Wave Equation: Space and Time Discretization, Light Cones, and Energy Conservation

December 19, 2025

Contents

1 Space Discretization **3**
- 1.1 Project Model . . . 3
- 1.2 Physical Interpretation of the Model . . . 3
- 1.3 Functional Spaces . . . 3
  - 1.3.1 Definition of $H^1(\Omega)$ . . . 3
  - 1.3.2 Test Space for Dirichlet Conditions . . . 3
- 1.4 Weak Formulation (Homogeneous Dirichlet Case) . . . 4
- 1.5 Matrix (Semi-Discrete) Form . . . 4
- 1.6 Nonhomogeneous Dirichlet Conditions via Lifting . . . 4
- 1.7 Matrix Form with Lifting . . . 5
- 1.8 Partition into Interior and Boundary Nodes . . . 5
- 1.9 Computational Aspects of Space Discretization . . . 6
  - 1.9.1 Finite Element Space and Mesh . . . 6
  - 1.9.2 Parallel Data Structures (MPI & Trilinos) . . . 6
  - 1.9.3 Matrix Assembly and Constraints . . . 6

2 Time Discretization **7**
- 2.1 Introduction to the Newmark-β Method . . . 7
- 2.2 General Newmark-β Update Equations . . . 7
- 2.3 Explicit Derivation of $\ddot{U}_{n+1}$ . . . 7
- 2.4 Coupling with the Equation of Motion . . . 7
- 2.5 Initial Acceleration $\ddot{U}_0$ . . . 8
- 2.6 Specific Schemes and Stability Analysis . . . 8
  - 2.6.1 Explicit Scheme . . . 8
  - 2.6.2 Implicit Scheme . . . 8
- 2.7 Computational Aspects of Time Discretization . . . 8
  - 2.7.1 Linear System Assembly (Implicit Case) . . . 8
  - 2.7.2 Solvers and Preconditioning . . . 9
  - 2.7.3 Optimization of the Explicit Case . . . 9

3 Light Cones and Conservation of Energy **10**
- 3.1 Light Cones and Characteristic Strips . . . 10
  - 3.1.1 General Nonlinear First-Order PDE . . . 10
  - 3.1.2 Characteristic Strips . . . 10
  - 3.1.3 Geometric Optics and the Eikonal Equation . . . 10
  - 3.1.4 Light Cone . . . 10
- 3.2 Numerical Implications: Explicit vs. Implicit . . . 11
  - 3.2.1 Explicit Scheme (e.g. Leapfrog) . . . 11
  - 3.2.2 Implicit Scheme (Newmark with $\beta = 0.25,\ \gamma = 0.5$) . . . 11
  - 3.2.3 Energy Conservation and Parameter Choice . . . 11
  - 3.2.4 Stability and Diagnostics . . . 11

3.3 Conservation of Energy for the Wave Equation 11
3.3.1 Total Energy 11
3.3.2 Energy Derivative 11
3.3.3 Integration by Parts (Free Case $f = 0$) 12
3.3.4 Discrete Energy Calculation 12
3.3.5 Case Analysis and Numerical Verification 12

4 General Conclusions **15**
4.0.1 Implicit Method: Accuracy and Stability 15
4.0.2 Explicit Method: Efficiency and Physical Consistency 15
4.0.3 Impact of MPI Parallelization 16
4.0.4 Convergence Analysis 16
4.0.5 Final Remarks 16

Chapter 1

Space Discretization

1.1 Project Model

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

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

1.2 Physical Interpretation of the Model

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 $x = (x,y)$ and time t.

- **Inertial Term** $\left(\dfrac{\partial^2 u}{\partial t^2}\right)$: Represents the acceleration of the membrane's mass.
- **Restoring Force** $(-\Delta u)$: Represents the elastic tension.
- **Source Term** (f): Represents a distributed external vertical force density acting on the membrane surface.

Interpretation of Initial and Boundary Conditions

- **Initial Conditions** (u_0, u_1): u_0 defines the initial shape (deformation) of the membrane, while u_1 defines its initial velocity.

Finally, the hyperbolic nature of the equation implies a finite propagation speed ($c = 1$).

1.3 Functional Spaces

1.3.1 Definition of $H^1(\Omega)$

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

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

1.3.2 Test Space for Dirichlet Conditions

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

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

1.4 Weak Formulation (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:

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

Using Green's identity,

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

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

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

1.5 Matrix (Semi-Discrete) Form

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,

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

Define:

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

$$F_i(t) := \int_\Omega f \, \varphi_i \, dx.$$

Then the semi-discrete system is:

$$M\ddot{U}(t) + KU(t) = F(t). \quad (1.3)$$

1.6 Nonhomogeneous Dirichlet Conditions via Lifting

We return to the project case $u = g$ on $\partial\Omega$. Introduce an extension $\tilde{g}(\cdot, t) \in H^1(\Omega)$ such that $\tilde{g}|_{\partial\Omega} = g$. Define the lifted variable

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

so $w(t) \in V$.

Substituting $u = w + \tilde{g}$ into the weak form and moving known terms to the right:

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

1.7 Matrix Form with Lifting

Discretize w and $\tilde{g}$ as

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

Proceeding as before, one obtains:

$$M\ddot{W}(t) + KW(t) = F(t) - M\ddot{G}(t) - KG(t). \tag{1.5}$$

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

$$w(\cdot,0) = u_0 - \tilde{g}(\cdot,0) \Rightarrow W(0) = U_0 - G(0),$$
$$w_t(\cdot,0) = u_1 - \tilde{g}_t(\cdot,0) \Rightarrow \dot{W}(0) = U_1 - \dot{G}(0).$$

1.8 Partition into Interior and Boundary Nodes

Split degrees of freedom into interior I and Dirichlet boundary D:

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

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

Since $w|_{\partial\Omega} = 0$, we have $W_D = 0$. Moreover we can choose an FEM lifting such that $G_I = 0$ (Dirichlet data prescribed only on boundary nodes). The interior block reduces to:

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

Reduced system with lifting. Starting from the lifted matrix formulation,

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

we exploit the properties of the lifted variable w.

Since $w = 0$ on 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 the Dirichlet data through $G_D(t)$, while the dynamic evolution is computed only for the interior unknowns $W_I(t)$.

1.9 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.

1.9.1 Finite Element Space and Mesh

The computational domain is generated as a hypercube (unit square in 2D).

- **Element Type:** We utilize `FE_Q<dim>(2)` elements. These are continuous, bi-quadratic Lagrangian elements (Q2). In 2D, this implies 9 degrees of freedom (DOFs) per cell, located at the vertices, edge midpoints, and the cell center.
- **Justification (Dispersion Control):** The choice of quadratic elements ($p = 2$) over linear ones ($p = 1$) is primarily motivated by the need to control numerical dispersion. Wave propagation problems are highly sensitive to dispersion: linear elements (Q1) tend to introduce significant phase lag, manifesting as spurious "ripples" trailing the main wavefront even on relatively fine meshes. Quadratic elements (Q2) provide a higher order of spatial accuracy.
- **Quadrature:** Numerical integration for the Mass and Stiffness matrices is performed using Gaussian Quadrature (`QGauss`). We select a quadrature formula of degree $p + 1$ (where $p = 1$ is the finite element degree) to ensure exact integration of the bilinear forms.

1.9.2 Parallel Data Structures (MPI & Trilinos)

The code relies on a distributed memory model using the Message Passing Interface (MPI) to ensure scalability on clusters.

- **Domain Decomposition:** The triangulation is partitioned across processors. Each MPI rank manages a subset of cells (`locally_owned_dofs`).
- **Distributed Vectors:** We use `TrilinosWrappers::MPI::Vector`. A crucial implementation detail is the distinction between:
  - **Locally Owned vectors:** Used for algebraic operations (dot products, norms).
  - **Locally Relevant (Ghosted) vectors:** Store data from neighboring processor halos. Used for evaluating gradients near partition boundaries and for outputting results (`outputresults` function).
- **Sparse Matrices:** The system matrices are stored in Compressed Row Storage (CRS) format via `TrilinosWrappers::SparseMatrix`, optimized for parallel matrix-vector products.

1.9.3 Matrix Assembly and Constraints

The assembly of the Mass (M) and Stiffness (K) matrices iterates over locally owned cells.

- **Sparsity Pattern:** 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 by the `AffineConstraints` class. This modifies the local matrix contributions during the assembly process (`distribute_local_to_global`) to strictly enforce constraints $u|_{\partial\Omega} = g(t)$.

Chapter 2

Time Discretization

2.1 Introduction to the Newmark-β Method

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

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

where $U(t)$ is the displacement vector, $\dot{U}(t)$ the velocity, and $\ddot{U}(t)$ 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., $U_n, \dot{U}_n, \ddot{U}_n$), it computes the state at $t_{n+1} = t_n + \Delta t$.

2.2 General Newmark-β Update Equations

Given a time step Δt, the updates are:

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

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

2.3 Explicit Derivation of $\ddot{U}_{n+1}$

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

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

Once U_{n+1} is known, $\ddot{U}_{n+1}$ follows directly.

2.4 Coupling with the Equation of Motion

Evaluate (2.1) at t_{n+1}:

$$M\ddot{U}_{n+1} + KU_{n+1} = F_{n+1}.$$

Substitute (2.4):

$$\left(\frac{1}{\beta\Delta t^2}M + K\right)U_{n+1} = F_{n+1} + MR_n,$$

where

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

Define the system matrix

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

Hence the linear system at each step is:

$$A_{\text{sys}} U_{n+1} = F_{n+1} + M R_n. \tag{2.5}$$

2.5 Initial Acceleration $\ddot{U}_0$

Newmark requires the full initial state $(U_0, \dot{U}_0, \ddot{U}_0)$. The physical problem typically provides U_0 and $\dot{U}_0$ only. Compute $\ddot{U}_0$ by enforcing the equation of motion at $t = 0$:

$$M\ddot{U}_0 + KU_0 = F(0) \;\Rightarrow\; \ddot{U}_0 = M^{-1} \left(F(0) - KU_0 \right).$$

2.6 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). We investigate two specific configurations:

2.6.1 Explicit Scheme

Setting $\beta = 0$, the method becomes explicit. The linear system (2.5) simplifies drastically if the mass matrix M is diagonal, as no matrix inversion is required.

- **Stability:** The scheme is conditionally stable. The time step must satisfy the CFL condition $\Delta t \leq Ch/c$.
- **Dispersion:** Requires small time steps, naturally limiting numerical dispersion errors.

2.6.2 Implicit Scheme

Setting $\beta = 0.25$, we obtain the unconditional stable scheme used as the default solver.

- **Stability:** The method is A-stable (unconditionally stable). There is no strict limit on Δt, allowing for larger steps compared to the explicit case.
- **Dispersion:** While stable, large time steps introduce period elongation error.

2.7 Computational Aspects of Time Discretization

The implementation of the time loop is handled within the `WaveEquation` class.

2.7.1 Linear System Assembly (Implicit Case)

For the implicit solver ($\beta = 0.25$), the system matrix $A_{\text{sys}} = \frac{1}{\beta \Delta t^2} M + K$ is time-independent (assuming constant Δt).

- **Efficiency:** 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.
- **Boundary Conditions:** Dirichlet conditions $u = g(t)$ are applied strongly by modifying the matrix rows and the RHS vector using algebraic lifting (via `AffineConstraints`).

2.7.2 Solvers and Preconditioning

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.

- **Preconditioner:** To ensure scalability and mesh-independent convergence, we employ an Algebraic Multigrid (AMG) preconditioner. This is particularly effective for elliptic-like operators such as A_{sys}.

2.7.3 Optimization of the Explicit Case

For the explicit solver ($\beta = 0$), we avoid solving a linear system entirely by utilizing **Mass Lumping**.

- The code computes a lumped mass matrix M_L by summing the rows of the consistent mass matrix.
- Since M_L is diagonal, its inverse is trivial. The acceleration update becomes a simple vector operation:
$$\ddot{U} = M_L^{-1}(F - KU),$$
drastically reducing the computational cost per step.

Explicit Ghost Synchronization: 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.

Chapter 3

Light Cones and Conservation of Energy

3.1 Light Cones and Characteristic Strips

3.1.1 General Nonlinear First-Order PDE

Consider a generic nonlinear first-order PDE:

$$F(x, y, u, u_x, u_y) = 0, \tag{3.1}$$

where $u(x, y)$ is the unknown surface. Denote $p = u_x$ and $q = u_y$.

3.1.2 Characteristic Strips

A solution carries both (x, y, u) and tangent information (p, q). The 5D object $(x(t), y(t), z(t), p(t), q(t))$ is a characteristic strip, governed by:

$$\frac{dx}{dt} = F_p, \qquad \frac{dy}{dt} = F_q, \tag{3.2}$$

$$\frac{dz}{dt} = pF_p + qF_q, \tag{3.3}$$

$$\frac{dp}{dt} = -F_x - pF_u, \qquad \frac{dq}{dt} = -F_y - qF_u. \tag{3.4}$$

A key property is that F is constant along characteristics, so if $F = 0$ initially, it remains 0.

3.1.3 Geometric Optics and the Eikonal Equation

For wave propagation, consider the Eikonal equation

$$c^2(u_x^2 + u_y^2) = 1 \iff F = \frac{1}{2}\left[c^2(p^2 + q^2) - 1\right] = 0. \tag{3.5}$$

The level sets $u(x, y) = t$ are wavefronts, and orthogonal trajectories represent rays.

3.1.4 Light Cone

The region of influence in spacetime is the light cone:

$$(x - x_0)^2 + (y - y_0)^2 = c^2(z - z_0)^2, \tag{3.6}$$

where z denotes the time coordinate.

3.2 Numerical Implications: Explicit vs. Implicit

3.2.1 Explicit Scheme (e.g. Leapfrog)

For an explicit choice, stability requires a CFL bound:

$$\Delta t \leq C\frac{h}{c}. \tag{3.7}$$

3.2.2 Implicit Scheme (Newmark with $\beta = 0.25,\ \gamma = 0.5$)

With $\beta = 0.25, \gamma = 0.5$, one solves a linear system each step. The method is unconditionally stable in the classical sense. However, large Δt still causes dispersion (phase error), even if the solution does not blow up.

3.2.3 Energy Conservation and Parameter Choice

- If $\gamma > 0.5$, the scheme becomes dissipative (artificial damping), which can contradict energy conservation in free systems.
- Choosing $\gamma = 0.5$ matches the conservative property in the homogeneous/free case.

3.2.4 Stability and Diagnostics

Automated CFL Checking

To ensure the simulation respects the physical domain of dependence (Light Cones), the code implements an automatic check routine (`auto_check_cfl_condition`).

- The code computes the global minimum mesh diameter $h_{\min}$ using an MPI reduction (`Utilities::MPI::min`).
- It calculates the critical time step $\Delta t_{\text{crit}} \approx \dfrac{h_{\min}}{c\sqrt{d}}$.
- If the explicit method is selected, the time step is automatically adjusted to $\Delta t = 0.5\,\Delta t_{\text{crit}}$ to satisfy the CFL condition. For the implicit solver, this metric serves as a warning for potential dispersion errors if the Courant number exceeds 1.

3.3 Conservation of Energy for the Wave Equation

Assume unit speed $c = 1$. Consider:

$$\begin{cases} u_{tt} - \Delta u = f & \text{in } \Omega \times (0,T], \\ u = g(t) & \text{on } \partial\Omega \times (0,T], \\ u(\cdot,0) = u_0, \quad u_t(\cdot,0) = u_1 & \text{in } \Omega. \end{cases} \tag{3.8}$$

3.3.1 Total Energy

Define total energy as kinetic plus potential:

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

3.3.2 Energy Derivative

Assuming sufficient smoothness,

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

3.3.3 Integration by Parts (Free Case $f = 0$)

If $f = 0$, then $u_{tt} = \Delta u$ and

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

Apply Green's identity:

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

The volume terms cancel with $\int_\Omega \nabla u \cdot \nabla u_t \, dx$, leaving:

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

3.3.4 Discrete Energy Calculation

To monitor conservation numerically, we do not perform quadrature integration at every step. Instead, we leverage the algebraic structure of the FEM matrices. The discrete total energy $E_h(t)$ is computed efficiently via parallel matrix-vector products:

$$E_h(t) = \frac{1}{2} V_n^T M V_n + \frac{1}{2} U_n^T K U_n \tag{3.1}$$

where V_n and U_n are the global velocity and displacement vectors. This operation involves distributed dot products provided by Trilinos, ensuring that the energy metric is consistent with the discretized weak form.

3.3.5 Case Analysis and Numerical Verification

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

Case 1: Isolated System (Validation)

Hypothesis: If the boundary conditions are homogeneous ($g(t) = 0$ on $\partial\Omega$), the system is mechanically isolated. The velocity at the boundary vanishes ($u_t|_{\partial\Omega} = 0$), and the boundary integral in 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 (up to solver tolerance).

- This result confirms that our implementation is **strictly non-dissipative** (symplectic).
- It proves that there is no artificial numerical damping introduced by the time integration scheme.

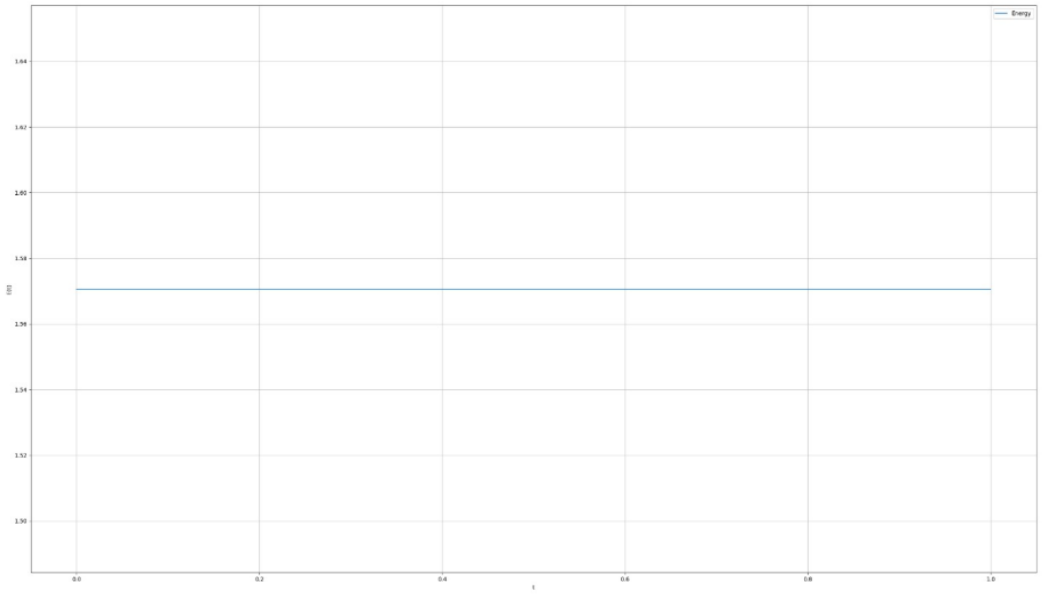

Figure 3.1: **Conservation of Energy in an Isolated System.** The plot shows $E(t)$ versus time for the case with homogeneous Dirichlet boundary conditions ($g = 0$). The horizontal line confirms that the total energy is conserved, validating the symplectic nature of the Newmark scheme ($\gamma = 0.5$).

Case 2: Driven System (Project Scenario)

Hypothesis: In the project scenario (e.g., "Pumping Wall"), the boundary moves according to a time-dependent function $u = g(t)$. Consequently, the velocity at the boundary is non-zero ($u_t = g'(t)$). The energy balance equation becomes:

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

Physically, this term represents the **mechanical power** injected into (or extracted from) the system by the moving boundary.

Numerical Result: Figure 3.2 displays the energy evolution for the driven case.

- The energy is **not constant**, but fluctuates significantly.
- Crucially, thanks to the validation in Case 1, we know these fluctuations are **not numerical errors**. They accurately represent the physical work done by the boundary forces pumping energy into the membrane.

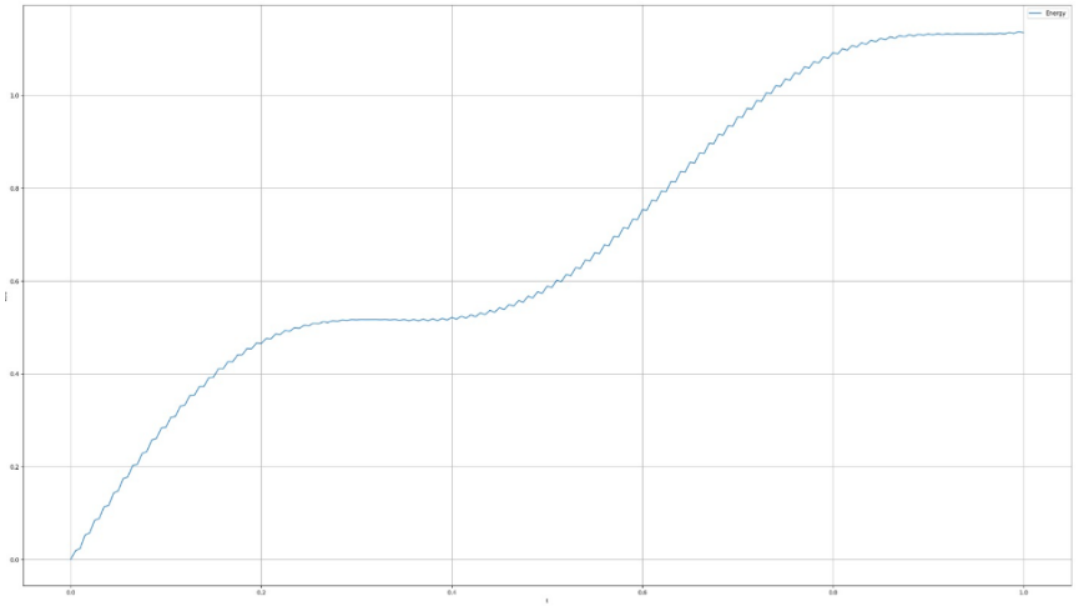

Figure 3.2: **Energy Evolution in a Driven System.** The plot shows $E(t)$ for the case with time-dependent boundary conditions ($g(t) \neq 0$). The variation in energy is physically expected and corresponds to the work performed by the boundary on the system.

Chapter 4

General Conclusions

Note: The following section provides a structured preliminary outline of the final discussion. Precise numerical data, energy plots, convergence graphs, and detailed quantitative performance measurements will be incorporated in the final version of the report.

4.0.1 Implicit Method: Accuracy and Stability

The serial implementation of the implicit time integration scheme demonstrated strong stability and convergence properties. For a quadratic finite element discretization (`fe = 2`), the method consistently achieved an error magnitude in the range 10^{-6} to 10^{-7}, provided that the spatial mesh and the time step were sufficiently refined.

From a theoretical standpoint, implicit schemes for second-order hyperbolic problems are unconditionally stable with respect to the time step size. This stability was clearly reflected in the numerical results, as no spurious oscillations or instabilities were observed, even for relatively large time steps.

The energetic analysis further confirmed the correctness of the implementation:

- In the homogeneous case, where both the forcing term f and the boundary conditions are set to zero, the total discrete energy remained constant over time. This behavior is consistent with the energy conservation property of the continuous wave equation.
- When a non-zero forcing term was introduced, the energy evolution closely followed the temporal behavior of f, in agreement with the relation

$$\frac{dE}{dt} = \mathbf{v}^T \mathbf{f},$$

  which describes the rate at which energy is injected into the system.
- In the presence of time-dependent boundary conditions, the energy exhibited a monotonic increase, indicating that the boundary actively pumps energy into the computational domain.

Overall, the implicit method produced results in excellent agreement with theoretical expectations, providing a robust and reliable reference solution.

4.0.2 Explicit Method: Efficiency and Physical Consistency

As anticipated, the explicit time integration scheme proved to be significantly more efficient than the implicit one. By avoiding the solution of large linear systems at each time step, the explicit method drastically reduced the computational cost and execution time.

Despite this increase in efficiency, the explicit scheme maintained an accuracy level comparable to that of the implicit method, provided that the Courant–Friedrichs–Lewy (CFL) condition

was respected. The observed error magnitude and convergence behavior were consistent with second-order accuracy in time.

The energy analysis yielded results analogous to those obtained with the implicit method:

- In conservative scenarios (zero forcing and homogeneous boundary conditions), the total energy remained approximately constant, with small oscillations attributable to discretization errors.
- In non-conservative cases, the energy evolution correctly reflected the influence of external forcing or boundary excitation.

These observations confirm that the explicit method, while conditionally stable, preserves the essential physical properties of the wave equation when applied within its stability limits, making it particularly suitable for large-scale simulations where performance is critical.

4.0.3 Impact of MPI Parallelization

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.

Further scalability studies, involving a larger number of MPI processes, are planned in order to quantitatively assess speedup, efficiency, and strong and weak scaling behavior.

4.0.4 Convergence Analysis

A systematic convergence study confirmed that the implemented numerical methods exhibit second-order accuracy. This behavior is consistent with both the temporal discretization schemes employed and the spatial finite element approximation.

In particular, when the spatial discretization was sufficiently refined and the time step appropriately chosen, the numerical error decreased quadratically with respect to the discretization parameter, validating the theoretical convergence rate of the methods.

4.0.5 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.

Bibliography

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

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