Project 2: Solving the Wave Equation

Analisi Matematica, Numerica e Strategie Implementative

Oleg Nedina , Nicolò Buttini , Diana Bonalumi

24 novembre 2025

Indice

Capitolo 1

Introduzione e Definizione del Problema

1.1 Obiettivo del Progetto

Il progetto richiede l'implementazione di un solutore agli Elementi Finiti (FEM) per l'equazione delle onde in un dominio bidimensionale Ω, come specificato nella traccia ufficiale *projects.pdf*. L'obiettivo è duplice: simulare la propagazione ondosa sotto condizioni al contorno variabili nel tempo e condurre un'analisi comparativa tra diversi schemi di integrazione temporale (Implicito ed Esplicito), valutandone stabilità, proprietà dispersive e scalabilità parallela (MPI).

1.2 Il Modello Matematico (Problema Forte)

Consideriamo l'equazione delle onde scalare con velocità di propagazione normalizzata $c = 1$. Il problema ai valori iniziali e al contorno (IBVP) è definito come:

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - \Delta u = f & \text{in } \Omega \times (0, T] \\ u = g(t) & \text{su } \partial\Omega \times (0, T] \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}) & \text{in } \Omega \\ \frac{\partial u}{\partial t}(\mathbf{x}, 0) = u_1(\mathbf{x}) & \text{in } \Omega \end{cases} \tag{1.1}$$

Dove f è il termine forzante, $g(t)$ rappresenta la condizione di Dirichlet non omogenea (time-dependent), e u_0, u_1 sono rispettivamente lo spostamento e la velocità iniziali. Come notato in *Salsa, Partial Differential Equations in Action*, la presenza di termini non omogenei al bordo richiede un trattamento specifico sia nell'analisi energetica che nella formulazione numerica (Lifting).

Capitolo 2

Analisi Fisica e Proprietà Iperboliche

2.1 Velocità Finita e Coni di Luce

L'equazione (1.1) è il prototipo delle equazioni alle derivate parziali iperboliche. A differenza delle equazioni paraboliche (come quella del calore), dove l'informazione si propaga istantaneamente, l'equazione delle onde presenta una **velocità di propagazione finita**.

Come dettagliato nel documento *Light Cones*, definiamo il **Cono di Luce** (o cono caratteristico) con vertice in $(\mathbf{x}_0, t_0)$ come l'insieme dei punti $(\mathbf{x}, t)$ tali che:

$$|\mathbf{x} - \mathbf{x}_0| \leq c|t - t_0|$$

Questo concetto definisce la struttura causale del problema:

- **Dominio di Dipendenza:** Il valore della soluzione in un punto $(\mathbf{x}, t)$ dipende esclusivamente dai dati iniziali contenuti nella base del cono passato intersecata con il piano $t = 0$.
- **Dominio di Influenza:** Una perturbazione in $(\mathbf{x}_0, 0)$ si propaga solo all'interno del cono futuro. I punti esterni al cono non avvertono alcun effetto fino a quando l'onda non li raggiunge.

Osservazione 2.1 (Implicazione Numerica)**.** *Questa proprietà fisica è la base teorica della condizione **CFL** (Courant-Friedrichs-Lewy) per gli schemi numerici espliciti. Affinché uno schema sia stabile, il "cono di dipendenza numerico" deve contenere interamente il "cono di dipendenza fisico". Se Δt è troppo grande, il dominio numerico è troppo stretto e si perde informazione fisica, causando instabilità (esplosione della soluzione).*

2.2 Analisi Energetica e Conservazione

L'energia totale del sistema al tempo t è data dalla somma dell'energia cinetica e dell'energia potenziale elastica:

$$E(t) = \frac{1}{2}\|u_t(t)\|^2_{L^2(\Omega)} + \frac{1}{2}\|\nabla u(t)\|^2_{L^2(\Omega)} \tag{2.1}$$

2.2.1 Dimostrazione della Conservazione (Caso Omogeneo)

Seguendo la derivazione presente in *Light Cones and Energy Conservation*, dimostriamo che per un sistema isolato ($f = 0, g = 0$), l'energia si conserva. Deriviamo $E(t)$ rispetto al tempo:

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

Sostituendo l'equazione delle onde $u_{tt} = \Delta u$ nel primo integrale:

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

Applicando la prima identità di Green al termine $\int u_t \Delta u$:

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

Sostituendo nell'espressione della derivata:

$$\frac{dE}{dt} = \left(-\int_\Omega \nabla u_t \cdot \nabla u + \int_\Omega \nabla u \cdot \nabla u_t \right) + \int_{\partial\Omega} u_t \frac{\partial u}{\partial n} \, d\sigma$$

I termini tra parentesi si cancellano. Rimane il termine di bordo. Se $u|_{\partial\Omega} = 0$, allora anche la velocità al bordo $u_t|_{\partial\Omega} = 0$, annullando l'integrale.

$$\Rightarrow \frac{dE}{dt} = 0 \quad \text{(Energia Costante)}$$

2.2.2 Caso Non Omogeneo (Il nostro progetto)

Nel nostro caso specifico, $u = g(t)$ sul bordo, quindi $u_t = g'(t) \neq 0$.

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

Il sistema non è conservativo: l'energia varia a causa del lavoro compiuto dalle forze esterne al bordo. **Nota di Implementazione:** Utilizzeremo comunque il calcolo dell'energia come strumento di debugging. Impostando temporaneamente $g = 0$ e $f = 0$, verificheremo che il nostro solver conservi l'energia numerica (a meno di errori di dispersione temporale).

Capitolo 3

Discretizzazione Spaziale (FEM)

3.1 Formulazione Debole

Introduciamo lo spazio di Sobolev $V = H^1(\Omega)$ e il sottospazio $V_0 = H_0^1(\Omega)$ delle funzioni nulle al bordo. Moltiplicando l'equazione forte per una funzione test $v \in V_0$ e integrando per parti, otteniamo:

$$(\ddot{u}, v) + (\nabla u, \nabla v) = (f, v) \quad \forall v \in V_0 \tag{3.1}$$

dove $(\cdot, \cdot)$ indica il prodotto scalare L^2.

3.2 Lifting e Sistema Semidiscreto

Poiché $u \notin V_0$, decomponiamo la soluzione come $u(t) = u_0(t) + u_g(t)$, dove $u_0 \in V_0$ è l'incognita e u_g è un rilevamento dei dati al bordo. Discretizzando con una base FEM $\{\varphi_j\}$, otteniamo il sistema matriciale:

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F} \tag{3.2}$$

3.2.1 Gestione Algebrica dei Vincoli (Partitioning)

Per implementare correttamente la condizione $u = g(t)$, partizioniamo i gradi di libertà in Interni ($\mathcal{I}$) e Bordo (Γ):

$$\begin{bmatrix} \mathbf{M}_{\mathcal{I}\mathcal{I}} & \mathbf{M}_{\mathcal{I}\Gamma} \\ \mathbf{M}_{\Gamma\mathcal{I}} & \mathbf{M}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{U}}_{\mathcal{I}} \\ \ddot{\mathbf{U}}_{\Gamma} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{\mathcal{I}\mathcal{I}} & \mathbf{K}_{\mathcal{I}\Gamma} \\ \mathbf{K}_{\Gamma\mathcal{I}} & \mathbf{K}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{\mathcal{I}} \\ \mathbf{U}_{\Gamma} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathcal{I}} \\ \mathbf{F}_{\Gamma} \end{bmatrix}$$

Poiché conosciamo $\mathbf{U}_\Gamma = g(t)$ e $\ddot{\mathbf{U}}_\Gamma = \ddot{g}(t)$, spostiamo i termini noti a destra (Lifting Algebrico) e risolviamo solo per $\mathbf{U}_\mathcal{I}$:

$$\mathbf{M}_{\mathcal{I}\mathcal{I}}\ddot{\mathbf{U}}_{\mathcal{I}} + \mathbf{K}_{\mathcal{I}\mathcal{I}}\mathbf{U}_{\mathcal{I}} = \mathbf{F}_{\mathcal{I}} - \left(\mathbf{K}_{\mathcal{I}\Gamma}\mathbf{U}_{\Gamma} + \mathbf{M}_{\mathcal{I}\Gamma}\ddot{\mathbf{U}}_{\Gamma}\right) \tag{3.3}$$

In `deal.II`, questo processo è automatizzato dalla classe `AffineConstraints`, che modifica la matrice di sistema e il vettore RHS ("condensing") prima della risoluzione.

Capitolo 4

Discretizzazione Temporale: Metodo di Newmark

Per l'integrazione temporale, adottiamo la famiglia di schemi **Newmark-β**, ampiamente discussa in *Quarteroni, Numerical Models for Differential Problems* (Cap. 13).

4.1 Formulazione Generale

Siano $\mathbf{U}_n, \dot{\mathbf{U}}_n, \ddot{\mathbf{U}}_n$ le approssimazioni al tempo t_n. Lo schema definisce lo stato al tempo t_{n+1} come:

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \Delta t \dot{\mathbf{U}}_n + \frac{\Delta t^2}{2}\left[(1-2\beta)\ddot{\mathbf{U}}_n + 2\beta\ddot{\mathbf{U}}_{n+1}\right] \tag{4.1}$$

$$\dot{\mathbf{U}}_{n+1} = \dot{\mathbf{U}}_n + \Delta t\left[(1-\gamma)\ddot{\mathbf{U}}_n + \gamma\ddot{\mathbf{U}}_{n+1}\right] \tag{4.2}$$

Combinando queste con l'equazione di equilibrio al tempo $n+1$:

$$\mathbf{M}\ddot{\mathbf{U}}_{n+1} + \mathbf{K}\mathbf{U}_{n+1} = \mathbf{F}_{n+1} \tag{4.3}$$

4.2 Derivazione del Sistema Risolvente

Per implementare il metodo, dobbiamo isolare l'incognita primaria. Se $\beta > 0$, il metodo è implicito. Ricaviamo $\ddot{\mathbf{U}}_{n+1}$ dalla (4.1):

$$\ddot{\mathbf{U}}_{n+1} = \frac{1}{\beta\Delta t^2}(\mathbf{U}_{n+1} - \mathbf{U}_n - \Delta t\dot{\mathbf{U}}_n) - \frac{1-2\beta}{2\beta}\ddot{\mathbf{U}}_n$$

Sostituendo nell'equazione di equilibrio (4.3), otteniamo un sistema lineare nella forma $\mathbf{A}_{sys}\mathbf{U}_{n+1} = \mathbf{b}$:

$$\underbrace{\left(\frac{1}{\beta\Delta t^2}\mathbf{M} + \mathbf{K}\right)}_{\mathbf{A}_{sys}}\mathbf{U}_{n+1} = \mathbf{F}_{n+1} + \mathbf{M}\left[\frac{1}{\beta\Delta t^2}\mathbf{U}_n + \frac{1}{\beta\Delta t}\dot{\mathbf{U}}_n + \frac{1-2\beta}{2\beta}\ddot{\mathbf{U}}_n\right] \tag{4.4}$$

4.3 Strategie Implementative a Confronto

4.3.1 Metodo A: Implicito ($\beta = 0.25, \gamma = 0.5$)

Corrisponde alla regola del trapezio (Constant Average Acceleration).

- **Stabilità:** Incondizionatamente stabile (A-stable). Nessun vincolo su Δt.

- **Proprietà:** Conservativo (simplettico) se $\gamma = 0.5$. Non introduce smorzamento numerico.
- **Costo:** Richiede la risoluzione di un sistema lineare ad ogni passo. In parallelo, usiamo il solver CG precondizionato con **AMG** (Algebraic Multigrid) tramite i wrapper Trilinos.

4.3.2 Metodo B: Esplicito ($\beta = 0, \gamma = 0.5$)

Corrisponde alle Differenze Centrali (Leap-frog).

- **Stabilità:** Condizionata dalla CFL: $\Delta t \leq h_{min}/c$.
- **Mass Lumping:** Ponendo $\beta = 0$, non possiamo dividere per zero nella formula implicita. Usiamo invece l'equazione del moto diretta: $\mathbf{M}\ddot{\mathbf{U}} = \mathbf{F} - \mathbf{K}\mathbf{U}$. Per evitare di invertire $\mathbf{M}$, la approssimiamo con una matrice diagonale $\mathbf{M}_L$ (Mass Lumping). L'inversione diventa banale (divisione scalare).
- **Efficienza:** Costo per passo bassissimo, ma richiede molti passi piccoli. Scalabilità MPI eccellente (poche comunicazioni).

Capitolo 5

Algoritmi e Pseudo-Codice

5.1 Inizializzazione Consistente

Un dettaglio cruciale spesso trascurato è il calcolo dell'accelerazione iniziale. I dati forniscono solo u_0 e v_0. Risolviamo al tempo $t = 0$:

Algorithm 1 Calcolo Accelerazione Iniziale

1: Assembla RHS: $\mathbf{b}_0 = \mathbf{F}(0) - \mathbf{K}\mathbf{U}_0$
2: Risolvi $\mathbf{M}\ddot{\mathbf{U}}_0 = \mathbf{b}_0$ (Usa CG o inversione Lumped a seconda del metodo)

5.2 Algoritmo Implicito (Caso A)

Algorithm 2 Newmark Implicito Solver

1: Assembla matrice $\mathbf{A}_{sys}$ (una tantum se Δt costante)
2: **while** $t < T_{end}$ **do**
3: $\quad t \leftarrow t + \Delta t$
4: $\quad$ **1. Predizione:** Calcola predittori $\mathbf{U}^{pred}, \dot{\mathbf{U}}^{pred}$ dai dati storici.
5: $\quad$ **2. Costruzione RHS:** $\mathbf{b} \leftarrow \mathbf{F}_{n+1}$ + termini inerziali predetti.
6: $\quad$ **3. Applicazione Vincoli:** Applica $g(t)$ a $\mathbf{b}$ e $\mathbf{A}_{sys}$ (condensing).
7: $\quad$ **4. Risoluzione:** Solver CG($\mathbf{A}_{sys}, \mathbf{U}_{n+1}, \mathbf{b}$) fino a convergenza residuo.
8: $\quad$ **5. Correzione:** Aggiorna $\ddot{\mathbf{U}}_{n+1}$ e $\dot{\mathbf{U}}_{n+1}$ usando le formule di Newmark.
9: **end while**

5.3 Algoritmo Esplicito (Caso B)

Algorithm 3 Newmark Esplicito (Mass Lumped)

Costruisci $\mathbf{M}_L$ (Diagonale) usando quadratura Lobatto.
Calcola Δt_{CFL} basato sulla mesh minima.
while $t < T_{end}$ **do**
    $t \leftarrow t + \Delta t$
    1. Update Posizione: $\mathbf{U}_{n+1} \leftarrow \mathbf{U}_n + \Delta t \dot{\mathbf{U}}_n + \frac{\Delta t^2}{2} \ddot{\mathbf{U}}_n$
    2. Imposizione Forte: $\mathbf{U}_{n+1}|_\Gamma \leftarrow g(t)$ (sovrascrivi valori bordo).
    3. Calcolo Residuo: $\mathbf{R} \leftarrow \mathbf{F}_{n+1} - \mathbf{K}\mathbf{U}_{n+1}$ (prodotto matrice-vettore).
    4. Update Accelerazione: $\ddot{\mathbf{U}}_{n+1} \leftarrow \mathbf{M}_L^{-1}\mathbf{R}$ (operazione element-wise).
    5. Update Velocità: $\dot{\mathbf{U}}_{n+1} \leftarrow \dot{\mathbf{U}}_n + \frac{\Delta t}{2}(\ddot{\mathbf{U}}_n + \ddot{\mathbf{U}}_{n+1})$.
end while

Capitolo 6

Strategia di Verifica e Conclusioni

Per validare l'accuratezza del codice, implementeremo una **Manufactured Solution**. Scegliamo una funzione analitica liscia, ad esempio:

$$u_{ex}(x, y, t) = \sin(\pi t) \sin(\pi x) \sin(\pi y)$$

Inserendola nell'equazione forte, ricaviamo i termini forzanti f_{ex} e g_{ex} esatti. Eseguiremo simulazioni su griglie progressivamente raffinate $(h, h/2, h/4)$ calcolando l'errore L^2 al tempo finale:

$$E_h = \|u_h(T) - u_{ex}(T)\|_{L^2(\Omega)}$$

Ci aspettiamo un ordine di convergenza asintotico pari a 2 ($E_h \propto h^2 + \Delta t^2$), confermando la correttezza dell'implementazione FEM lineare e dello schema temporale del secondo ordine.