Appunti Calcolo Numerico

Luca Seggiani
2025

1 Lezione del 24-02-25

1.1 Introduzione al corso

Il corso di **calcolo numerico**, o come viene definito oggi *analisi numerica*, tratta lo studio degli algoritmi per problemi in campi continui (incognite in $\mathbb{R}$, siano queste numeri o funzioni, ecc...) o su grandi moli di dati.

Il programma del corso è così suddiviso:

1. Analisi dell'errore per funzioni scalari;
2. Richiami di algebra lineare (calcolo vettoriale e matriciale, ecc...);
3. Risoluzione di sistemi lineari, cioè forme $A\mathbf{x} = b$;
4. Interpolazione e approssimazione di funzioni nel senso dei minimi quadrati;
5. Metodi per l'integrazione, cioè per forme $\int_a^b f(x)\,dx$;
6. Equazioni non lineari, cioè ricerca dei punti $g(x) = 0$ per $g(x)$ non lineari;
7. Problemi agli autovalori, cioè date matrici $A \in \mathbb{C}^{m\times n}$, trovare $(\lambda, \mathbf{x})$ tali che $A\mathbf{x} = \lambda\mathbf{x}$.

1.1.1 Matematica del continuo

Abbiamo detto che i valori trattati sono continui, che intendiamo per appartenenti ad $\mathbb{R}$. Un problema apparente dei numeri reali è che richiedono teoricamente un numero infinito di cifre per la loro rappresentazione. Si rende quindi necessaria un'approssimazione in modo da ovviare ai problemi:

- Rappresentare oggetti matematici con un numero infinito di parametri;
- Risolvere problemi che non hanno formule chiuse per la soluzione, ma richiedono approcci iterativi (discesa a gradiente, ecc...) e quindi che richiedono un approssimazione data dall'impossibilità di effettuare infiniti passi.

1.1.2 Errori

Avremo quindi bisogno di valutare degli **errori**, che saranno gli *errori di approssimazione* come riportata sopra, uniti agli *errori nei dati* già presenti nella nostra mole di dati.

Una domanda che ptoremo porci è come questi errori influiscono sul risultato che ci interessa. Una prima distinzione può essere fra algoritmi **instabili** e **stabili**, cioè che *amplificano* l'errore o lo mantengono costante. Una seconda distinzione può essere sul **condizionamento** del problema, cioè la tendenza del problema a reagire in maniera drastica a piccole variazioni delle condizioni iniziali.

1.1.3 Efficienza

Un'altra considerazione importante è quella dell'**efficienza** degli algoritmi, cioè il tempo che questi richiedono per convergere ad una soluzione valida (non ottima, in quanto abbiamo visto dobbiamo troncare il numero di passi nel caso di approcci iterativi, che altrimenti potrebbe tendere ad infinito). Questo viene stimato attraverso il *costo computazionale*, tenendo conto di una certa *accuratezza* che vogliamo stabilire.

1.2 Rappresentazione dei reali

Introduciamo la classe dei **numeri reali in virgola mobile**, atta a rappresentare numeri $x \in \mathbb{R}$ usando parametri che stanno in $\mathbb{N}$, o al limite in $\mathbb{Z}$, in quanto abbiamo visto possiamo gestire numeri di questo tipo in modo esatto nei calcolatori.

Teorema 1.1: Rappresentazione dei reali in virgola mobile

Fissata una base $\beta > 1$, con $\beta \in \mathbb{N}$, si può sempre trovare un esponente $e \in \mathbb{Z}$ e una successione di cifre $\{\alpha_j\}_{j=1,2,...}$ tali per cui:

$$x = \text{sgn}(x) \cdot \beta^e \cdot \sum_{j=1}^{\infty} \alpha_j \beta^{-j}$$

Osserviamo che scelte tipiche per β sono 10 (decimale), 2 (binario) e 16 (esadecimale).

Un problema di questa rappresentazione è che non è propriamente **unica**: ad esempio, potremo scrivere un approssimazione di π come 3.14, o come $0.314 \cdot 10$, equivalentemente. Inoltre, possono esistere casi di numeri periodici, come $2.\overline{9}$, che sono effettivamente uguali ad altri (in questo caso 3).

Sfruttiamo allora il seguente teorema, dato senza dimostrazione:

Teorema 1.2: Teorema di rappresentazione

Dati $\beta \in \mathbb{N}, \beta > 1$ base e $x \in \mathbb{R}, x \neq 0$, allora esiste ed è unica la rappresentazione $x = \text{sgn}(x) \cdot \beta^e \cdot \sum_{j=1}^{\infty} \alpha_j \beta^{-j}$ (dal Teorema 1.1) tale che:

- $\alpha_1 \neq 0$;
- $!\exists k \in \mathbb{N} : \alpha_j = \beta - 1 \; \forall j > k$.

Introducendo queste due limitazioni possiamo quindi ovviare al problema dell'unicità.

Diamo quindi alcune definizioni, riguardo alla rappresentazione appena vista:

Definizione 1.1: Rappresentazione in virgola mobile normalizzata

L'unica rappresentazione che verifica il Teorema 1.2 si dice **rappresentazione in virgola mobile normalizzata**.

e riguardo alla successione $\{\alpha_j\}$:

Definizione 1.2: Mantissa

Prende il nome di **mantissa** la serie $\sum_{j=1}^{\infty} \alpha_j \beta^{-j}$.

Notiamo che vale:

$$\frac{1}{\beta} \leq \sum_{j=1}^{\infty} \alpha_j \beta^{-j} < 1$$

In quanto:

- Per il limite inferiore:

$$\sum_{j=1}^{\infty} \alpha_j \beta^{-j} \geq \beta^{-1} = \frac{1}{\beta}$$

- Per il limite superiore:

$$\sum_{j=1}^{\infty} \alpha_j \beta^{-j} < (\beta - 1) \sum_{j=1}^{\infty} \beta^{-j} = (\beta - 1) \left(\frac{1}{1 - \beta^{-1}} - 1 \right)$$

$$= (\beta - 1) \frac{\beta^{-1}}{1 - \beta^{-1}} = (\beta - 1) \frac{1}{\beta - 1} = 1$$

Vediamo che la mantissa non è effettivamente rappresentabile nella sua interezza in un calcolatore, in quanto richiederebbe un numero infinito di cifre. Si tronca quindi la mantissa, e si considera un range limitato per l'esponente. Si definisce quindi un sottinsieme:

Definizione 1.3: Numeri di macchina

Dati: $\beta \in \mathbb{N}$, $m \in \mathbb{N}$, $L, U \in \mathbb{Z}$ tali che $L \leq U$, si definisce $F(\beta, m, L, U)$ come:

$$F = \{x \in \mathbb{R} : \text{sgn}(x) \cdot \beta^e \cdot \sum_{j=1}^{m} \alpha_j \beta^{-j},\ L \leq e \leq U,\ \alpha_j = \{0, ..., \beta - 1\},\ \alpha_j \neq 0\} \cup \{0\}$$

detto **numero di macchina**.

L'inclusione dello zero è necessario in quanto questo non è compreso nel primo insieme dato.

Notiamo che l'insieme dei numeri di macchina non è equispaziato (ci sono più numeri vicino allo zero). Presi intervalli $[\beta^{e-1}, \beta^e)$, questi risulterebbero equispaziati se venissero considerati su una scala logaritmica base β. All'interno di questi intervalli, invece, si hanno effettivamente numeri equispaziati, con periodo $\beta^e \cdot \beta^{-m} = \beta^{e-m}$.

1.2.1 Limiti inferiori e superiori

Potremmo chiederci quali sono i numeri **minimi** e **massimi** rappresentabili.

Osservando la definizione di F si ha che il numero più piccolo possibile è quello che si ha prendendo $\beta = L$, e $\alpha_j = 0$ per ogni $j > 1$, e $\alpha_1 = 1$, quindi:

$$\beta^L \cdot 1 \cdot \beta^{-1} = \beta^{L-1}$$

Il numero più grande si ha invece prendendo $\beta = U$ e $\alpha_j = \beta - 1$, e quindi:

$$\beta^U(\beta-1)\sum_{j=1}^{m}\beta^{-j} = \beta^U\left(\frac{1-\beta^{-m-1}}{1-\beta^{-1}} - 1\right)(\beta-1)$$

$$= \beta^U\left(\frac{1-\beta^{-m}}{\beta-1}\right)(\beta-1) = \beta^U(1-\beta^{-m})$$

Potremmo poi chiederci quanti numeri macchina esistono fissati β, m, L e U. Si hanno intanto due scelte di segno, $(U - L + 1)$ scelte di esponenti e $(\beta^m - \beta^{m-1})$ scelte di mantisse più lo zero, da cui:

$$2 \cdot (U - L + 1) \cdot (\beta^m - \beta^{m-1}) + 1$$

Le rappresentazioni più comuni dei numeri macchina sono definite dallo standard IEEE 754, che definisce:

Precisione	β	m	L	U	**Dimensione**
Singola	2	24	-125	128	4 byte (32 bit)
Doppia	2	53	-1021	1024	8 byte (64 bit)
Quadrupla	2	113	-16381	16384	8 byte (64 bit)

Casi particolari potrebbero richiedere precisioni più grandi o più lasche. Ultimamente, in particolare, si sono diffuse rappresentazioni a precisione più bassa (su 16 o addirittura 8 bit), in particolare nel campo delle reti neurali.

Il pacchetto MATLAB utilizza di default la precisione **doppia** come definita dallo standard IEEE 754.

1.3 Arrotondamento

Abbiamo visto come ci stiamo spostando dalla matematica esatta a una serie di approssimazioni. In generale, vorremo partire da un certo numero reale $x \in \mathbb{R}$, ma non appartenente all'insieme dei numeri macchina ($x \notin F(\beta, m, L, U)$). L'obiettivo è quello di riportare x ad una sua *approssimazione* appartenente ad $F(\beta, m, L, U)$.

Esistono 3 possibili situazioni:

- $|x| > \beta^U(1 - \beta^{-m})$, cioè x maggiore del massimo rappresentabile (**overflow**);
- $|x| < \beta^{L-1}$, cioè x minore del minimo rappresentabile (**underflow**);
- $\beta^{L-1} \leq x \leq \beta^U(1 - \beta^{-m})$. Se nei casi precedenti si poteva scegliere, rispettivamente, un M molto grande, o ∞, e 0, qui si può effettivamente procedere con l'arrotondamento.

Definizione 1.4: Arrotondamento

Un arrotondamento è una funzione $RD : \mathbb{R} \to F(\beta, m, L, U)$, con $RD(x)$ uno dei numeri di macchina in $F(\beta, m, L, U)$ "vicini" ad x.

Preso un certo reale x, avremo un numero di macchina a sinistra è uno a destra, cioè il primo più piccolo e il primo più grande. Possiamo allora definire i seguenti arrotondamenti:

- **Troncamento** (*round down*):
$$Tr(x) = \lfloor x \rfloor$$
- **Round-up:**
$$Ru(x) = \lceil x \rceil$$
- **Round-to-nearest:**
$$Rn(x) = \begin{cases} \lceil x \rceil, & \alpha_{m+1} \geq \frac{\beta}{2} \\ \lfloor x \rfloor, & \alpha_{m+1} < \frac{\beta}{2} \end{cases}$$
con α_{m+1} la prima cifra che viene scartata dall'arrotondamento.

Notiamo poi che preso il troncamento $\text{sgn}(x)\beta^e \sum_{j=1}^m \alpha_j \beta^{-j}$, il round up corrispondente sarà $\text{sgn}(x)\beta^e \left(\sum_{j=1}^m \alpha_j \beta^{-j} + \beta^{-m}\right)$.

1.3.1 Errore di arrotondamento

Notiamo che valgono le disuguaglianze:

- $|Tr(x) - x| \leq \beta^{e-m}$
- $|Rn(x) - x| \leq \frac{\beta^{e-m}}{2}$, che è il minimo errore possibile cioè:
$$|Rn(x) - x| = \min_{RD(x)}(RD(x) - x)$$
Per questo motivo da qui in poi assumeremo quindi di prendere sempre $Rn(x)$.

Definizione 1.5: Errori di arrotondamento

Definiamo:

- **Errore assoluto di arrotondamento:** $x - Rn(x) = \sigma_x$
- **Errore relativo di arrotondamento:** $\frac{x - Rn(x)}{x} = \epsilon_x$

La definizione di errore relativo è utile per avere un errore pressochè costante su tutta la retta dei reali. Si ha quindi che:

- Riguardo all'errore assoluto:
$$|\sigma_x| \leq \frac{\beta^{e-m}}{2}$$
- Riguardo all'errore relativo:
$$|\epsilon_x| \leq \frac{\beta^{e-m}}{2\beta^{e-1}} = \frac{1}{2}\beta^{1-m}$$
visto che $|x| > \beta^{e-1}$. Notiamo che questo errore non dipende dall'esponente e, che è quello che desideravamo.

Abbiamo quindi che l'insieme dei numeri in virgola mobile garantiscono un errore relativo limitato in modo uniforme, se non si incombe in overflow o underflow.

Definizione 1.6: Precisione

La quantità $U = \beta^{1-m}$ viene detta **precisione di macchina** di una certa rappresentazione a virgola mobile.

Ad esempio, nella precisioni doppia e singola, $U \approx 10^{-16}$ e $U \approx 10^{-8}$, che significa rispettivamente prime 15 o prime 7 cifre esatte.

1.3.2 Dettagli di implementazione

Prendiamo ad esempio la doppia precisione. Avremo:

- 1 bit di segno;
- 52 bit di mantissa (con 1 bit implicito impostato ad 1, per $\alpha_1 = 1$);
- 11 bit di esponente in rappresentazione con offset.

Riguardo agli altri formati si ha:

Precisione	Segno	Esponente	Mantissa	U
Singola	1	8	23	$\approx 10^{-8}$
Doppia	1	11	52	$\approx 10^{-16}$
Quadrupla	1	15	112	$\approx 10^{-34}$

1.4 Numeri sottonormalizzati

Vediamo una tecnica per la rappresentazione di numeri più piccoli di β^{L-1}, implementata nella maggior parte dei pacchetti software moderni (e nelle implementazioni degli standard a virgola mobile disponibili nei processori moderni). Si ha che se $x \in [0, \beta^{L-1}]$, allora si assume come prima cifra della mantissa 0, con esponente fisso a L. Questo ci permette di rappresentare più numeri vicino allo zero.

Si avranno quindi numeri equispaziati fra 0 e β^{L-1} con distanza β^{L-m}. Inoltre, sui numeri sottonormalizzati si avrà un errore relativo non limitato da $\frac{1}{2}\beta^{1-m}$, ma che invece aumenta avvicinandosi a 0.

I numeri sottonormalizzati sono indicati da un **valore speciale dell'esponente**, cosa che accade anche per:

- I valori $+\infty$ e $-\infty$;
- I valori NaN (Not a Number) con relativi codici di errore in mantissa.

La presenza di questi valori rende necessaria l'approssimazione delle precisioni per i numeri in virgola mobile.