretization

1.1 Project Model and Physical Interpretation

Let $\Omega \subset \mathbb{R}^2$ and $T > 0$. We consider the wave equation:

$$\begin{cases} u_{tt} - \Delta u = f & \text{in } \Omega \times (0,T), \\ u = g & \text{on } \partial\Omega \times (0,T), \\ u(\cdot,0) = u_0, \quad u_t(\cdot,0) = u_1 & \text{in } \Omega. \end{cases} \tag{1.1}$$

The mathematical problem represents the prototype of hyperbolic partial differential equations. Physically, in a two-dimensional domain Ω, this equation models the vibrations of an elastic, homogeneous membrane under tension.
In this context, the variable $u(x,t)$ represents the vertical displacement of the membrane from its equilibrium plane at position $\mathbf{x} = (x,y)^T$ and time t. In this physical framework, the governing equation can be interpreted as a balance of forces. The second time derivative, $\frac{\partial^2 u}{\partial t^2}$, represents the inertial term corresponding to the acceleration of the membrane's mass, whereas the Laplacian operator $-\Delta u$ acts as the restoring force induced by elastic tension, typical of the wave equation. These internal forces balance the *source term* f, which denotes a distributed external vertical force density acting on the surface. Finally, the system evolution is uniquely determined by the *initial conditions*, where u_0 prescribes the initial configuration (deformation) and u_1 imposes the initial velocity field.
A defining feature of this class of equations is that information travels at a finite speed. By comparing our model with the general form of the wave equation

$$u_{tt} - c^2 \Delta u = f$$

we can identify the wave propagation speed as $c = 1$. This parameter c represents the physical speed at which mechanical disturbances and energy propagate across the domain Ω.

1.2 Weak Formulation and Matrix Form

We define the Sobolev space $H^1(\Omega)$

$$H^1(\Omega) = \left\{ v \in L^2(\Omega) : \frac{\partial v}{\partial x_i} \in L^2(\Omega),\ i = 1,2 \right\}.$$

For homogeneous Dirichlet conditions, the test space (and solution space) is

$$V := H_0^1(\Omega) = \{ v \in H^1(\Omega) : v|_{\partial\Omega} = 0 \}.$$

1.2.1 Weak Formulation and Semi-Discrete Matrix Form: Homogeneous Dirichlet Case

First consider the homogeneous case $u = 0$ on $\partial\Omega$. Take $v \in V$ and multiply (1.1) by v and integrate over Ω:

$$\int_\Omega u_{tt} v \, d\mathbf{x} - \int_\Omega \Delta u \, v \, d\mathbf{x} = \int_\Omega f v \, d\mathbf{x}.$$

Using Green's identity,

$$-\int_\Omega \Delta u \, v \, d\mathbf{x} = \int_\Omega \nabla u \cdot \nabla v \, d\mathbf{x} - \int_{\partial\Omega} \frac{\partial u}{\partial \mathbf{n}} v \, ds,$$

and since $v|_{\partial\Omega} = 0$, the boundary term vanishes. Therefore the weak form is:

$$\int_\Omega u_{tt}(x,t) v(x) \, d\mathbf{x} + \int_\Omega \nabla u(x,t) \cdot \nabla v(x) \, d\mathbf{x} = \int_\Omega f(x,t) v(x) \, d\mathbf{x}, \quad \forall v \in V, \ \forall t \in (0,T). \quad (1.2)$$

Let $V_h \subset V$ be a finite element space of dimension N with nodal basis $\{\varphi_1, \ldots, \varphi_N\}$. We seek

$$u_h(x,t) = \sum_{j=1}^{N} U_j(t) \, \varphi_j(x),$$

and we choose test functions $v_h = \varphi_i$ for $i = 1, \ldots, N$. Since basis functions do not depend on time,

$$(u_h)_{tt}(x,t) = \sum_{j=1}^{N} \ddot{U}_j(t) \, \varphi_j(x), \qquad \nabla u_h(x,t) = \sum_{j=1}^{N} U_j(t) \, \nabla\varphi_j(x).$$

Plugging into (1.2) yields, for each i,

$$\sum_{j=1}^{N} \left(\int_\Omega \varphi_j \varphi_i \, d\mathbf{x} \right) \ddot{U}_j(t) + \sum_{j=1}^{N} \left(\int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i \, d\mathbf{x} \right) U_j(t) = \int_\Omega f \, \varphi_i \, d\mathbf{x}.$$

We define:

$$M_{ij} := \int_\Omega \varphi_j \varphi_i \, d\mathbf{x} \text{ (mass matrix)}, \qquad K_{ij} := \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i \, d\mathbf{x} \text{ (stiffness matrix)},$$

$$F_i(t) := \int_\Omega f \, \varphi_i \, d\mathbf{x}.$$

We introduce the vector of unknowns $\mathbf{U}(t) = [U_1(t), \ldots, U_N(t)]^T$. Then the semi-discrete system is [2, Chapt. 5]:

$$M\dot{\mathbf{U}}(t) + K\mathbf{U}(t) = \mathbf{F}(t). \quad (1.2)$$

1.2.2 Nonhomogeneous Dirichlet Conditions and Matrix Form via Lifting

Now we analize the case in which $u = g$ on $\partial\Omega$. We introduce an extension $\tilde{g}(\cdot,t) \in H^1(\Omega)$ such that $\tilde{g}|_{\partial\Omega} = g$. We define the lifted variable

$$w := u - \tilde{g} \ \Rightarrow \ w|_{\partial\Omega} = 0,$$

so that $w(t) \in V$.
Substituting $u = w + \tilde{g}$ into the weak form and moving known terms to the right:

$$\int_\Omega w_{tt} v \, d\mathbf{x} + \int_\Omega \nabla w \cdot \nabla v \, d\mathbf{x} = \int_\Omega f v \, d\mathbf{x} - \int_\Omega \tilde{g}_{tt} v \, d\mathbf{x} - \int_\Omega \nabla\tilde{g} \cdot \nabla v \, d\mathbf{x}, \quad \forall v \in V. \quad (1.4)$$

We discretize w and $\tilde{g}$ as

$$w_h(x,t) = \sum_{j=1}^{N} W_j(t)\varphi_j(x), \quad \tilde{g}_h(x,t) = \sum_{j=1}^{N} G_j(t)\varphi_j(x).$$

We introduce the vectors of the nodal values $\mathbf{W}(t) = [W_1(t), \ldots, W_N(t)]^T$ and $\mathbf{G}(t) = [G_1(t), \ldots, G_N(t)]^T$. Proceeding as before, one obtains:

$$M\ddot{\mathbf{W}}(t) + K\mathbf{W}(t) = \mathbf{F}(t) - M\ddot{\mathbf{G}}(t) - K\mathbf{G}(t). \tag{1.3}$$

Because $w = u - \tilde{g}$, the initial conditions are:

$$w(\cdot,0) = u_0 - \tilde{g}(\cdot,0) \Rightarrow W(0) = U_0 - G(0),$$

$$w_t(\cdot,0) = u_1 - \tilde{g}_t(\cdot,0) \Rightarrow \dot{W}(0) = U_1 - \dot{G}(0).$$

1.2.3 Partition into Interior and Boundary Nodes

We split degrees of freedom into interior I and Dirichlet boundary D:

$$W = \begin{pmatrix} W_I \\ W_D \end{pmatrix}, \quad G = \begin{pmatrix} G_I \\ G_D \end{pmatrix}, \quad F = \begin{pmatrix} F_I \\ F_D \end{pmatrix},$$

$$M = \begin{pmatrix} M_{II} & M_{ID} \\ M_{DI} & M_{DD} \end{pmatrix}, \quad K = \begin{pmatrix} K_{II} & K_{ID} \\ K_{DI} & K_{DD} \end{pmatrix}.$$

Starting from the lifted matrix formulation,

$$\begin{aligned} &\begin{pmatrix} M_{II} & M_{ID} \\ M_{DI} & M_{DD} \end{pmatrix} \begin{pmatrix} \ddot{W}_I \\ \ddot{W}_D \end{pmatrix} + \begin{pmatrix} K_{II} & K_{ID} \\ K_{DI} & K_{DD} \end{pmatrix} \begin{pmatrix} W_I \\ W_D \end{pmatrix} \\ &= \begin{pmatrix} F_I \\ F_D \end{pmatrix} - \begin{pmatrix} M_{II} & M_{ID} \\ M_{DI} & M_{DD} \end{pmatrix} \begin{pmatrix} \ddot{G}_I \\ \ddot{G}_D \end{pmatrix} - \begin{pmatrix} K_{II} & K_{ID} \\ K_{DI} & K_{DD} \end{pmatrix} \begin{pmatrix} G_I \\ G_D \end{pmatrix}, \end{aligned} \tag{1.4}$$

we exploit the properties of the lifted variable w.
Since $w = 0$ on the boundary nodes, we have $W_D = 0$. Moreover, we choose the FEM lifting $\tilde{g}_h$ such that all interior degrees of freedom vanish, i.e. $G_I = 0$ (Dirichlet data are prescribed only on boundary nodes).
With these choices, the first block row of the system (corresponding to interior nodes) reduces to

$$M_{II}\ddot{W}_I + M_{ID}\ddot{W}_D + K_{II}W_I + K_{ID}W_D = F_I - \left(M_{II}\ddot{G}_I + M_{ID}\ddot{G}_D + K_{II}G_I + K_{ID}G_D\right).$$

Imposing $W_D = 0$ and $G_I = 0$, we finally obtain

$$M_{II}\ddot{W}_I + K_{II}W_I = F_I - \left(M_{ID}\ddot{G}_D + K_{ID}G_D\right).$$

This is exactly the reduced formulation involving only the interior degrees of freedom:

$$M_{II}\ddot{W}_I(t) + K_{II}W_I(t) = F_I(t) - \left(M_{ID}\ddot{G}_D(t) + K_{ID}G_D(t)\right). \tag{1.7}$$

This is the equation solved numerically: boundary degrees of freedom are entirely determined by Dirichlet data through $G_D(t)$, while the dynamic evolution is computed only for the interior unknowns $W_I(t)$.

1.3 Computational Aspects of Space Discretization

The implementation of the spatial discretization is based on the deal.II finite element library, coupled with Trilinos for high-performance parallel linear algebra.

1.3.1 Finite Element Space and Mesh

The computational domain is generated as a hypercube (unit square in 2D).
Depending on the specific time integration scheme employed—which will be formally introduced in Chapter 2 we utilize either linear `FE_Q<dim>(1)` or quadratic `FE_Q<dim>(2)` elements. Specifically, linear elements are paired with the explicit solver, while quadratic elements are adopted for the implicit one.

The numerical integration for the Mass and Stiffness matrices is performed using Gaussian Quadrature (`QGauss`). We select a quadrature formula of degree $p+1$ to ensure exact integration of the bilinear forms.

Dispersion Control. The choice of quadratic elements ($p = 2$) over linear ones ($p = 1$) for the implicit formulation (which is detailed in Chapter 2) is primarily motivated by the need to minimize *numerical dispersion*. In a spatially discretized medium, waves of different frequencies propagate at varying numerical wave speeds. As observed in the plot, linear elements (Q_1, red curve) introduce a noticeable phase error: high-frequency components travel slower than the actual physical wavefront. This artificial lag causes the wave to lose its original profile, leading to shape distortion over time. Conversely, employing quadratic elements (Q_2, blue curve) provides superior spatial accuracy, drastically mitigating this phase error and ensuring that the physical wave shape is correctly preserved and perfectly aligned with the exact solution.

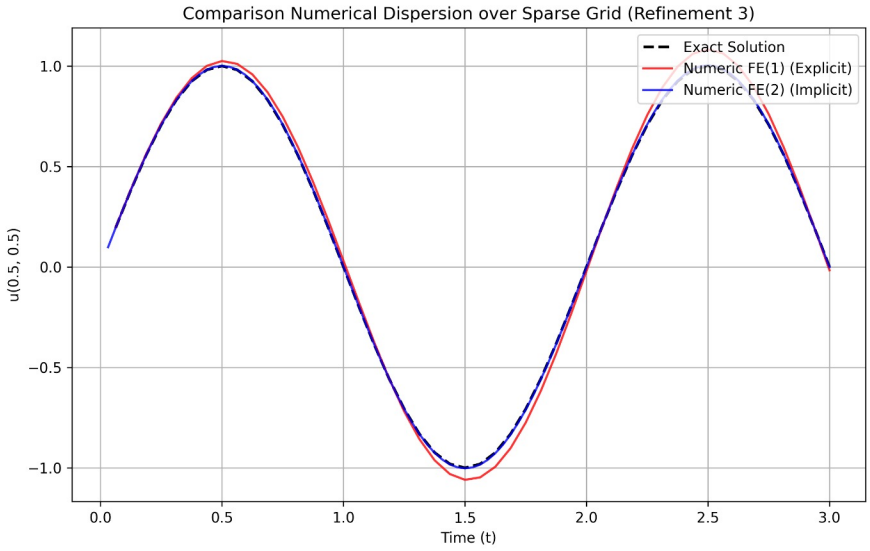

Figure 1.1: Comparison of numerical dispersion effects on a coarse spatial grid (refinement level 3). The plot displays the time history of the displacement at the center of the domain, $u(0.5, 0.5)$. The numerical solution employing linear elements (FE1, red solid line) exhibits noticeable amplitude distortion and a slight phase shift compared to the exact analytical solution (black dashed line). Conversely, the use of quadratic elements (FE2, blue solid line) significantly mitigates dispersive errors, providing a highly accurate approximation. This evidence robustly justifies the adoption of Q_2 elements to preserve wave propagation fidelity.

1.3.2 Parallel Data Structures (MPI & Trilinos)

The code ensures scalability on clusters by relying on a distributed memory model based on the Message Passing Interface (MPI). The underlying triangulation is partitioned across processors through domain decomposition, where each MPI rank manages a specific subset of cells and locally owned degrees of freedom.

To handle data distribution efficiently, we utilize `TrilinosWrappers::MPI::Vector`, which requires a crucial distinction between vector types. *Locally Owned* vectors are employed for standard algebraic operations, such as dot products and norms. However, computations on the partition interface require *Locally Relevant* (or *Ghosted*) vectors. While the assembly process iterates over locally owned cells, elements located at the interface share degrees of freedom (DoFs) with neighboring ranks. Consequently, to correctly evaluate the solution u_h or its gradient ∇u_h inside such local cells, access to the coefficients U_j of all associated DoFs is required, even if some are owned by a neighbor. These ghosted vectors provide the necessary read-access to these non-local values, ensuring mathematical consistency.

Finally, the linear system matrices are stored in Compressed Row Storage (CRS) format via `TrilinosWrappers::SparseMatrix`, a structure specifically optimized for parallel matrix-vector products. The code relies on a distributed memory model using the Message Passing Interface (MPI) to ensure scalability on clusters.

1.3.3 Matrix Assembly and Constraints

The assembly of the Mass (M) and Stiffness (K) matrices iterates over locally owned cells.
To optimize memory usage, a `TrilinosWrappers::SparsityPattern` is computed before assembly, pre-allocating the necessary non-zero entries based on the mesh connectivity.
Algebraic Constraints: Dirichlet boundary conditions and hanging nodes are handled algebraically via the `AffineConstraints` class. Instead of using penalty methods, these constraints are enforced *exactly* by modifying the linear system during assembly. For every degree of freedom i constrained to a value g_i, the corresponding row in the global matrix is replaced by a unit row (zero entries everywhere except for $A_{ii} = 1$), and the corresponding right-hand side entry is set to the prescribed value g_i. This guarantees that the solution satisfies $U_i = g_i$ to machine precision without increasing the size of the system.

Chapter 2

Time Discretization

2.1 Newmark-β Method

After space discretization, the wave problem leads to a second-order dynamical system:

$$M\ddot{\mathbf{U}}(t) + K\mathbf{U}(t) = \mathbf{F}(t), \tag{2.1}$$

where $\mathbf{U}(t)$ is the displacement vector, $\dot{\mathbf{U}}(t)$ the velocity, and $\ddot{\mathbf{U}}(t)$ the acceleration. The Newmark-β method is a one-step time integrator for systems of the form (2.1). Given the state at time t_n (i.e., $\mathbf{U}_n, \dot{\mathbf{U}}_n, \ddot{\mathbf{U}}_n$), it computes the state at $t_{n+1} = t_n + \Delta t$.
Given a time step Δt, the updates are [1, Sec. 9.1].:

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \Delta t\, \dot{\mathbf{U}}_n + \frac{\Delta t^2}{2}\left[(1-2\beta)\ddot{\mathbf{U}}_n + 2\beta\ddot{\mathbf{U}}_{n+1}\right], \tag{2.2}$$

$$\dot{\mathbf{U}}_{n+1} = \dot{\mathbf{U}}_n + \Delta t\left[(1-\gamma)\ddot{\mathbf{U}}_n + \gamma\ddot{\mathbf{U}}_{n+1}\right]. \tag{2.3}$$

From (2.2) one can isolate $\ddot{\mathbf{U}}_{n+1}$:

$$\ddot{\mathbf{U}}_{n+1} = \frac{1}{\beta\Delta t^2}\left(\mathbf{U}_{n+1} - \mathbf{U}_n - \Delta t\, \dot{\mathbf{U}}_n\right) - \frac{1-2\beta}{2\beta}\ddot{\mathbf{U}}_n. \tag{2.4}$$

Once $\mathbf{U}_{n+1}$ is computed, the new acceleration $\ddot{\mathbf{U}}_{n+1}$ can be derived directly via (2.4). However, it is crucial to note that this update relies on the availability of the full kinematic state from the previous step. Therefore, the algorithm must explicitly store and track not only the displacement, but also the velocity $\dot{\mathbf{U}}_n$ and the acceleration $\ddot{\mathbf{U}}_n$ throughout the time integration process.
We evaluate (2.1) at t_{n+1}:

$$M\ddot{\mathbf{U}}_{n+1} + K\mathbf{U}_{n+1} = \mathbf{F}_{n+1}.$$

We substitute (2.4):

$$\left(\frac{1}{\beta\Delta t^2}M + K\right)\mathbf{U}_{n+1} = \mathbf{F}_{n+1} + M\mathbf{R}_n,$$

where

$$\mathbf{R}_n := \frac{1}{\beta\Delta t^2}\left(\mathbf{U}_n + \Delta t\, \dot{\mathbf{U}}_n\right) + \frac{1-2\beta}{2\beta}\ddot{\mathbf{U}}_n.$$

We define the system matrix

$$A_{\text{sys}} := \frac{1}{\beta\Delta t^2}M + K.$$

Hence the linear system at each step is:

$$A_{\text{sys}}\mathbf{U}_{n+1} = \mathbf{F}_{n+1} + M\mathbf{R}_n. \tag{2.5}$$

Newmark requires the full initial state $(\mathbf{U}_0, \dot{\mathbf{U}}_0, \ddot{\mathbf{U}}_0)$. The physical problem typically provides $\mathbf{U}_0$ and $\dot{\mathbf{U}}_0$ only. Compute $\ddot{\mathbf{U}}_0$ by enforcing the equation of motion at $t = 0$:

$$M\ddot{\mathbf{U}}_0 + K\mathbf{U}_0 = \mathbf{F}(0) \quad \Rightarrow \quad \ddot{\mathbf{U}}_0 = M^{-1}\left(\mathbf{F}(0) - K\mathbf{U}_0\right).$$

2.2 Specific Schemes and Stability Analysis

The Newmark family includes distinct integration schemes depending on the choice of parameters β and γ. In this work, we set $\gamma = 0.5$ to ensure second-order accuracy and energy conservation (non-dissipative schemes) [1, Sec. 9.1]. The implementation of the time loop is handled within the `WaveEquation` class. We investigate two specific configurations:

2.2.1 Explicit Scheme

Setting $\beta = 0$, the method becomes explicit. However, the standard finite element mass matrix is sparse but not diagonal. To address this, we apply the Mass Lumping technique, which approximates M by summing all row entries onto the diagonal [2, Chapt. 5]. Since the resulting lumped mass matrix M is strictly diagonal, the linear system simplifies drastically, meaning no computationally expensive matrix inversion is required.
The acceleration update becomes a simple vector operation:

$$\ddot{\mathbf{U}} = M_L^{-1}(\mathbf{F} - K\mathbf{U})$$

drastically reducing the computational cost per step.

The scheme is conditionally stable, requiring the time step to satisfy the Courant-Friedrichs-Lewy (CFL) condition:

$$\Delta t \leq C\frac{h}{c}, \tag{2.1}$$

where C is a constant depending on the finite element polynomial degree (typically $C \propto 1/p$).

This constraint is not merely a numerical artifact but stems directly from the hyperbolic nature of the problem. Physically, the wave equation describes information propagating at a finite speed c, defining a physical domain of dependence often referred to as the "light cone" [3, Chap. 4.6]. For the numerical solution to be stable, the numerical domain of dependence (determined by the mesh size h and time step Δt) must fully contain the physical domain of dependence.
Geometrically, this implies that the numerical wave speed $h/\Delta t$ must be at least as fast as the physical wave speed c. If this condition is violated ($\Delta t > Ch/c$), the numerical scheme attempts to compute the solution at a node using information which has not yet physically reached that point (i.e., outside the light cone), leading to an exponential growth of numerical errors and instability. Compliance with this strict bound ensures that the discrete model respects the physics of wave propagation, naturally minimizing numerical dispersion.

Computational aspects: unlike the implicit solver, where the linear solver handles data exchange automatically, the explicit scheme requires manual synchronization of parallel vectors. At each time step, after updating U_{n+1} and V_{n+1} on locally owned DOFs, we explicitly invoke `update_ghost_values()`. This communication step is critical to ensure that the stencil operations (matrix-vector multiplication with K) have access to the correct neighbor data across processor boundaries.

From an implementation standpoint, satisfying the stability condition requires computing the global minimum mesh size $h_{\min}$ across the entire domain. In a parallel environment, this

involves an `MPI_Allreduce` operation to find the minimum h among all processors. This value is then used to determine the critical time step limit. The solver performs this check during the initialization phase, ensuring that the selected Δt respects the CFL bound before the time loop begins, thus preventing numerical divergence.

2.2.2 Implicit Scheme

Setting $\beta = 0.25$, we obtain the unconditional stable scheme used as the default solver.
The method is A-stable (unconditionally stable) [1, Sec. 9.1]. Unlike the explicit case, stability theory imposes no upper bound on the time step size Δt. This allows for significantly larger steps without the risk of the solution blowing up.
While stability is guaranteed, using large time steps introduces a numerical error. This means the numerical waves propagate slower than the physical ones, causing the simulated period of oscillation to be artificially larger than the exact period.

Computational aspects: for this solver, the system matrix $A_{\text{sys}} = \dfrac{1}{\beta \Delta t^2} M + K$ is time-independent (assuming constant Δt). Therefore we assemble A_{sys} only once before the time loop. At each step, only the Right-Hand Side (RHS) is updated with the new forcing terms and predictor values. Dirichlet conditions $u = g(t)$ are applied strongly by modifying the matrix rows and the RHS vector using algebraic lifting.
The system matrix A_{sys} is Symmetric and Positive Definite (SPD). We solve equation (2.5) using the Conjugate Gradient (CG) method provided by Trilinos. To ensure scalability and a mesh-independent convergence rate of the iterative solver (evaluated with respect to the discrete l^2-norm of the algebraic residual), we employ an Algebraic Multigrid (AMG) preconditioner. This is particularly effective for elliptic-like operators such as A_{sys} [2, Sec. 7.2].

.

Chapter 3

Conservation of Energy

3.1 Total Energy and Discrete Energy Calculation

We define total energy as kinetic plus potential [3, Chap. 5].:

$$E(t) := \frac{1}{2}\int_\Omega u_t^2 \, d\mathbf{x} + \frac{1}{2}\int_\Omega |\nabla u|^2 \, d\mathbf{x}. \tag{3.9}$$

Assuming sufficient smoothness,

$$\frac{dE}{dt} = \int_\Omega u_t u_{tt} \, d\mathbf{x} + \int_\Omega \nabla u \cdot \nabla u_t \, d\mathbf{x}. \tag{3.10}$$

Now we integrate by parts assuming $f = 0$, then $u_{tt} = \Delta u$ and

$$\frac{dE}{dt} = \int_\Omega u_t \Delta u \, d\mathbf{x} + \int_\Omega \nabla u \cdot \nabla u_t \, d\mathbf{x}.$$

We apply Green's identity:

$$\int_\Omega u_t \Delta u \, d\mathbf{x} = -\int_\Omega \nabla u_t \cdot \nabla u \, d\mathbf{x} + \int_{\partial\Omega} u_t \frac{\partial u}{\partial \mathbf{n}} \, ds.$$

The volume terms cancel with $\int_\Omega \nabla u \cdot \nabla u_t \, d\mathbf{x}$, leaving:

$$\frac{dE}{dt} = \int_{\partial\Omega} u_t \frac{\partial u}{\partial \mathbf{n}} \, ds. \tag{3.11}$$

Physically, the term on the right-hand side represents the net energy flux across the boundary $\partial\Omega$. It can be interpreted as the work done by the elastic forces (related to the normal derivative $\frac{\partial u}{\partial \mathbf{n}}$) acting on the moving boundary (with velocity $\frac{\partial u}{\partial t}$).
This relation highlights the role of boundary conditions in the thermodynamics of the system. In the case of homogeneous Dirichlet conditions ($u = 0$ on $\partial\Omega$), the boundary velocity vanishes ($\frac{\partial u}{\partial t} = 0$) everywhere on the boundary. Consequently, the boundary integral becomes zero, ensuring the conservation of total energy ($\frac{dE}{dt} = 0$). This confirms that in an isolated system with fixed boundaries and no external forcing, the energy remains constant over time [3, Chap. 5].

To show the consistency between the continuous energy $E(t)$ defined in Eq. (3.9) and its discrete counterpart, we evaluate the integral expression at the discrete time step t_n. Substituting the finite element approximation at time t_n, denoted as $u_h(t_n, \mathbf{x}) = \sum_{j=1}^{N_h} U_j^n \varphi_j(\mathbf{x})$, into the integral expression yields the matrix form.

The kinetic energy term transforms as follows:

$$\begin{aligned}\frac{1}{2}\int_\Omega \left(\frac{\partial u_h}{\partial t}(t_n)\right)^2 d\mathbf{x} &= \frac{1}{2}\int_\Omega \left(\sum_i V_i^n \varphi_i\right)\left(\sum_j V_j^n \varphi_j\right) d\mathbf{x} \\ &= \frac{1}{2}\sum_{i,j} V_i^n V_j^n \underbrace{\int_\Omega \varphi_i \varphi_j \, d\mathbf{x}}_{M_{ij}} = \frac{1}{2}(\mathbf{V}^n)^T M \mathbf{V}^n.\end{aligned} \tag{3.1}$$

Similarly, the potential elastic energy term becomes:

$$\begin{aligned}\frac{1}{2}\int_\Omega |\nabla u_h(t_n)|^2 d\mathbf{x} &= \frac{1}{2}\int_\Omega \left(\sum_i U_i^n \nabla\varphi_i\right)\cdot\left(\sum_j U_j^n \nabla\varphi_j\right) d\mathbf{x} \\ &= \frac{1}{2}\sum_{i,j} U_i^n U_j^n \underbrace{\int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \, d\mathbf{x}}_{K_{ij}} = \frac{1}{2}(\mathbf{U}^n)^T K \mathbf{U}^n.\end{aligned} \tag{3.2}$$

where M and K are the mass and stiffness matrices, $\mathbf{U}^n$ is the displacement vector, and $\mathbf{V}^n$ is the **velocity vector** at time t_n (defined as $\mathbf{V}^n = \dot{\mathbf{U}}^n$).
Summing the kinetic and potential contributions, we obtain the explicit formula for the discrete total energy at time t_n:

$$E_h(t_n) = \frac{1}{2}(\mathbf{V}^n)^T M \mathbf{V}^n + \frac{1}{2}(\mathbf{U}^n)^T K \mathbf{U}^n. \tag{3.3}$$

Thus, the discrete energy computed in the code is exactly the evaluation of the continuous energy functional on the finite element subspace at each time step t_n.

3.1.1 Case Analysis and Numerical Verification

We define two distinct regimes to validate the physical consistency of our solver.

Case 1: Isolated System

In this scenario we assume that the boundary conditions are homogeneous ($g(t) = 0$ on $\partial\Omega$), so as mentioned before the energy balance equation becomes zero:

$$\frac{dE}{dt} = 0 \implies E(t) = E(0) = \text{constant}.$$

Numerical Result: We simulated the wave equation with $g = 0$ using our Newmark solver with $\gamma = 0.5$. As shown in Figure 3.1, the total energy remains perfectly constant over time. This result confirms that our implementation is strictly non-dissipative. It demonstrates that there is no artificial numerical damping introduced by the time integration scheme.

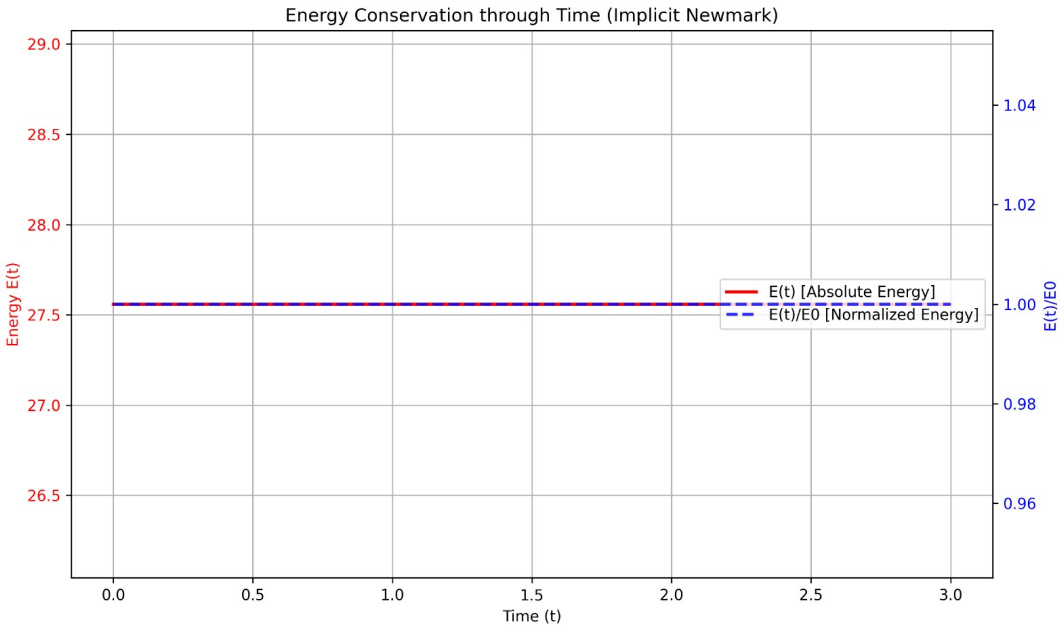

Figure 3.1: **Conservation of Energy in an Isolated System.** The plot shows $E(t)$ versus time for the case with homogeneous Dirichlet boundary conditions ($g = 0$). The horizontal line confirms that the total energy is conserved, validating the symplectic nature of the Newmark scheme ($\gamma = 0.5$).

Case 2: Driven System

In the scenario where the boundary moves according to a time-dependent function $u = g(t, y) = 0.5\sin(5t)\sin(\pi y)$, the velocity at the boundary is non-zero ($u_t = g'(t)$). The energy balance equation becomes:

$$\frac{dE}{dt} = \int_{\partial\Omega} g'(t)\frac{\partial u}{\partial \mathbf{n}}\, ds \neq 0$$

Figure 3.2 illustrates the time evolution of the total discrete energy for the driven configuration. Unlike the unforced scenario, the energy profile here is not constant but exhibits significant fluctuations. The strict conservation observed in the previous validation benchmark (Case 1) allows us to rule out numerical instability or artificial dissipation as the cause of these variations. Consequently, these fluctuations are correctly interpreted as the physical manifestation of the work done by the time-dependent boundary forces.

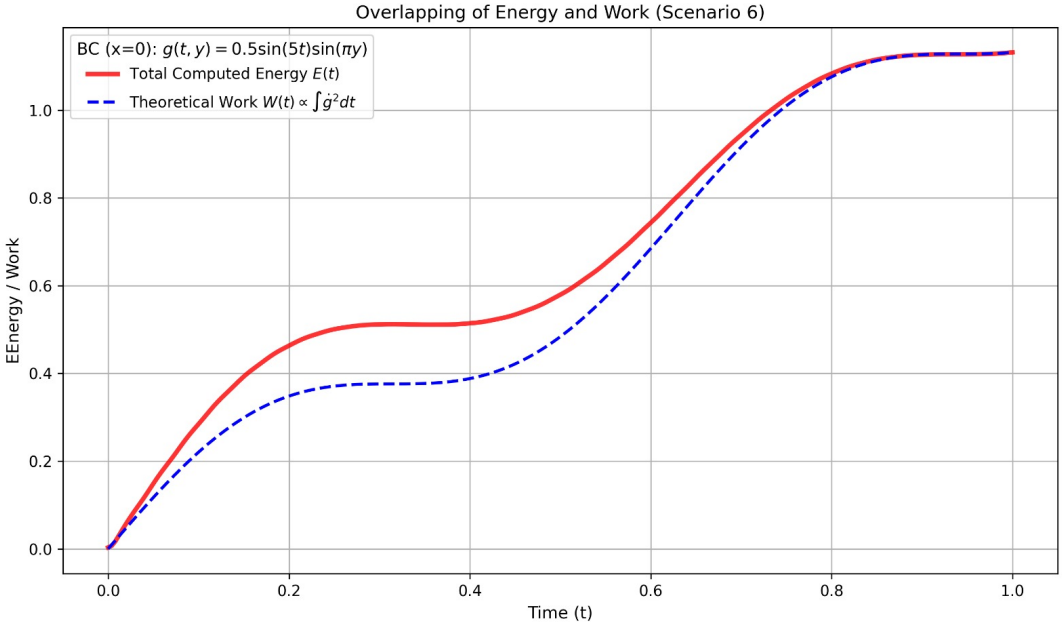

Figure 3.2: **Energy Evolution in a Driven System.** The plot shows $E(t)$ for the case with time-dependent boundary conditions ($g(t) \neq 0$). The observed increase in energy is physically expected and is consistent with the theoretical work injected into the system by the moving boundary."

Case 3: Algorithmic Dissipation and the Choice of γ

To justify our parameter selection, we simulated a strictly isolated system ($f = 0, g = 0$) using a dissipative Newmark scheme ($\gamma = 0.6$, $\beta = 0.3$).
As shown in Figure 3.3, the total energy exhibits a continuous decay. This test rigorously justifies our strict use of the average acceleration method ($\gamma = 0.5$, $\beta = 0.25$) in all primary simulations, as it is the configuration that preserves the thermodynamics and symplecticity of the physical model without artificial energy loss.

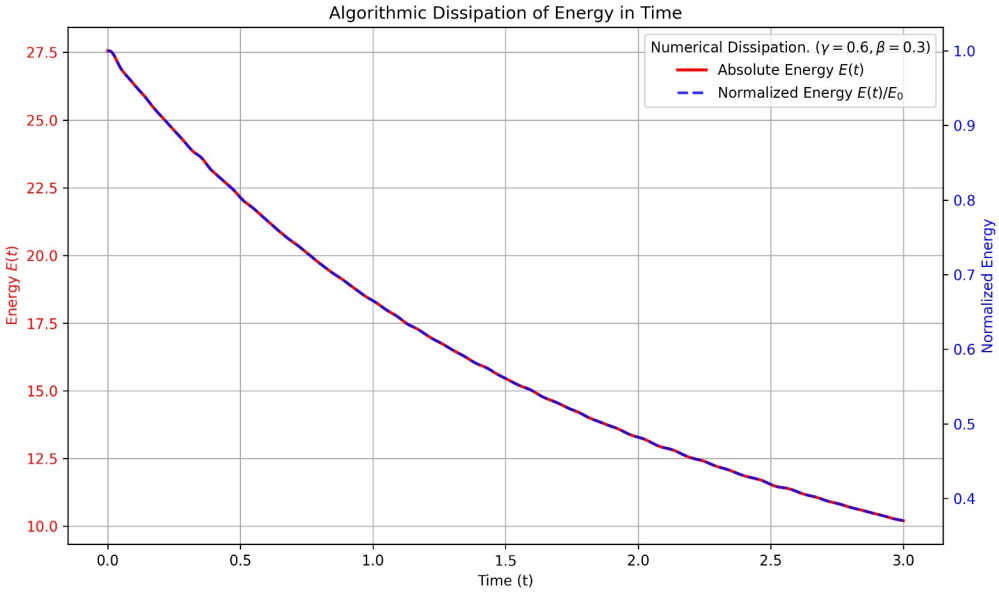

Figure 3.3: **Algorithmic Energy Dissipation.** Total energy evolution in an isolated system using $\gamma = 0.6$ and $\beta = 0.3$

Chapter 4

Convergence and Scalability

4.0.1 Convergence Analysis

To verify the accuracy and correctness of our numerical implementation, we conducted a spatial and temporal convergence study using the exact solution

$$u(\mathbf{x}, t) = \sin(\pi t) \prod_{i=1}^{d} \sin(\pi x_i) \tag{4.1}$$

The tests were executed for both the Explicit and Implicit time-marching schemes by progressively refining the spatial grid (decreasing the mesh size h) and measuring the global error in the L^2-norm. [1]
The log-log plots in Figure 4.1 illustrate the reduction of the L^2 error as the grid size h decreases. In such graphs, the slope of the curve directly represents the global order of convergence of the numerical solver.

Both the Explicit (Velocity Verlet) and Implicit (Newmark-β with $\gamma = 0.5$) time integration algorithms are designed to be second-order accurate [1], yielding a temporal error of $\mathcal{O}(\Delta t^2)$. During the simulations, the automated CFL condition rigidly couples the time step to the mesh size linearly ($\Delta t \propto h$). As a result, the total global error is expected to scale quadratically, effectively collapsing to $\mathcal{O}(h^2)$ [2]. The experimental data flawlessly reflects this theoretical framework.

The computed slopes are 2.03 for the explicit scheme and 2.01 for the implicit one, effectively the theoretical reference line. This alignment with the theoretical quadratic rate validates the robustness of our solver, confirming that both the spatial assembly and the temporal marching loops are implemented correctly without any unwanted numerical dissipation.

4.0.2 Impact of MPI Parallelization and Scalability Analysis

Numerical Consistency

The introduction of MPI-based parallelization resulted in a negligible numerical discrepancy when compared to serial execution. The small differences observed between serial and parallel results can be attributed to floating-point round-off effects and the non-associativity of parallel reduction operations. From a practical perspective, this minor numerical variation is largely outweighed by the significant performance improvements achieved through domain decomposition and parallel execution. The parallel implementation demonstrated correct synchronization of distributed vectors, consistent enforcement of boundary conditions, and stable evaluation of the system energy across multiple processes.

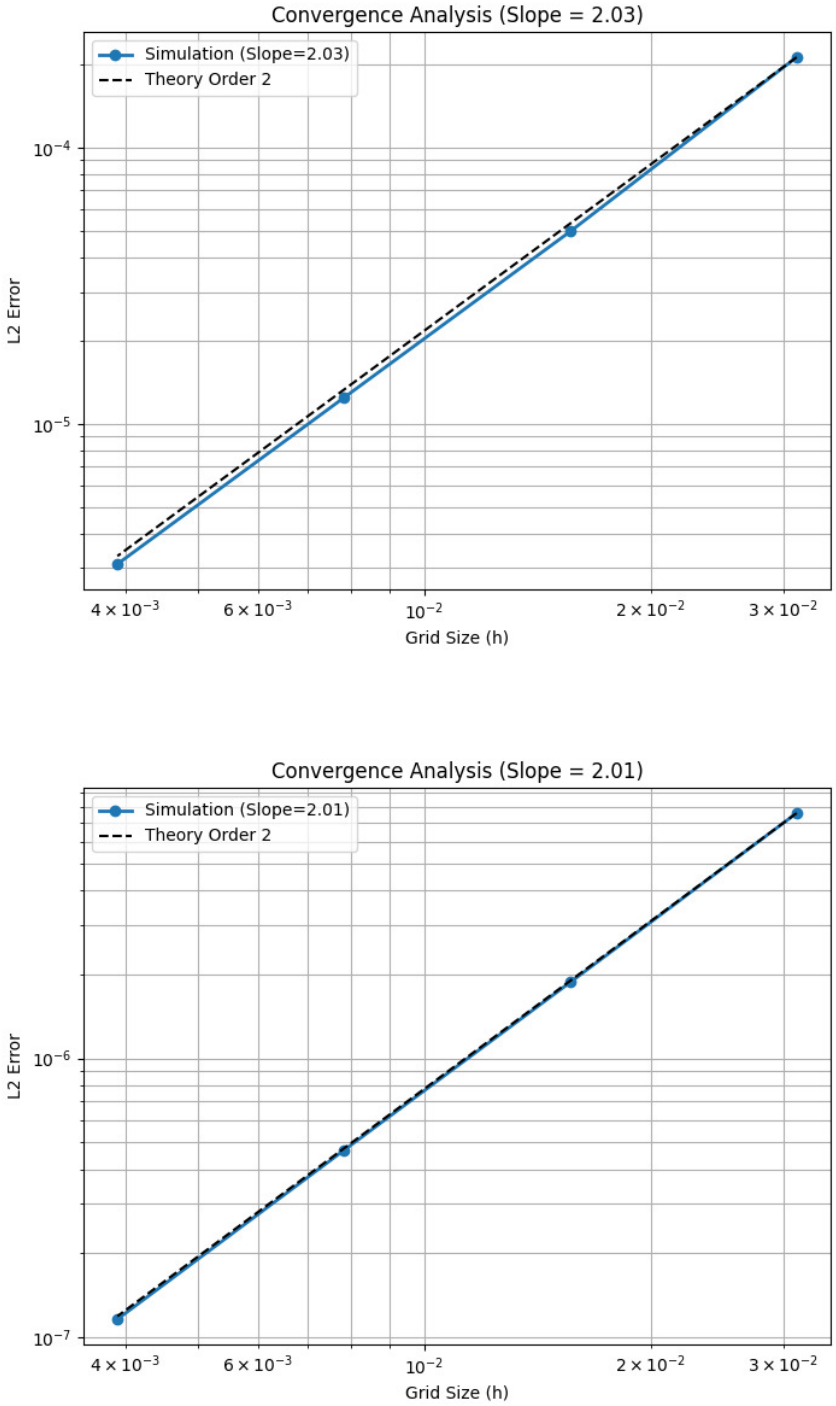

Figure 4.1: **Convergence Rates.** Global L^2-norm error versus mesh size h plotted on a log-log scale. **Top:** Explicit Scheme. **Bottom:** Implicit Scheme. The dashed line represents the theoretical quadratic slope.

Parallel Performance and Scalability Analysis

To quantitatively assess the scalability of our distributed explicit solver, performance tests were conducted under controlled workloads. To isolate spatial scaling from temporal scaling, the final simulation time T was halved at each spatial refinement step. Since the CFL condition requires $\Delta t \propto h$, this adjustment ensures the total number of time iterations remains strictly constant across all tests.

Strong Scaling. Strong scaling evaluates the speedup when solving a fixed-size problem with an increasing number of processors. In our setup, the global mesh remains unchanged but is partitioned into progressively smaller subdomains per core. As shown in Figure 4.2 (Top), for a Refinement 8 grid, the solver demonstrates excellent efficiency up to 4 cores, achieving a speedup of ~ 2.95. This indicates well-balanced local workloads and negligible communication overhead at this scale.

Weak Scaling. Conversely, weak scaling measures parallel efficiency when both the problem size and the processor count are scaled proportionally, ideally yielding a constant execution time. In our code, adding MPI ranks accompanies a systematic mesh refinement, maintaining a constant workload (degrees of freedom) per core. Figure 4.2 (Bottom) shows that while efficiency is above 95% up to 4 cores, it noticeably degrades to approximately 0.44 at 16 cores.

Bottleneck Diagnosis. The efficiency drop at 16 cores highlights inherent hardware and granularity limitations of explicit PDE solvers rather than implementation flaws. First, regarding Network Latency, 4-core runs typically operate within a single CPU socket using ultra-fast shared memory, whereas 16-core runs are distributed across multiple network nodes, introducing unavoidable physical latency. Secondly, MPI Granularity becomes critical: the explicit scheme requires exchanging boundary "ghost cells" at every single time step. At 16 cores, the subdomains' surface-to-volume ratio worsens significantly, causing MPI communication time to heavily outweigh local interior computations. Finally, the global mesh setup and initial partitioning carry a fixed serial overhead, which disproportionately penalizes the overall efficiency as the parallel computation time shrinks.

4.0.3 Final Remarks

The numerical experiments conducted in this project demonstrate that both the implicit and explicit solvers are correctly implemented, physically consistent, and numerically robust. The implicit method provides excellent stability and accuracy, while the explicit method offers substantial computational advantages with minimal loss in precision. The MPI-parallel implementation preserves these properties while enabling efficient large-scale simulations.
Together, these results confirm the effectiveness of the chosen numerical strategies for solving the wave equation and provide a solid foundation for further extensions and performance-oriented studies.

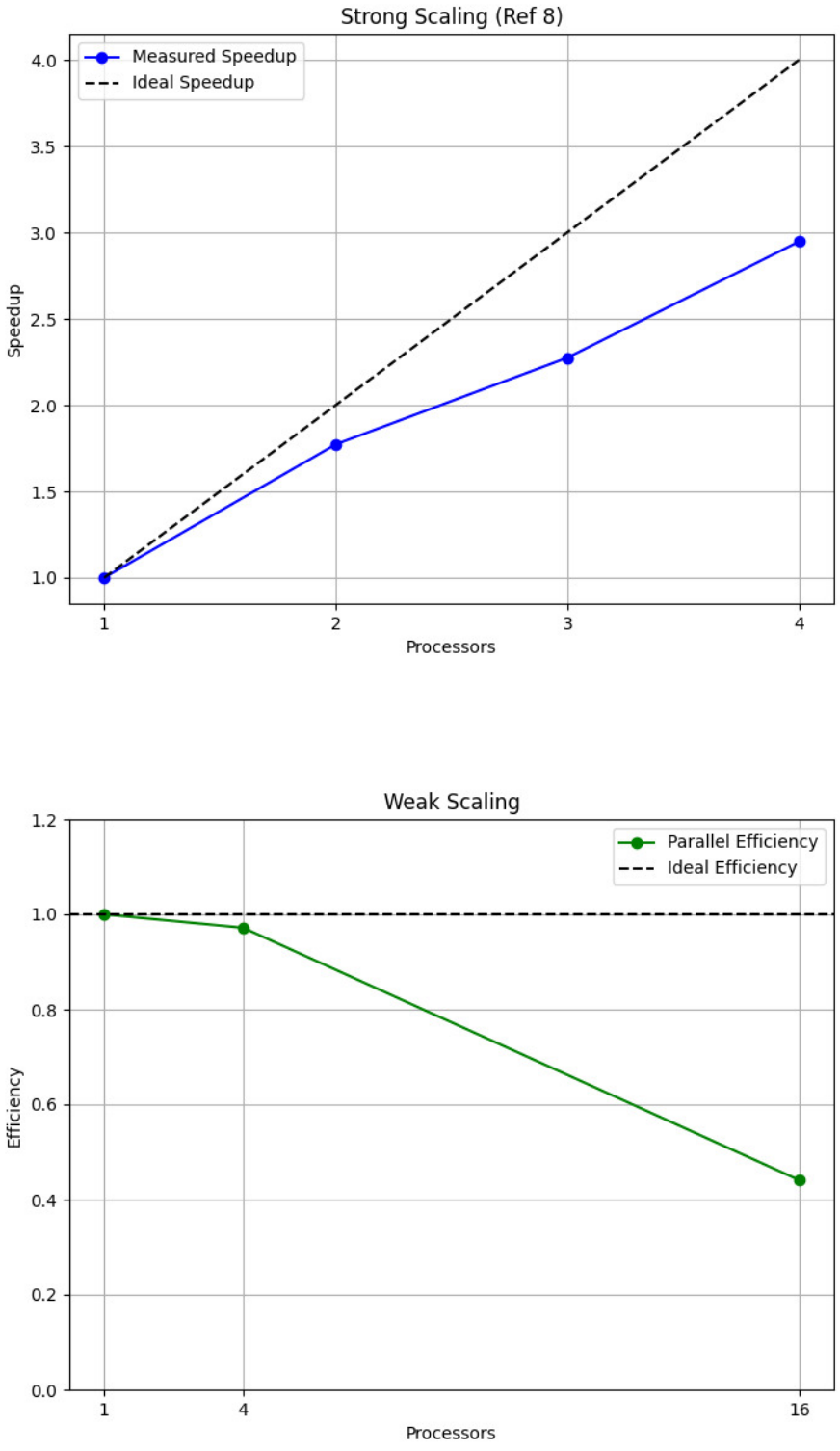

Figure 4.2: **Parallel Scalability. Top:** Strong Scaling speedup measured for a fixed problem size (Refinement 8). **Bottom:** Weak Scaling parallel efficiency measured up to 16 processors, scaling the problem size proportionally to the core count.

Bibliography

[1] Thomas J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Mineola, NY, 2000. Republication of the 1987 edition.

[2] Alfio Quarteroni. *Numerical Models for Differential Problems*. MS&A – Modeling, Simulation and Applications. Springer, 3rd edition, 2017.

[3] Sandro Salsa. *Partial Differential Equations in Action: From Modelling to Theory*. Universitext. Springer-Verlag Italia, Milano, 2008.