PROGETTO 3: QUADRATURA AI MINIMI QUADRATI SU DATI SCATTERED APPLICATA ALL'INTEGRAZIONE DELL' EQUAZIONE DELLE ONDE

Sommario. Progetto di calcolo scientifico avanzato: fino a 4 punti bonus.

1. Contesto e fine

L'equazione delle onde è una delle equazioni più importanti della Fisica Matematica perchè (nelle sue varianti) regola tutti i fenomeni ondulatori. La forma più semplice di tale equazione è

$$\begin{cases} \frac{\partial^2}{\partial t^2}u(x,t) = \frac{\partial^2}{\partial t^2}u(x,t) + f(x,t) & x \in]0,2\pi[, t > 0 \\ u(-1,t) = u(1,t) & t > 0 \\ u(x,0) = u_0(x) & x \in]0,2\pi[\\ \frac{\partial}{\partial t}u(x,0) = u_1(x) & x \in]0,2\pi[\end{cases}, \tag{1}$$

dove abbiamo indicato con $\partial^2/\partial_x^2 u(x,t)$ la derivata seconda di u rispetto a x (trattando t come un parametro) e $\partial^2/\partial_t^2 u(x,t)$ quella rispetto a t. Possiamo pensare ad $u(x,t)$ come allo spostamento di un materiale elastico al tempo t nella posizione x quando sottoposto ad una forza f che dipende da tempo ed è distribuita nello spazio.

La condizione $u(-1,t) = u(1,t)$, $\forall t > 0$ è detta condizione di periodicità (scelta qui per semplificare il problema) e induce a pensare u come funzione periodica, ovvero definita su tutto $\mathbb{R}$ replicando il periodo $[0, 2\pi]$ all'infinito.

Ci proponiamo di costruire un programma per l'approssimazione della funzione u quando si dispone di sole misurazioni della funzione f su dati scattered *ovvero la f è stata misurata con sensori posti in posizioni sparse in $]0,2\pi[$ che non sono controllabili dall'utente del software.*

2. Metodi matematici utilizzati

2.1. **Separazione delle variabili.**

Un approssimazione della soluzione può essere costruita seguendo il *metodo della separazione delle variabili.* Possiamo costruire un'approssimazione di $u(\cdot,t) : [0,2\pi] \to \mathbb{R}$ utilizzando la sua *serie di Fourier* troncata, ovvero

$$u(x,t) \approx S_N u(x,t) := \sum_{k=-N}^{N} \hat{u}_k(t) e^{ikx},$$

dove

$$\hat{u}_k(t) := \frac{1}{2\pi}\int_{-1}^{1} u(x,t)e^{-ikx}dx.$$

E' importante notare che l' operazione $u(\cdot,t) \mapsto (\hat{u}_{-N}(t), \hat{u}_{-N+1}(t), \ldots, \hat{u}_{-N}(t))^t$ è la proiezione ortogonale della funzione $u(\cdot,t)$ (u a tempo fissato) sullo spazio dei *polinomi trigonometrici di grado al più N*, rispetto al prodotto

$$(f,g) := \frac{1}{2\pi}\int_0^{2\pi} \bar{f}(x)g(x)dx, \tag{2}$$

rispetto al quale $\{e^{-iNx}, e^{-i(N-1)x}, \dots, e^{-ix}, 1, e^{ix}, \dots, e^{iNx}\}$ sono una base *ortonormale* che viene detta la *base dei caratteri di Fourier*. In altre parole

$$\|u(\cdot,t) - S_N u(\cdot,t)\|_2^2 = \operatorname*{argmin}_{p \in \mathcal{S}_N} \|u(\cdot,t) - p\|_2^2,$$

dove $\|\cdot\|_2$ è la norma indotta da (2) e $\mathcal{S}_N$ è lo spazio dei polinomi trigonometrici di grado al più N.

Seguendo questo possibile schema di approssimazione avremmo

$$\begin{aligned}
\frac{\partial^2}{\partial t^2}u(x,t) \approx & \frac{\partial^2}{\partial t^2}S_N u(x,t) := \sum_{k=-N}^{N} \hat{u}_k''(t)e^{ikx} \\
\frac{\partial^2}{\partial x^2}u(x,t) \approx & \frac{\partial^2}{\partial x^2}S_N u(x,t) := \sum_{k=-N}^{N} \hat{u}_k(t)\frac{\partial^2}{\partial x^2}e^{ikx} = \sum_{k=-N}^{N} (ik)^2 \hat{u}_k(t)e^{ikx} = -\sum_{k=-N}^{N} k^2 \hat{u}_k(t)e^{ikx} \\
u_0(x) \approx & S_N u_0(x) = \sum_{k=-N}^{N} \hat{u}_{k,0}e^{ikx} \\
u_1(x) \approx & S_N u_1(x) = \sum_{k=-N}^{N} \hat{u}_{k,1}e^{ikx} \\
f(x,t) \approx & S_N f(x,t) = \sum_{k=-N}^{N} \hat{f}_k(t)e^{ikx}
\end{aligned}$$

Inserendo tali approssimazioni in (1) si ottiene

$$\begin{cases} \sum_{k=-N}^{N} (\hat{u}_k''(t) + k^2\hat{u}_k(t) - \hat{f}_k(t))e^{ikx} = 0 & t > 0,\ x \in]0, 2\pi[\\ \sum_{k=-N}^{N} (\hat{u}_k(0) - \hat{u}_{k,0})e^{ikx} = 0 & x \in]0, 2\pi[\end{cases},$$

ma, dovendo le relazioni valere per ogni x, otteniamo

$$\begin{cases} \hat{u}_k''(t) + k^2\hat{u}_k(t) - \hat{f}_k(t) = 0 & t > 0 \\ \hat{u}_k'(0) = \hat{u}_{k,1} \\ \hat{u}_k(0) = \hat{u}_{k,0} \end{cases}, \forall k \in \{-N, \dots, N\}. \tag{3}$$

Ovvero i coefficienti del polinomio trigonometrico $S_N u(\cdot,t)$ soddisfano un equazione differenziale ordinaria, lineare del secondo ordine e non omogenea. Con il cambio di variable $\hat{u}_k' = \hat{v}_k$ possiamo riscriverla come sistema di equazioni del primo ordine

$$\begin{cases} \hat{v}_k'(t) = -k^2\hat{u}_k(t) + \hat{f}_k(t) & t > 0 \\ \hat{u}_k'(t) = \hat{v}_k(t) \\ \hat{v}_k(0) = \hat{u}_{k,1} \\ \hat{u}_k(0) = \hat{u}_{k,0} \end{cases}, \forall k \in \{-N, \dots, N\}, \tag{4}$$

ovvero

$$\begin{cases} Y_k'(t) = E_k Y_k(t) + g_k(t) & t > 0 \\ Y_k(0) = Y_{k,0} \end{cases}, \text{ dove} \tag{5}$$

$$Y_{k,0} := \begin{pmatrix} \hat{u}_{k,1} \\ \hat{u}_{k,0} \end{pmatrix}, \quad E_k := \begin{bmatrix} 0 & -k^2 \\ 1 & 0 \end{bmatrix}, \ g_k(t) := \begin{pmatrix} \hat{f}_k(t) \\ 0 \end{pmatrix}, \ Y_k(t) := \begin{pmatrix} \hat{v}_k(t) \\ \hat{u}_k(t) \end{pmatrix}. \tag{6}$$

2.2. **Metodo di Eulero implicito (o all'indietro).** Vogliamo dunque approssimare la soluzione di $Y_k' = E_k Y + g_k$ con il metodo di Eulero implicito. Supponiamo di fissare un passo temporale $h > 0$ (dunque una discretizzazione del tempo come $\{t_0, t_1, \dots, t_n\} := \{j \cdot h, j = 0, 1, 2, \dots, n\}$) e costruiamo un approssimazione $Y_{k,j}$ di $Y_k(t_j)$, $j = 1, 2, \dots, n$ ponendo, $g_{k,j} := g_k(t_j)$ e risolvendo iterativamente

$$\frac{1}{h}(Y_{k,j} - Y_{k,j-1}) = E_k Y_{k,j} + g_{k,j}.$$

Tale metodo è detto implicito perchè stiamo approssimando la forzante con il valore nel nuovo punto da calcolarsi e questo porta ad un equazione da risolvere. Tale equazione nel nostro caso è un semplice sistema lineare che riordinato porta a

$$(\mathbb{I}_2 - hE_k)Y_{k,j} = Y_{k,j-1} + hg_{k,j} \Rightarrow Y_{k,j} = (\mathbb{I}_2 - hE_k)^{-1}(Y_{k,j-1} + hg_{k,j}).$$

Notiamo che tutti i $2N + 1$ sistemi in (5) possono essere risolti come un unico sistema tridiagonale ponendo

$$Z(:, j) := \begin{bmatrix} Y_{-N,j} \\ Y_{-N+1,j} \\ \vdots \\ Y_{N,j} \end{bmatrix}, \; A := \begin{pmatrix} \mathbb{I}_2 - hE_{-N} & 0 & 0 & \dots \\ 0 & \mathbb{I}_2 - hE_{1-N} & 0 & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \mathbb{I}_2 - hE_N \end{pmatrix} =: \mathbb{I}_{2N+1} - hE$$

e risolvendo, per $j = 1, 2, \dots$ $AZ(:, j) = Z(:, j-1) + hg_{:,j}$. Nel nostro caso specifico, vista l'elementarietà dell'inversione di A, potremmo anche calcolare di fatto $Z(:, j) = A^{-1}(Z(:, j-1) + hg_{:,j})$ con

$$A^{-1} = \begin{pmatrix} (\mathbb{I}_2 - hE_{-N})^{-1} & 0 & 0 & \dots \\ 0 & (\mathbb{I}_2 - hE_{1-N})^{-1} & 0 & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & (\mathbb{I}_2 - hE_N)^{-1} \end{pmatrix}.$$

2.3. **Costruzione di quadratiura stabile su dati scattered.** Molto spesso in problemi applicati al mondo reale capita di disporre di dati scattered, privi di reale struttura e non raccolti in modo finalizzato all'uso che se ne vorrebbe fare. Nella nostra applicazione immaginiamo di disporre delle misurazioni

$$F := \{f(x_i, t_j)\}_{i=1,2,\dots,m; j=1,2,\dots,n}, \tag{7}$$

dove assumiamo che $m >> 2N + 1$, e vogliamo un metodo per l'approssimazione di

$$\hat{f}_{k,j} := \frac{1}{2\pi} \int_0^{2\pi} f(x, t_j) e^{-ikx} dx,$$

che pensiamo di raccogliere in una matrice

$$\hat{F} := \{\hat{f}_{k,j}\}_{-N \le k \le N; j=1,2,\dots,n}, \tag{8}$$

Un primo approccio può essere il seguente. Se $\Lambda_N : \mathscr{C}^0_{\text{per}}([0, 2\pi]) \to \mathcal{S}_N$ è la proiezione ai minimi quadrati discreti sui punti $X := \{x_1, \dots, x_m\}$, allora possiamo scrivere

$$\hat{F}_{k,j} = \frac{1}{2\pi} \int_0^{2\pi} f(x, t_j) e^{-ikx} dx \approx \frac{1}{2\pi} \int_0^{2\pi} \Lambda_N f(x, t_j) e^{-ikx} dx = \sum_{l=1}^{\tilde{m}} \Lambda_N f(z_l, t_j) e^{-ikz_l} \tilde{w}_l =: \hat{\mathcal{F}}_{k,j}, \tag{9}$$

dove $(z_1, \dots z_{\tilde{m}})$ sono i nodi e $(\tilde{w}_1, \dots \tilde{w}_{\tilde{m}})$ sono i pesi di una formula di quadratura per $dx/(2\pi)$ su $[0, 2\pi]$.

In realtà abbiamo a disposizione una scelta naturale. Per i polinomi trigonometrici vale un risultato simile a quello di Gauss che in questo caso asserisce che

$$\begin{aligned} z_l :=& \frac{2\pi}{2N+1}(l-1), l = 1, 2, \dots, 2N+2 \\ \tilde{w}_l :=& 1/(2N+2) \end{aligned}$$

sono nodi e pesi di una formula esatta su $\mathcal{S}_{2N}$, ovvero

$$\frac{1}{2\pi}\int_0^{2\pi} s(x)dx = \frac{1}{2N+2}\sum_{l=1}^{2N+2} s(z_l), \quad \forall s \in \mathcal{S}_{2N}.$$

Si noti che per definizione sia $\Lambda_N f(\cdot, t_j)$ che $e^{-ik\cdot}$ sono in $\mathcal{S}_N$ e dunque $\Lambda_N f(\cdot, t_j)e^{-ik\cdot} \in \mathcal{S}_{2N}$ per ogni $k = -N, \dots, N$.

Notiamo però che la (9) non è ancora nella forma di una formula di quadratura. Per poterla scrivere in tal modo dobbiamo osservare quanto segue. Sia V la matrice di Vandermonde delle basi $\{e^{-iNx}, \dots, e^{iNx}\}$ sui punti $\{x_1, \dots, x_m\}$ e $\tilde{V}$ la matrice di Vandermonde della stessa base sui punti $z_1, \dots z_{\tilde{m}}$, allora, denotato con F la matrice $(f(x_i, t_j))_{i_1,\dots,m,j=1,\dots,n}$ abbiamo

$$\hat{\mathcal{F}}_{k,j} = \left(\sum_{l=1}^{2N+2} \Lambda_N f(z_l, t_j)e^{-ikz_l}\tilde{w}_l\right)_{k=-N,\dots,N,j=1,\dots,n} = V^H diag(\tilde{w})\tilde{V}(V^H V)^{-1}V^H F,$$

dove V^H è la trasposta coniugata di V (l'operatore `V'` di matlab fa questo!). Sia ora $QR = V$ la fattorizzazione QR di V e siano Q_0 e R_0 le "parti rilevanti" (vedi minimi quadrati con metodo QR), abbiamo allora (la matrice F è definita in (7))

$$\begin{aligned} \hat{\mathcal{F}} =& \tilde{V}^H diag(\tilde{w})\tilde{V}(V^H V)^{-1}V^H F = \tilde{V}^H diag(\tilde{w})\tilde{V}(R^H Q^H QR)^{-1}R^H Q^H F \\ =& \tilde{V}^H diag(\tilde{w})\tilde{V}(R_0^H Q_0^H Q_0 R_0)^{-1}R_0^H Q_0^H F = \tilde{V}^H diag(\tilde{w})\tilde{V}(R_0^H R_0)^{-1}R_0^H Q_0^H F \\ =& \tilde{V}^H diag(\tilde{w})\tilde{V}R_0^{-1}R_0^{-H}R_0^H Q_0^H F = \tilde{V}^H diag(\tilde{w})\tilde{V}R_0^{-1}Q_0^H F \\ =& \tilde{V}^H diag(\tilde{w})\tilde{V}R_0^{-1}Q_0^H F \\ =& R_0^{-1}Q_0^H F, \end{aligned}$$

dove nell'ultimo passaggio abbiamo ssfruttato il fatto che i caratteri di Fourier sono ortogonali rispetto ad un prodotto che stiamo quadrando esattamente.

Nota: su idee molto simili si basa uno degli algoritmi più usati al mondo, la FFT (fast Fourier Transform).

3. Implementazione di un solutore e codice chiamante

I building blocks del codice saranno dunque:

- una function `Waves_Op` che dato N costruisce la matrice E definita nella sezione 2.1
- una function `DLS_FT` che implementa il calcolo dei coefficienti di Fourier come descritto nella sezione 2.3 (input: F, x, N e output $\hat{\mathcal{F}}$);
- Una function `Backw_EM` che implementa il metodo di Eulero implicito come descritto nella sezione 2.2 (input: $A, Z0$ (vettore colonna di Z al tempo 0) g, output: Z matrice con valore di Z a tutti i tempi considerati
- Una function `Eval_SN` che dati i coefficienti di fourier della soluzione (i $\hat{u}_k$ e i $\hat{v}_k$) valuta spostamento e velocità (ovvero u') ad ogni tempo e ogni punto di una griglia di valutazione assegnata;

- Un main `WAVES_debug` che approssima la soluzione del problema test riportato nella sezione successiva: di tale problema conosciamo la soluzione esatta, possiamo dunque ad esempio plottare soluzione ($u(x,t_j)$ con j fissato) e la sua approssimazione, aggiornando la figura e inserendo un comando `pause(0.5)` tra un plot e l'altro ottenendo un'animazione. Utilizzare questo main per tarare il passo temporale $h > 0$ (sufficientemente piccolo).
- Un main `WAVES_SD` che legge i dati scattered in input e integra l'equazione delle onde facendo possibilmente un animazione della soluzione calcolata. Per caricare correttamente i files forniti dal docente iniziare `WAVES_SD` con le seguenti righe:

```
load U0
load V0
load F
load t_samples load theta_samples.mat
Z0mat=[conj(V0)';conj(U0)']; Z0=Z0mat(:);
```

4. Sperimentazione

Possiamo considerare soluzioni test del tipo

$$u(x,t) := e^{-t}\sin(Ax - Bt), \quad A, B \in \mathbb{R},$$

che sono onde che transitano a velocità B e si smorzano esponenzialmente nel tempo, e ricavare opportuni dati iniziali e funzione f affinchè questa sia la soluzione.

Derivando e valutando otteniamo

$$\begin{aligned}
u(x,0) =& \sin(Ax)\\
\frac{\partial}{\partial t}u(x,0) =& e^{-t}(-\sin(Ax-Bt) - B\cos(Ax-Bt))\Big|_{t=0} = -(\sin(Ax) + B\cos(Ax))\\
\frac{\partial^2}{\partial t^2}u(x,t) =& e^{-t}((1-B^2)\sin(Ax-Bt) + 2B\cos(Ax-Bt))\\
\frac{\partial^2}{\partial x^2}u(x,t) =& e^{-t}A\cos(Ax-Bt),
\end{aligned}$$

quindi, fissati A e B a piacimento $u(x,t) := e^{-t}\sin(Ax-Bt)$ è (unica) soluzione di

$$\begin{cases}
\frac{\partial^2}{\partial t^2}u(x,t) = \frac{\partial^2}{\partial x^2}u(x,t) + e^{-t}((1-B^2)\sin(Ax-Bt) + (2B-A)\cos(Ax-Bt)) & x \in [0,2\pi],\ t > 0\\
u'(x,0) = -(\sin(Ax) + B\cos(Ax)) & x \in [0,2\pi]\\
u(x,0) = \sin(Ax) & x \in [0,2\pi]
\end{cases}.$$

Grazie all'uso di questa soluzione test, possiamo sperimentare il solutore implementato generando punti scattered in $[0,2\pi]$ (ad esempio con la function `rand` di matlab), valutando la forzante

$$f(x,t) := +e^{-t}((1-B^2)\sin(Ax-Bt) + (2B-A)\cos(Ax-Bt))$$

su tali punti e costruendo la soluzione approssimata con le functions sviluppate. A seconda dei valori di A e B potrebbero rendersi necessari passi temporali h anche molto piccoli e $N >> 1$.

I dati scattered forniti sono campionamenti rumorosi di una forzante ottenuta da somme di funzioni del tipo sopra descritto, dunque sperimenteremo il software sui dati reali solo dopo averlo tarato sui dati sintetici.