Appunti di Calcolo Numerico

Indice

1 I numeri macchina

I numeri macchina sono i numeri rappresentabili esattamente al calcolatore (IEEE Standard 754 for double precision). L'insieme dei numeri macchina è chiamato Sistema Floating Point e denotato $\mathbb{F}$.
In Matlab ogni numero è rappresentabile con 64 bit.

$$x = s \cdot (\frac{d_1}{\beta} + \frac{d_2}{\beta} + \cdots + \frac{d_t}{\beta}) \cdot \beta^p$$

dove:

- s è il segno del numero, che può essere $+1$ o -1. 1 bit
- $d_1, \ldots, d_t$ rappresenta la mantissa del numero. d_t sono le cifre della mantissa, β è la base del sistema di numerazione (tipicamente 2 per la rappresentazione binaria), e t è il numero totale di cifre nella mantissa. 52 bit
- β^p è l'esponente del numero, che scala il valore rappresentato dalla mantissa. p è un intero che può assumere valori positivi o negativi, permettendo di rappresentare numeri molto grandi o molto piccoli. $p \in [-1022, 1023]$: 11 bit
- $i = 1 \ldots t,\ 0 \leq d_i \leq \beta - 1$

La rappresentazione in virgola mobile consente una vasta gamma di valori da rappresentare con una precisione relativa costante. Permette di gestire numeri estremamente grandi o piccoli. Nonostante i suoi vantaggi, la rappresentazione in virgola mobile ha alcune limitazioni:

- Precisione finita: Con un numero limitato di bit per la mantissa, solo certi numeri reali possono essere rappresentati esattamente. Questo può portare a errori di arrotondamento.
- Numeri speciali: Il formato IEEE 754 include rappresentazioni per "infinito" (positivo e negativo) e "NaN" (Not a Number) per gestire situazioni particolari come divisioni per zero e radici quadrate di numeri negativi.

L'efficienza e la flessibilità della rappresentazione in virgola mobile rendono il formato ideale per molte applicazioni di calcolo, nonostante queste limitazioni.

1.1 Realmin e Realmax

In MATLAB, realmax e realmin sono due funzioni che restituiscono, rispettivamente, il valore massimo e minimo rappresentabile in virgola mobile nella precisione utilizzata. Questi valori rappresentano i limiti per i numeri in virgola mobile che MATLAB può gestire senza ricorrere all'infinito o allo zero, rispettivamente. Il comando **realmax** restituirà il massimo numero rappresentabile, che è circa 1.7977×10^{308} per la doppia precisione. Ciò significa che qualsiasi valore superiore a **realmax** verrà interpretato da MATLAB come infinito (**Inf**).

Al contrario, **realmin** restituisce il più piccolo valore positivo normalizzato rappresentabile in MATLAB per un numero in virgola mobile a doppia precisione, ovvero 2.2251×10^{-308}. I valori inferiori a **realmin** sono considerati "denormalizzati", il che significa che non possono essere rappresentati con piena precisione a causa della limitazione nei bit disponibili per la mantissa.

Conoscere i limiti realmax e realmin aiuta a progettare algoritmi robusti che evitano automaticamente problemi di overflow e underflow, garantendo che i risultati dei calcoli rimangano nel range di valori che MATLAB può gestire in modo affidabile.

1.2 Epsilon

In MATLAB, **eps** rappresenta la distanza tra 1 e il più piccolo valore maggiore di 1 che è rappresentabile nella precisione di calcolo utilizzata. È una funzione di MATLAB che fornisce una misura della precisione della rappresentazione in virgola mobile, indicando così il limite inferiore della differenza tra due numeri distinti rappresentabili.
Quando viene chiamata senza argomenti, **eps** restituisce il valore di epsilon per la doppia precisione, che è 2.2204×10^{-16}. Questo valore è la più piccola quantità che aggiunta a 1.0 dà un risultato diverso da 1.0, quando lavoriamo con numeri a doppia precisione.

$$1 + x > 1$$

$$\varepsilon = min\{x \in \mathbb{F} \mid 1 + x > 1\}$$

Quando **eps** viene chiamata con un numero come argomento, restituisce il valore di epsilon relativo a quel particolare numero.

Sapere quanto sia piccola la variazione minima tra due numeri consecutivi aiuta a stabilire criteri di arresto per gli algoritmi iterativi e a scegliere soglie appropriate per confronti numerici, evitando così decisioni errate causate dalla precisione finita dei calcoli in virgola mobile.

2 Precisione degli algoritmi

2.1 Errore relativo e assoluto

L'errore assoluto di una stima è la differenza tra il valore esatto (o valore di riferimento) e il valore approssimato calcolato.

$$E_a = |valore\ esatto\ -\ valore\ approssimato|$$

L'errore assoluto dà un'idea della grandezza dell'errore in termini dell'unità di misura in uso. Tuttavia, questo tipo di errore non fornisce indicazioni sulla "grandezza" dell'errore in relazione al valore esatto, il che può essere limitante, soprattutto quando si lavora con valori molto grandi o molto piccoli.

L'errore relativo cerca di superare questo limite fornendo una misura dell'errore in rapporto al valore esatto. Questo rende l'errore relativo una misura percentuale dell'errore, offrendo una prospettiva sulla significatività dell'errore in relazione all'ordine di grandezza del valore esatto.

$$E_{rel} = \frac{E_a}{|valore\ esatto|}$$

dove x non deve essere zero. Se x è zero, l'errore relativo non è definito, poiché qualsiasi errore in un valore atteso di zero è significativo.

2.2 Numero di condizionamento

Un problema è considerato ben condizionato quando piccole variazioni nei dati di input causano solo piccole variazioni nei risultati; al contrario, è mal condizionato se piccole variazioni nei dati di input possono causare grandi variazioni nei risultati. I problemi mal condizionati sono particolarmente problematici in presenza di errori di arrotondamento e di approssimazione, che sono inevitabili nel calcolo numerico. Questi errori possono essere significativamente amplificati in un'analisi, portando a risultati poco affidabili o addirittura errati.

Un modello $f(x)$ si dice ben condizionato se vale una relazione del tipo

$$\frac{\|f(x+\delta x) - f(x)\|}{\|f(x)\|} \leq k \frac{\|\delta x\|}{\|x\|}$$

con k piccolo, $f(x) \neq 0, x \neq 0$. k viene definito numero di condizionamento.

- $f(x)$: È la funzione di interesse, $f(x) \neq 0$.
- x: È il punto (o vettore di input) su cui è valutata la funzione.
- δx: Rappresenta una piccola variazione applicata a x. Questa variazione può essere introdotta, ad esempio, da errori di misurazione o arrotondamento.
- $\|f(x+\delta x) - f(x)\|$: Indica la norma della differenza tra il valore della funzione in $x+\delta x$ e il valore della funzione in x, misurando quindi l'impatto della variazione δx sull'output della funzione.
- $\|f(x)\|$: È la norma del valore della funzione in x, prima di qualsiasi perturbazione.
- k: È una costante non negativa che rappresenta il limite superiore della sensibilità della funzione alle variazioni nell'input. In altre parole, k viene definito numero di condizionamento e quantifica quanto sia "condizionata" la funzione rispetto alle variazioni dell'input. Un valore basso di k indica un problema ben condizionato, mentre un valore alto indica un problema mal condizionato.
- $\frac{\|\delta x\|}{\|x\|}$: Rappresenta la misura relativa della perturbazione applicata all'input x, ovvero quanto è grande la variazione δx rispetto al valore originale x.

L'inequazione stabilisce un limite superiore all'effetto relativo di una variazione dell'input sulla funzione. Il termine a sinistra dell'inequazione misura l'impatto relativo della variazione δx sull'output della funzione $f(x)$, normalizzato rispetto al valore di $f(x)$ stesso. Il termine a destra stabilisce un limite proporzionale alla grandezza relativa della perturbazione δx rispetto al valore originale x, moltiplicato per una costante k.

È possibile migliorare il condizionamento di un problema mediante tecniche di precondizionamento, selezione di algoritmi numerici più stabili, o riformulazione del problema stesso. Queste strategie possono aiutare a ottenere soluzioni più precise e affidabili, anche in presenza di problemi inizialmente mal condizionati.

3 Metodi per la ricerca di radici di equazioni non lineari

3.1 Metodo di bisezione

Teorema di esistenza degli zeri per funzioni continue

Ipotesi di base:

- La funzione f deve essere continua sull'intervallo $[a, b]$.
- Deve essere verificato che $f(a)f(b) < 0$ (teorema degli zeri), indicando f cambia segno sull'intervallo e, per il teorema degli zeri, implica l'esistenza di almeno una radice in $[a, b]$.

Il metodo di bisezione cerca di restringere sistematicamente l'intervallo in cui si sa che deve esistere una radice, dividendo l'intervallo a metà (da qui il termine "bisezione") e selezionando il sotto-intervallo in cui la funzione cambia segno. Questo processo viene ripetuto iterativamente fino a quando l'intervallo non è sufficientemente piccolo, garantendo che il punto medio dell'intervallo finale sia una approssimazione della radice con la precisione desiderata. Algoritmo:

1. Inizializzazione: Parti con un intervallo iniziale $[a, b]$ dove $f(a)f(b) < 0$.
2. Calcolo del punto medio: Calcola il punto medio $m = \frac{a+b}{2}$ dell'intervallo.
3. Verifica della condizione di arresto: Se $f(m) = 0$ o l'ampiezza dell'intervallo $[a, b]$ è inferiore a una tolleranza prestabilita ε, allora m è la radice cercata o una sua buona approssimazione.
4. Selezione del nuovo intervallo: Se $f(m)f(a) < 0$, la radice si trova nel sotto-intervallo $[a, m]$, altrimenti si trova in $[m, b]$. Aggiorna a o b di conseguenza.
5. Iterazione: Ripeti i passaggi da 2 a 4 fino a che non si raggiunge la condizione di arresto.

$$
\begin{aligned}
&I_k = [a^{(k)}, b^{(k)}] \text{ t.c. } f(a^{(k)})f(b^{(k)}) < 0 \\
&m^{(k)} = \frac{a^{(k)} + b^{(k)}}{2} \\
&\text{se } f(a^{(k)})f(m^{(k)}) = 0 \rightarrow m^{(k)} \text{ è radice} \\
&\text{se } f(a^{(k)})f(m^{(k)}) < 0 \rightarrow I_{k+1} = [a^{(k+1)}, b^{(k+1)}] = [a^{(k)}, m^{(k)}] \\
&\text{se } f(a^{(k)})f(m^{(k)}) > 0 \rightarrow f(m^{(k)})f(b^{(k)}) < 0 \rightarrow I_{k+1} = [a^{(k+1)}, b^{(k+1)}] = [m^{(k)}, b^{(k)}]
\end{aligned}
$$

Il metodo di bisezione è detto globalmente convergente perché, dato un intervallo iniziale $[a, b]$ che soddisfa la condizione $f(m)f(a) < 0$, il metodo convergerà sempre verso una radice, indipendentemente dalla scelta iniziale di a e b, purché la funzione f sia continua in $[a, b]$. La convergenza globale è una proprietà desiderabile nei metodi numerici perché assicura che, seguendo il metodo, si arriverà a una soluzione senza la necessità di fare ipotesi iniziali vicine alla radice vera. Supponiamo di essere al passo m il p.to medio è $x^{(m)}$:

$$e^{(m)} = |x^{(m)} - \alpha| \leq |I_m| = |b^{(m)} - a^{(m)}| = \frac{|I_0|}{s^m}$$

quindi $lim_{m \to +\infty} e^{(m)} = 0$

- $e^{(m)}$ rappresenta l'errore assoluto alla m-esima iterazione, che è la differenza in valore assoluto tra l'approssimazione corrente della radice $x^{(m)}$ e la vera radice α.

- $|I_m|$ indica la lunghezza dell'intervallo alla m-esima iterazione, che è la differenza tra i limiti superiore $b^{(m)}$ e inferiore $a^{(m)}$ di tale intervallo.
- $|I_0|$ è la lunghezza dell'intervallo iniziale.
- s è il fattore di riduzione dell'intervallo ad ogni iterazione, che nel caso del metodo di bisezione è sempre 2 perché l'intervallo viene diviso in due parti uguali ad ogni passo.
- m è il numero di iterazioni.
- $e^{(m)} = |x^{(m)} - \alpha|$ ci dice quanto sia vicino il nostro punto medio $x^{(m)}$ alla vera radice α, cioè l'accuratezza della nostra approssimazione attuale.
- $|I_m| = |b^{(m)} - a^{(m)}| = \frac{|I_0|}{s^m}$ ci mostra come la lunghezza dell'intervallo si riduca esponenzialmente con il numero di iterazioni m, dimezzandosi ad ogni passo. Questo calo rapido della dimensione dell'intervallo è la chiave della convergenza del metodo di bisezione.

La lunghezza dell'intervallo $|I_m|$ funge da limitazione superiore per l'errore $e^{(m)}$, dal momento che la vera radice α deve trovarsi all'interno dell'intervallo $[a, b]$. Quindi, se l'intervallo è ridotto a una lunghezza molto piccola, anche l'errore sarà necessariamente piccolo, implicando che $x^{(m)}$ è una buona approssimazione di α.

Per determinare dopo quanti passi l'algoritmo può essere arrestato occorre una tolleranza ε

$$e^{(m)} \leq \frac{|I_0|}{2^m} < \varepsilon \text{ quindi } \frac{b-a}{2^m} < \varepsilon$$

Che si basa sul fatto che l'errore è limitato superiormente dalla lunghezza dell'intervallo, che si dimezza ad ogni iterazione nel metodo di bisezione. Da qui, se l'intervallo iniziale è $|I_0| = b - a$, possiamo riscrivere la condizione come:

$$\frac{b-a}{2^m} < \varepsilon$$

Per determinare il numero minimo di iterazioni m necessarie per raggiungere questa tolleranza, risolviamo la disequazione rispetto a m:

$$m > \frac{log(b-a) - log(\varepsilon)}{log(2)}$$

Questa formula ci dice quante iterazioni sono necessarie, almeno in teoria, per assicurarci che l'errore di approssimazione della radice sia inferiore alla tolleranza ε specificata.

Il teorema di esistenza degli zeri ci assicura l'esistenza di almeno una radice. Il teorema fondamentale dell'algebra assicura che un polinomio p di grado n ammette n radici.

$$\alpha_i \ \ i = 1 \dots n \text{ reali o complesse}$$

Sia $f \in e^m(a, b)$ ossia derivabile m volte nell'intervallo (a, b), $m \in \mathbb{N}^+$

- Radice semplice: Una radice α si dice radice semplice se $f(\alpha) = 0$ e la prima derivata $f'(\alpha) \neq 0$.
- Radice di ordine m: Una radice α si dice radice di ordine m se tutte le derivate della funzione fino all'ordine $m-1$ sono nulle in α, $f^{m-1}(\alpha) = f^{m-2}(\alpha) = \dots = f'(\alpha) = f(\alpha) = 0$, ma la m-esima derivata $f^{(m)}(\alpha) \neq 0$. In tal caso $f(x) = (x - \alpha)^m \cdot h(x)$, $h(\alpha) \neq 0$.

Il problema di ricerca di radici di equazioni non lineari consiste nel cercare $\alpha \in \mathbb{R}$ t.c. $f(\alpha) = 0$.

3.2 Condizionamento del problema di calcolo di una radice

Stiamo considerando un problema in cui si cerca un valore x tale che $f(x) = \varphi(x) - d = 0$ ovvero stiamo cercando i valori di x per i quali $\varphi(x) = d$. Qui, d funge da "obiettivo" o "target" per il valore di $\varphi(x)$, e il problema diventa quello di trovare le radici dell'equazione, ovvero i valori di x per i quali l'equazione è soddisfatta. Consideriamo anche una piccola variazione δd sul valore target d, possiamo esprimere $d + \delta d$ come lo sviluppo di Taylor di $\varphi(x + \delta x)$ attorno al punto x.

$$d + \delta d = \varphi(x + \delta x) = \varphi(x) + \varphi'(x)\delta x + \frac{\varphi''(x)}{2}(\delta x)^2 + \dots + \frac{\varphi^{(k)}(x)}{k!}(\delta x)^k + \dots$$

Lo sviluppo di Taylor mostra come la variazione δd in d possa essere collegata a δx, la variazione nella radice x che stiamo cercando. Questo sviluppo di Taylor ci permette di vedere in dettaglio come cambia $\varphi(x)$ in risposta a piccole variazioni in x.

Supponiamo che x sia radice di ordine m, quindi $f'(x) = \varphi'(x), \ldots, f^{(m-1)}(x) = \varphi^{(m-1)}(x)$. Quando si suppone che x sia una radice di ordine m per φ, significa che le prime $m-1$ derivate di φ in x sono nulle, e la derivata m-esima è la prima derivata non nulla, $\varphi^{(k)}(x) = 0 \quad k = 1 \ldots m-1$. Per cui:

$$d + \delta d = \varphi(x) + \frac{\varphi^{(m)}(x)}{m!}(\delta x)^m$$

Sapendo che $\varphi(x) = d$, possiamo semplificare ulteriormente:

$$\delta d = \frac{\varphi^{(m)}(x)}{m!}(\delta x)^m$$

La presenza di una radice di ordine m indica che la funzione ha un punto di "piattezza" estrema vicino alla radice, poiché non solo il valore della funzione ma anche le sue derivate fino all'ordine $m-1$ sono nulle. Questo significa che, per piccole variazioni δx attorno alla radice, i cambiamenti in $\varphi(x)$ sono inizialmente molto piccoli, crescendo in modo significativo solo quando i termini di ordine m diventano rilevanti. Questa piattezza si traduce in un "ritardo" nella risposta di δd a δx, nel senso che per variazioni piccole o moderate di δx, l'impatto su δd (il cambiamento nel valore che $\varphi(x)$ mira a raggiungere) potrebbe essere minimo. Questo ritardo è una manifestazione della minore sensibilità diretta della funzione alle perturbazioni vicino a radici di ordine superiore.

$$cond = sup\frac{\|\delta x\|}{\|\delta s\|}$$
$$\delta d = \frac{f^{(m)}(x)}{m!} \cdot (\delta x)^m$$

Isoliamo $(\delta x)^m$:

$$\delta x = \Big(\frac{m! \cdot \delta d}{f^{(m)}(x)}\Big)^{\frac{1}{m}}$$
$$cond = sup\Big|\frac{m! \cdot \delta d}{f^{(m)}(x)}\Big|^{\frac{1}{m}} \cdot \frac{1}{|\delta d|}$$

Per $m = 1$ il problema è mal condizionato se $f^m(x) \simeq 0$. Per $m > 1$ il problema potrebbe comunque essere mal condizionato.

Convergenza metodo di bisezione

Possiamo dire che il metodo di bisezione è globalmente convergente, ma rispetto ad altri metodi, è lenta a convergere. Il metodo di bisezione non è a convergenza monotona, ovvero non garantisce che il valore della funzione calcolato nell'intervallo diminuisca monotonicamente ad ogni iterazione. In effetti, il valore della funzione può aumentare in alcune iterazioni, anche se l'intervallo di ricerca viene sempre ridotto della metà. Questo perché il metodo di bisezione si basa sull'idea di partizionare l'intervallo di ricerca in due parti uguali e di verificare in quale delle due parti si trova la radice cercata. In alcune situazioni, può accadere che la radice si trovi in una delle due parti minori, dove il valore della funzione è maggiore rispetto all'altra parte. In questo caso, il valore della funzione potrebbe aumentare invece di diminuire durante l'iterazione. Può succedere che l'errore in una certa iterazione $k+1$ sia maggiore dell'errore all'iterazione precedente k.

$$e^{(k+1)} > e^{(k)}$$

Nel metodo di bisezione, ad ogni iterazione k, l'intervallo $[a^{(k)}, b^{(k)}]$ che contiene la radice viene dimezzato. Quindi, l'errore $|x^{(k)} - \alpha|$, che possiamo definire come la distanza tra l'approssimazione corrente della radice $x^{(f)}$(tipicamente il punto medio dell'intervallo $[a^{(k)}, b^{(k)}]$) e la radice vera α, è ridotto di metà ad ogni iterazione. Questo significa che l'errore massimo possibile dopo k iterazioni è dato da:

$$|x^{(k)} - \alpha| \leq \frac{b-a}{2^k}$$

Questa proprietà mostra una convergenza garantita, ma con un tasso che è lineare in termini della riduzione dell'intervallo, non necessariamente della distanza $|x^{(k)} - \alpha|$ stessa in relazione a una potenza p di essa stessa come nella definizione di convergenza di ordine p.

Si dice che la successione $\left\{x^{(k)}\right\}_{k=1\ldots}$ di approssimanti della radice ottenuta con un metodo numerico converge con ordine p se:

$$\exists c > 0 \ : \frac{|x^{(k+1)} - \alpha|}{|x^{(k)} - \alpha|^p} \leq c \ \ \forall k \geq k_0, \ k_0 \in \mathbb{N}$$

Quando parliamo di convergenza di ordine $p = 1$, $|x^{(k+1)-\alpha}| \leq c|x^{(k)} - \alpha|$, ci riferiamo a una convergenza lineare, dove l'errore si riduce proporzionalmente a sé stesso ad ogni iterazione, condizionato da $c < 1$. Il metodo di bisezione non ha ordine nemmeno pari ad 1. Questo significa che, sebbene l'algoritmo sia convergente, la velocità di convergenza non è particolarmente rapida. In particolare, la velocità di convergenza del metodo di bisezione dipende dalla scelta iniziale dell'intervallo e dalla posizione della radice, ma in generale l'errore si riduce di un fattore 1/2 ad ogni iterazione.

3.3 Algoritmi con ordine di convergenza superiore ma solo localmente convergenti

L'equazione

$$0 = f(\alpha) = f(x_0) + f'(c)(\alpha - x_0)$$

si basa sul teorema del valore medio di Lagrange, che afferma che esiste almeno un punto c nell'intervallo aperto tra α (la radice vera) e x_0 (l'approssimazione corrente della radice) tale che

$$f'(c) = \frac{f(\alpha) - f(x_0)}{\alpha - x_0}$$

se f è continua su $[x_0, \alpha]$ e differenziabile su (x_0, α). Poiché $f(\alpha) = 0$, possiamo riscrivere la formula come:

$$0 = f(\alpha) = f(x_0) + f'(c)(\alpha - x_0)$$

Non conoscendo c per evitare il calcolo diretto di $f'(c)$, utilizziamo q_0 come approssimazione di $f'(c)$, derivando quindi una formula iterativa per l'approssimazione successiva della radice:

$$f(x_0) + q_0(x_1 - x_0) = 0$$

$$x_1 = x_0 - \frac{f(x_0)}{q_0}$$

Reiterando il procedimento, otteniamo una formula iterativa generale per trovare la radice:

$$x_{k+1} = x_k - \frac{f(x_k)}{q_k}$$

dove q_k è una stima della derivata $f'(c)$ vicino a x_k, che può essere aggiornata o mantenuta costante in ogni iterazione, a seconda del metodo specifico utilizzato.

3.3.1 Metodo delle corde

Il metodo si basa sull'uso di una "corda", ossia una linea retta che congiunge i punti $(a, f(a))$ e $(b, f(b))$ sul grafico della funzione. La pendenza di questa corda è data da:

$$q_k = \text{costante} = \frac{f(b) - f(a)}{b - a}$$

Questa pendenza q_k è usata come approssimazione della derivata di f nell'intervallo $[a, b]$, e si assume costante per tutte le iterazioni.

Dato un punto iniziale x_0, la successione $\{x_k\}$ di approssimanti della radice è generata la formula:

$$x_{k+1} = x_k - f(x_k)\frac{b - a}{f(b) - f(a)}$$

La retta che passa per i punti $(a, f(a))$ e $(b, f(b))$ ha equazione:

$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a)$$

Questa equazione rappresenta una retta che approssima il comportamento della funzione $f(x)$ tra i punti a e b, dove si presume che $f(a)$ e $f(b)$ abbiano segni opposti, indicando la presenza di una radice nell'intervallo (a, b).

Per trovare una nuova approssimazione della radice, costruisci una retta parallela a quella iniziale, ma passante per il punto $(x_k, f(x_k))$, dove x_k è l'attuale approssimazione della radice:

$$\tilde{y} = \frac{f(b) - f(a)}{b - a}(x - x_k) + f(x_k)$$

Imponendo $\tilde{y} = 0$ per trovare il punto in cui questa nuova retta interseca l'asse delle ascisse, ottieniamo l'equazione:

$$0 = \frac{f(b) - f(a)}{b - a}(x_{k+1} - x_k) + f(x_k)$$

Risolvendo per x_{k+1}, l'iterazione successiva nella ricerca della radice, si ottiene:

$$x_{k+1} = x_k - \frac{b - a}{f(b) - f(a)} f(x_k)$$

L'ordine di convergenza del metodo delle corde è 1, il che significa che converge linearmente. Questo è dovuto al fatto che la "corda", ovvero la pendenza costante usata per approssimare la derivata della funzione, rimane invariata durante il processo iterativo. La pendenza è data da $\frac{f(b)-f(a)}{b-a}$, dove a e b sono i punti iniziali che definiscono l'intervallo di ricerca. A causa di questa pendenza costante, il metodo non sfrutta appieno le informazioni locali sulla curvatura della funzione, a differenza del metodo di Newton, che aggiorna la pendenza (derivata) ad ogni iterazione.

A differenza del metodo di bisezione, che è globalmente convergente sotto le sue ipotesi operative (continuità di f e cambio di segno di f agli estremi dell'intervallo), il metodo delle corde non è globalmente convergente. Ciò significa che non c'è una garanzia universale che il metodo convergerà alla radice da qualsiasi punto iniziale. La convergenza del metodo delle corde dipende strettamente dalla scelta dei punti iniziali a e b, e dalla natura della funzione f. Il metodo può fallire o convergere molto lentamente se la "corda" non rappresenta adeguatamente la pendenza di f vicino alla radice, o se f ha un comportamento non lineare pronunciato nell'intervallo considerato. Inoltre, se $f(a)$ e $f(b)$ non racchiudono una singola radice o se la funzione f non è monotona e/o non presenta derivata continua nell'intervallo $[a, b]$, il metodo potrebbe non convergere al valore corretto.

3.3.2 Metodo di Newton (o metodo delle tangenti)

Il metodo di Newton si basa su un approccio iterativo che utilizza la derivata prima $f'(x_k)$ della funzione per trovare una successione di valori x_k che converge alla radice α. A ogni iterazione, il metodo calcola il punto in cui la tangente alla curva di $f(x)$ in x_k interseca l'asse x, e usa questo punto come la prossima approssimazione x_{k+1} della radice. L'equazione iterativa è:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Il metodo di Newton è noto per la sua convergenza quadratica ($p = 2$), il che significa che, sotto certe condizioni, l'errore si riduce quadraticamente ad ogni iterazione. Questo rende il metodo di Newton estremamente efficiente vicino alla radice, specialmente quando α è una radice semplice di $f(x)$ e $f'(x)$ non è zero in α.

Affinché il metodo di Newton converga con ordine 2, sono necessarie alcune ipotesi:

1. $f(x)$ deve essere sufficientemente differenziabile nella vicinanza della radice α, almeno due volte differenziabile.
2. La radice α deve essere semplice, cioè $f'(\alpha) \neq 0$.
3. L'approssimazione iniziale x_0 deve essere sufficientemente vicina alla radice vera α.

Il metodo di Newton non è globalmente convergente. Questo significa che non ci sono garanzie che partendo da qualsiasi punto iniziale x_0, il metodo convergerà alla radice desiderata α. La scelta di x_0 ha un impatto significativo sulla convergenza: una scelta inadeguata può portare a divergenza o a convergenza verso una radice diversa da α, specialmente in presenza di funzioni con multiple radici o con derivata prima vicina a zero.

3.4 Criteri d'arresto

I criteri d'arresto sono essenziali nel definire quando un algoritmo iterativo per la ricerca delle radici di una funzione debba considerarsi concluso, basandosi su una tolleranza ε che rappresenta l'accettabilità dell'errore nelle approssimazioni calcolate.

3.4.1 Criterio del residuo

Questo criterio valuta direttamente quanto sia vicina a zero la funzione $f(x)$ per l'approssimazione corrente x_k della radice. L'algoritmo si arresta quando il valore assoluto del residuo (cioè, il valore della funzione) diventa minore di una tolleranza prefissata ε:

$$|f(x_k)| < \varepsilon$$

Questo criterio è utile perché offre una misura diretta di quanto siamo vicini a soddisfare la condizione $f(x) = 0$, indicando che x_k è un'approssimazione sufficientemente buona della radice.

3.4.2 Criterio dell'incremento

Questo criterio si basa sulla differenza tra due approssimazioni successive della radice, $x^{(k+1)}$ e $x^{(k)}$, e si arresta quando questa differenza diventa minore della tolleranza ε:

$$|x^{(k+1)} - x^{(k)}| < \varepsilon$$

Questo criterio è indicativo della "stabilizzazione" dell'algoritmo: se le successive approssimazioni della radice cambiano di una quantità inferiore a ε, si assume che l'algoritmo abbia raggiunto un livello di precisione accettabile.

Criticità del criterio dell'incremento: se la convergenza del metodo verso la radice è molto lenta allora la differenza tra due approssimanti successivi sarà "piccola" anche se l'errore complesso è ancora "grande".

3.4.3 Numero massimo di iterazioni

Questo criterio pone un limite al numero di iterazioni che l'algoritmo può eseguire, interrompendo l'esecuzione una volta raggiunto questo limite. Questo criterio è importante per prevenire cicli infiniti o esecuzioni prolungate senza convergenza, garantendo che l'algoritmo termini in un tempo ragionevole:

$$k = k_{max}$$

4 Problema dell'interpolazione

Il problema dell'interpolazione coinvolge la ricerca di un polinomio $p(x)$ di grado m che passa per $n+1$ punti dati (x_i, y_i), dove $i = 0, 1, \ldots, n$, tale che $p(x_i) = y_i$. I punti tutti distinti sono detti nodi di interpolazione, mentre la funzione p è detto interpolatore. Questo tipo di problema si presenta ad esempio quando si desidera trovare una funzione che approssimi una serie di dati sperimentali o quando si necessita di un modello matematico per descrivere il comportamento di un sistema fisico a partire da osservazioni discrete.

$$p(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$$

$$\begin{cases} p(x_0) = a_m x_0^m + a_{m-1} x_0^{m-1} + \cdots + a_1 x_0 + a_0 = y_0 \\ p(x_1) = a_m x_1^m + a_{m-1} x_1^{m-1} + \cdots + a_1 x_1 + a_0 = y_1 \\ \vdots \\ p(x_n) = a_m x_n^m + a_{m-1} x_n^{m-1} + \cdots + a_1 x_n + a_0 = y_n \end{cases}$$

$n+1$ equazioni, $m+1$ incognite $a_0 \ldots a_m$.
La formulazione del problema porta alla creazione di un sistema lineare $Aa = y$, dove A è una matrice di Vandermonde $A_{m,n}$, a è il vettore delle incognite a_i, e y è il vettore dei valori y_i osservati nei punti x_i.

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^m \\ 1 & x_1 & x_1^2 & \cdots & x_1^m \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Tipi di sistemi:

- Sovradeterminato ($n > m$): Ci sono più equazioni che incognite, il che significa che il sistema può non avere soluzioni esatte.
- Sottodeterminato ($n < m$): Ci sono meno equazioni che incognite, rendendo il sistema indeterminato con infinite soluzioni possibili.
- Determinato ($n = m$) In questo caso, il sistema ha esattamente tante equazioni quante sono le incognite. Se il determinante della matrice A (matrice di Vandermonde nel nostro caso) è diverso da zero, esiste un'unica soluzione al sistema.

Matrice di Vandermonde

In matlab VANDER costruisce la matrice Vandermonde associata ai nodi specialmente per gradi elevati del polinomio e/o per un ampio intervallo dei punti x_i. Questo significa che piccoli errori nei dati y_i o nell'aritmetica del calcolatore possono portare a grandi errori nella soluzione trovata. Per ovviare a questo problema, un approccio comune è utilizzare la forma di interpolazione di Lagrange.

4.1 Interpolazione polinomiale di Lagrange

L'interpolazione polinomiale di Lagrange è un approccio alternativo che non richiede la soluzione di un sistema lineare. Invece, il polinomio interpolante è costruito direttamente come una combinazione lineare di polinomi di base di Lagrange, ciascuno dei quali è costruito per essere uguale a 1 in uno dei punti di interpolazione e 0 in tutti gli altri.

Data una serie di $n+1$ punti (x_i, y_i) con $i = 0, 1, \cdots, n$ il polinomio interpolante $p(x)$ di grado al massimo n è dato da:

$$p(x) = \sum_{i=0}^{n} y_i L_i(x)$$

dove $L_i(x)$ è l'i-esimo polinomio di Lagrange definito come:

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Vorremmo che $p(x_i) = y_i$

$$y_0 = p(x_0) = \sum_{i=0}^{n} y_i L_i(x_0) = y_0 L_0(x_0) + y_1 L_1(x_0) + \cdots + y_{n-1} L_{n-1}(x_0) + y_n L_n(x_0)$$

$$y_1 = p(x_1) = \sum_{i=0}^{n} y_i L_i(x_1) = y_0 L_0(x_1) + y_1 L_1(x_1) + \cdots + y_{n-1} L_{n-1}(x_1) + y_n L_n(x_1)$$

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \Big(\frac{x - x_0}{x_i - x_0}\Big)\Big(\frac{x - x_1}{x_i - x_1}\Big)\Big(\frac{x - x_2}{x_i - x_2}\Big) \dots \Big(\frac{x - x_{i-1}}{x_i - x_{i-1}}\Big)\Big(\frac{x - x_{i+1}}{x_i - x_{i+1}}\Big) \dots \Big(\frac{x - x_{n-1}}{x_i - x_{n-1}}\Big)\Big(\frac{x - x_n}{x_i - x_n}\Big)$$

4.2 Metodo di Newton (metodo alle differenze divise)

Il metodo delle differenze divise di Newton rappresenta un'alternativa efficace al polinomio interpolatore di Lagrange, specialmente quando si aggiungono nuovi punti ai dati esistenti. A differenza del polinomio di Lagrange, il metodo di Newton permette di aggiungere un nuovo punto senza dover ricalcolare da zero i coefficienti del polinomio interpolante. Questo aspetto lo rende particolarmente vantaggioso in termini di efficienza computazionale quando si lavora con set di dati in evoluzione o si eseguono aggiunte incrementali.

Quando si inizia con un singolo punto (x_0, y_0) e si costruisce il polinomio costante $p_0(x) = y_0$, questo polinomio rappresenta l'interpolazione più semplice possibile, che è semplicemente una linea orizzontale al

livello di y_0. Il polinomio $p_0(x)$ "interpolante" è costante perché non abbiamo ancora introdotto variazione o pendenza derivante da ulteriori punti dati. Quando si aggiunge un altro punto (x_1, y_1), si vuole modificare il polinomio esistente per passare anche attraverso questo nuovo punto. Tuttavia, $p_0(x)$ non ha la "capacità" di adattarsi a nuovi punti dato che è un polinomio di grado zero (una costante). Per permettere al nostro polinomio interpolante di passare attraverso entrambi i punti, abbiamo bisogno di "estendere" il nostro polinomio aggiungendo un termine che si annulla in x_0 (per preservare il valore di y_0 in quel punto) e che contribuisce alla differenza in y necessaria per raggiungere y_1 da y_0.

Per questo, costruiamo $p_1(x)$ come $p_0(x) + q_1(x)$, dove $q_1(x)$ è scelto in modo tale da soddisfare le seguenti condizioni:

1. $q_1(x_0) = 0$, così che $p_1(x_0) = p_0(x_0) = y_0$, preservando il valore in x_0.
2. $q_1(x_1)$ contribuisce con esattamente quanto necessario per che $p_1(x_1) = y_1$.

$$p_1(x) = p_0(x) + q_1(x)$$
$$q_1(x) = p_1(x) - p_0(x)$$

La forma tipica per $q_1(x)$ in un polinomio di Newton è $a_1(x - x_0)$, dove a_1 è il coefficiente che determina la pendenza necessaria per passare da y_0 a y_1. Questo coefficiente a_1 può essere trovato utilizzando le differenze divise basate sui punti x_0, y_0 e x_1, y_1.
Così facendo, $p_1(x)$ diventa un polinomio di grado 1, o una linea retta, che passa per entrambi i punti dati.

$$y_1 = p_1(x_1) = p_0(x_1) + a(x_1 - x_0)$$

Da cui:

$$a = \frac{y_1 - p_0(x_1)}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

Questo calcolo di aa rappresenta la prima differenza divisa, che è fondamentalmente la pendenza della retta che collega i punti (x_0, y_0) e (x_1, y_1). Incorporando il valore di a trovato, il polinomio interpolante $p_1(x)$ diventa:

$$p_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

Il processo può essere esteso aggiungendo ulteriori punti, incrementando di grado il polinomio a ogni passo e utilizzando le differenze divise per calcolare i nuovi coefficienti. Questo metodo consente di costruire polinomi interpolanti di grado progressivamente più alto in modo efficiente, adattandosi perfettamente ai punti forniti uno per uno.

$$p_2(x) = p_1(x) + q_2(x)$$

Vogliamo che $p_2(x)$ interpoli (x_0, y_0), (x_1, y_1) e (x_2, y_2) quindi:

$$p_2(x_0) = y_0 \to p_1(x_0) + q_2(x_0) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_0 - x_0) + q_2(x_0) = y_0$$

$$p_2(x_0) = y_0 \to p_1(x_1) + q_2(x_1) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + q_2(x_1) = y_1$$

$$p_2(x_2) = y_2$$

quindi cerco $q_2(x) = a(x - x_0)(x - x_1)$. Impongo il passaggio per (x_2, y_2) per determinare il valore di a:

$$y_2 = p_2(x_2) = p_1(x_2) + q_2(x_2) = p_1(x_2) + a(x_2 - x_0)(x_2 - x_1)$$

$$a = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}$$

$$p_2(x_2) = p_1(x) + \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}(x - x_0)(x - x_1) =$$

$$= p_0(x) + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}(x - x_0)(x - x_1) =$$

$$= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}(x - x_0)(x - x_1) =$$

Possiamo costruirci una tabella delle differenze divise. Ogni coefficiente nel polinomio di Newton corrisponde a una differenza divisa di un certo ordine. Il calcolo di queste differenze divise è un processo gerarchico, in cui ogni differenza divisa di ordine superiore si basa su quelle di ordine inferiore.
Per $n+1$ punti dati $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, il polinomio interpolante di Newton può essere scritto come:

$$p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, \ldots, x_n](x - x_0)(x - x_1)(x - x_2)\ldots(x - x_{n-1})$$

Le differenze divise si calcolano così:

- $f[x_i] = y_i$ (per $i = 0, 1, \ldots, n$)
- $f[x_i, x_{i+1}] = \dfrac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$ (per $i = 0, 1, \ldots, n-1$)
- $f[x_i, x_{i+1}, \ldots, x_{i+j}] = \dfrac{f[x_{i+1}, \ldots, x_{i+j}] - f[x_i, \ldots, x_{i+j-1}]}{x_{i+j} - x_i}$ (per $j = 2, 3, \ldots, n-i$)

Differenza divisa di ordine 0:

- $y_0 = f[x_0]$
- $y_1 = f[x_1]$

Differenza divisa di ordine 1:

- $f[x_0, x_1] = \dfrac{f[x_1] - f[x_0]}{x_1 - x_0} = \dfrac{y_1 - y_0}{x_1 - x_0}$
- $f[x_i, x_{i+1}] = \dfrac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$ (per $i = 0, \ldots, n-1$)

Differenza divisa di ordine 2:

- $f[x_i, x_{i+1}, x_{i+2}] = \dfrac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$ (per $i = 0, \ldots, n-2$)

Differenza divisa di ordine k:

- $f[x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k}] = \dfrac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$ (per $i = 0, \ldots, n-k$)

Nel metodo delle differenze divise di Newton per l'interpolazione polinomiale, l'ordine in cui scegli i punti (x_i, y_i) non influisce sul polinomio finale risultante o sulla sua capacità di interpolare i dati. In altre parole, indipendentemente dall'ordine dei punti, il polinomio interpolante di Newton passerà attraverso tutti i punti dati.

4.3 Condizionamento del problema di interpolazione polinomiale

Il problema dell'interpolazione polinomiale implica trovare un polinomio $p(x)$ di grado minimo che passa attraverso un insieme di punti (x_i, y_i) per $i = 0 \ldots n$, esprimibile come:

$$p(x) = \sum_{i=0}^{n} y_i L_i(x)$$

Questo metodo si basa sull'uso dei polinomi di Lagrange $L_i(x)$ per costruire il polinomio interpolante. Consideriamo i vettori:

$$y = (y_0, y_1, \ldots, y_n)$$
$$\tilde{y} = (\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_n)$$
$$|y - \tilde{y}| = (|y_0 - \tilde{y}_0|, |y_1 - \tilde{y}_1|, \ldots, |y_n - \tilde{y}_n|)$$

Nel contesto dell'interpolazione polinomiale, il condizionamento del problema riflette la sensibilità della soluzione $p(x)$ rispetto a variazioni nei dati di input y_i. Se si considerano piccole perturbazioni nei punti dati, ovvero $(x_i, \tilde{y}_i)$ per $i = 0 \ldots n$, il nuovo polinomio interpolante diventa:

$$\tilde{p}(x) = \sum_{i=0}^{n} \tilde{y}_i L_i(x)$$

L'effetto di tali perturbazioni può essere quantificato tramite la differenza:

$$|p(x) - \tilde{p}(x)| = |\sum_{i=0}^{n}(y_i - \tilde{y}_i)L_i(x)| \leq \sum_{i=0}^{n}|y_i - \tilde{y}_i||L_i(x)| \leq \|y - \tilde{y}\|_\infty \sum_{i=0}^{n}|L_i(x)|$$

dove $\|y - \tilde{y}\|_\infty$ rappresenta la norma infinito della differenza vettoriale tra i vettori y e $\tilde{y}$, ovvero l'errore commesso sul dato. Questo risultato implica che il massimo errore in $p(x)$ causato dalle perturbazioni nei dati è amplificato dalla somma dei valori assoluti dei polinomi di Lagrange, ossia la costante di Lebesgue Λ_n, che rappresenta il termine di condizionamento.

$$\|p(x) - \tilde{p}(x)\|_\infty = \max_{x\in I}|p(x) - \tilde{p}(x)| \leq \|y - \tilde{y}\|_\infty \| \sum_{i=0}^{n}|\mathbb{L}_i(x)|\|_\infty$$

Questa formula quantifica l'errore massimo di interpolazione in termini della norma infinito, evidenziando come l'errore massimo possibile nel polinomio interpolante sia controllato dalla massima perturbazione nei dati e dalla massima somma dei valori assoluti dei polinomi di Lagrange su tutto l'intervallo di interpolazione (costante di Lebesgue).

$$\|p(x)\|_\infty = \max_{x\in I}|p(x)| \geq \max_{i=0\ldots n}|p(x_i)| = \max_{i=0\ldots n}|y_i| = \|y\|_\infty$$

Questa equazione stabilisce che la norma infinito del polinomio interpolante $p(x)$ è almeno tanto grande quanto il massimo valore assoluto dei dati interpolati y_i, che si verifica ai nodi di interpolazione. Questo legame è essenziale per valutare quanto il polinomio $p(x)$ rappresenti fedelmente i valori ai nodi.

$$\frac{\|p(x) - \tilde{p}(x)\|_\infty}{\|p(x)\|_\infty} \leq \Lambda_n \frac{\|y - \tilde{y}\|_\infty}{\|y\|_\infty}$$

Questa espressione fornisce una stima relativa dell'errore nel contesto dell'interpolazione polinomiale. Mostra che il rapporto tra l'errore massimo e il valore massimo del polinomio originale è limitato dal prodotto della costante di Lebesgue e dal rapporto tra le perturbazioni massime nei dati e il valore massimo dei dati originali. Questo quantifica in termini relativi la "stabilità" del problema di interpolazione rispetto a perturbazioni nei dati.

La costante di Lebesgue Λ_n, definita come:

$$\Lambda_n = \| \sum_{i=0}^{n}|L_i(x)|\|_\infty$$

quantifica l'amplificazione massima dell'errore nei dati nell'output interpolato, essendo un indice della stabilità del problema di interpolazione. La scelta dei nodi x_i influenza direttamente il valore di Λ_n. Quando si utilizzano nodi equispaziati nell'interpolazione polinomiale, la costante di Lebesgue Λ_n mostra una crescita molto rapida, approssimativamente data dalla formula:

$$\Lambda_n \simeq \frac{2^{n+1}}{e\, n\, logn} \to +\infty \text{ per } n \to \infty$$

Questa espressione indica che la costante di Lebesgue tende all'infinito all'aumentare di n, suggerendo una crescente instabilità numerica per polinomi di grado elevato. Questo fenomeno, noto come fenomeno di Runge, evidenzia l'alta sensibilità dell'interpolazione a variazioni nei dati di input quando si usano nodi equispaziati, rendendo il processo di interpolazione numericamente instabile, specialmente ai bordi dell'intervallo di interpolazione.

Al contrario, quando si utilizzano i nodi di Chebyshev, la crescita della costante di Lebesgue è molto più contenuta ed è descritta dalla relazione:

$$\Lambda_n \simeq \frac{2}{\pi} logn \to +\infty \text{ per } n \to \infty$$

Questa formula mostra che, anche se Λ_n aumenta con il numero di nodi n, la crescita è logaritmica e moderata dal fattore $2/\pi$. L'uso dei nodi di Chebyshev migliora significativamente la stabilità dell'interpolazione rispetto ai nodi equispaziati, mitigando il rischio del fenomeno di Runge e garantendo risultati più affidabili. La determinazione dei nodi di Chebyshev, sono definiti nel modo seguente:

$$x_i^{(n)} = cos\left(\frac{2i+1}{2n+2}\pi\right), \text{ per } i = 0, 1, \ldots, n \;\; x_i \in [-1, 1]$$

La formula sfrutta il fatto che il coseno di angoli equidistanti nella circonferenza unitaria produce valori che sono ottimamente spaziati per l'interpolazione polinomiale. Il termine $\frac{2i+1}{2n+2}\pi$ rappresenta angoli che sono distribuiti simmetricamente attorno alla metà dell'intervallo del coseno, ovvero intorno a $\pi/2$, garantendo che i nodi siano simmetricamente distribuiti rispetto all'origine e si concentrino vicino agli estremi.

Teorema

Si consideri una funzione $f(x)$ che è continua e derivabile fino all'ordine $n+1$ sull'intervallo $[I_x]$, dove I_x è il minimo intervallo contenente i nodi di interpolazione e l'ascissa x.

$$x_0 \dots x_n$$

allora esiste un'ascissa $c_x \in I_x$ tale che:

$$E_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}(c_x)}{(n+1)!}\omega_{n+1}(x),$$

dove $\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i) = (x - x_0)(x - x_1)\dots(x - x_n)$ rappresenta il prodotto delle differenze tra x e i nodi di interpolazione. Inoltre, il modulo massimo dell'errore è calcolato come segue:

$$|E_n(x)| = \frac{|f^{(n+1)}(c_x)|}{(n+1)!}|\omega_{n+1}(x)| \leq \frac{\max_{x\in[I_x]}|f^{(n+1)}(x)|}{(n+1)!}|I_x|^{n+1}$$

L'analisi dell'errore di interpolazione è essenziale per comprendere come la scelta del grado di interpolazione e la distribuzione dei nodi influenzino l'accuratezza del polinomio interpolante. I nodi di interpolazione, $x_0, \dots, x_n$, sono selezionati strategicamente per minimizzare l'errore $|\omega_{n+1}(x)|$, cruciale per ridurre l'errore complessivo. L'utilizzo di nodi come i nodi di Chebyshev può ridurre questo errore in modo più efficace rispetto ai nodi equispaziati, sfruttando la loro distribuzione ottimale per mitigare le oscillazioni e gli errori specialmente nelle estremità dell'intervallo di interpolazione.

Inoltre, la derivata di ordine $n+1$ della funzione, $f^{(n+1)(c_x)}$, evidenzia la sensibilità di f alle variazioni in x e come queste influenzano l'errore di interpolazione. Il fattore $(n+1)!$ nel denominatore normalizza l'errore in funzione del numero dei nodi, dimostrando che un grado di interpolazione più elevato può risultare in un errore minore se la funzione è sufficientemente liscia. In pratica, questo significa che per funzioni con buone proprietà di lisciatura (ovvero funzioni con derivata di ordine elevato continua), aumentare il grado del polinomio interpolante può portare a una maggiore precisione dell'interpolazione senza incorrere in un aumento proporzionale dell'errore.

4.4 Interpolazione composita

Un metodo per approssimare una funzione su un intervallo ampio tramite l'uso di più polinomi di basso grado anziché un unico polinomio di alto grado. Questa tecnica viene impiegata per dividere l'intervallo di definizione della funzione in m sottointervalli, ciascuno denotato da $[x_k, x_{k+1}]$ per $k = 0, \dots, m-1$, dove x_k sono i nodi di interpolazione. In ogni sottointervallo, si costruisce un polinomio interpolatore semplice (Lagrange) che utilizza $n+1$ nodi equispaziati.

Questo approccio mira a ridurre gli effetti del fenomeno di Runge che spesso si verifica con l'uso di polinomi di grado elevato su intervalli larghi. L'interpolazione in segmenti più piccoli consente di mantenere il grado del polinomio relativamente basso, limitando le oscillazioni e migliorando l'accuratezza dell'approssimazione locale. Il massimo errore di interpolazione in questo contesto è indicato da H, che rappresenta la massima distanza tra i nodi consecutivi $x_{k+1} - x_k$. Una riduzione di H comporta tipicamente una diminuzione dell'errore di interpolazione, poiché il polinomio può adattarsi più precisamente alla funzione su intervalli più ristretti.

$$H = \max_{k=0\dots m-1}|x_{k+1} - x_k|$$

$$\|f - p_n^c\|_\infty \leq cH^{n+1}\|f^{n+1}\|_\infty$$

Riducendo la distanza massima H tra i nodi di interpolazione, l'errore di interpolazione tende a zero con un ordine di convergenza proporzionale a H^{n+1}. Questo significa che, man mano che i nodi diventano più fitti (cioè H si riduce), l'errore di interpolazione diminuisce molto rapidamente, seguendo un tasso proporzionale alla potenza $n+1$ di H. La regolarità della funzione f ricopre un ruolo fondamentale, poiché la presenza di derivate continue e limitate fino all'ordine $n+1$ è essenziale per assicurare che la stima dell'errore sia valida.

4.5 Metodo dei minimi quadrati

$$\sum_{i=0}^{n}[y_i - \hat{p}(x_i)]^2 \leq \sum_{i=0}^{n}[y_i - p(x_i)]^2 \ \ \forall p \text{ polinomio di grado 1}$$

Il problema si concentra sulla determinazione dei coefficienti m e q del polinomio di primo grado, cioè la retta $\hat{p}(x) = mx+q$, che meglio approssima un insieme di dati $\{(x_i, y_i)\}$ nel senso dei minimi quadrati. L'obiettivo è minimizzare la somma dei quadrati degli scarti verticali tra i punti dati e i valori previsti dalla retta. L'errore quadratico è rappresentato dalla formula:

$$\Phi(m, q) = \sum_{i=0}^{n}[y_i - (mx_i + q)]^2 = \sum_{i=0}^{n}[y_i^2 + m^2x_i^2 + q^2 + 2mqx_i - 2mx_iy_i - 2y_iq]$$

dove y_i sono i valori osservati e $mx_i + q$ sono i valori stimati dal modello.

Per minimizzare questo errore, deriviamo $\Phi(m, q)$ rispetto a m e q e poniamo queste derivate pari a zero, ottenendo due equazioni fondamentali. D'ora in poi, per migliorare la leggibilità, quando verrà usato il simbolo di sommatoria senza indici $\sum$, sarà sottointeso $\sum_{i=0}^{n}$. La derivata rispetto a m è:

$$\frac{\partial\Phi}{\partial m} = -2\sum(y_i - mx_i - q)x_i = 0$$

e la derivata rispetto a q è:

$$\frac{\partial\Phi}{\partial q} = -2\sum(y_i - mx_i - q) = 0$$

Queste sono le equazioni normali del metodo dei minimi quadrati per una regressione lineare.

Risolvendo questo sistema di equazioni normali, determiniamo i valori di m e q che minimizzano l'errore quadratico. I valori sono dati da:

$$m = \frac{(n+1)\sum x_iy_i - \sum x_i \sum y_i}{(n+1)\sum x_i^2 - (\sum x_i)^2}$$

$$q = \frac{\sum y_i \sum x_i^2 - \sum x_i(\sum x_iy_i)}{(n+1)\sum x_i^2 - (\sum x_i)^2}$$

dove n è il numero di punti dati.

Questo processo è essenziale per la regressione lineare e viene utilizzato frequentemente non solo in statistica ma anche in ambiti scientifici e ingegneristici per modellare relazioni lineari tra variabili. Le equazioni normali forniscono un modo diretto e matematicamente solido per calcolare i parametri m (pendenza della retta) e q (intercetta con l'asse y), assicurando che la somma dei quadrati degli scarti sia minimizzata e offrendo una soluzione esatta facilmente calcolabile anche tramite software.

5 Integrazione numerica

Il problema base affrontato consiste nel calcolare l'integrale di $f(x)$ su $[a, b]$, indicato come

$$\int_a^b f(x)\,dx.$$

L'idea è di approssimare $f(x)$ tramite un polinomio $p(x)$ che passa per $n + 1$ punti dati, che suddividono l'intervallo $[a, b]$ in n sottointervalli. Il polinomio interpolante $p(x)$ è costruito per interpolare i valori di f in questi punti e si esprime come

$$p(x) = \sum_{i=0}^{n} f(x_i)L_i(x),$$

dove $L_i(x)$ sono i polinomi base di Lagrange derivati dai nodi x_i.

L'approssimazione dell'integrale di f viene calcolata integrando il polinomio interpolante, risultando in

$$I[f] = \int_a^b f(x)\,dx \approx \int_a^b p(x)\,dx = \sum_{i=0}^{n} f(x_i)\int_a^b L_i(x)\,dx.$$

Questo metodo converte l'integrale della funzione in una somma pesata dei valori della funzione ai nodi di interpolazione, con i pesi determinati dagli integrali dei polinomi base di Lagrange.

Definiamo grado di esattezza di una formula di quadratura indica il grado massimo di un polinomio tale che la formula di quadratura sia in grado di calcolare l'integrale esatto di ogni polinomio di quel grado o inferiore. Il grado di esattezza di una formula di quadratura Q è il più grande intero r tale che

$$Q_n(p) = I[p]$$

per ogni polinomio $p(x)$ di grado al massimo r.

5.1 Formula del punto medio (formula del rettangolo)

Assumendo che la funzione f sia sufficientemente differenziabile sull'intervallo $[a, b]$, la formula base del punto medio è:

$$\int_a^b f(x)\,dx \approx (b-a) f\left(\frac{a+b}{2}\right),$$

utilizzando il valore della funzione nel punto medio come rappresentativo per l'intero intervallo. Per esaminare l'accuratezza di questa formula, si utilizza l'espansione di Taylor di f attorno al punto medio $\frac{a+b}{2}$, includendo i termini fino alla derivata di secondo ordine:

$$f(x) = f\left(\frac{a+b}{2}\right) + f'\left(\frac{a+b}{2}\right)\left(x - \frac{a+b}{2}\right) + \frac{f''(\eta)}{2}\left(x - \frac{a+b}{2}\right)^2,$$

dove η è un punto nell'intervallo $[a, b]$.

L'errore nell'approssimazione dell'integrale usando la formula del punto medio emerge principalmente dal termine quadratico dell'espansione di Taylor. Dato che il termine lineare si annulla a causa della simmetria dell'intervallo rispetto al punto medio, l'errore dominante è quindi dato da:

$$\begin{aligned}\int_a^b f(x)\,dx &\simeq f\left(\frac{a+b}{2}\right)(b-a) + f'\left(\frac{a+b}{2}\right)\int_a^b \left(x - \frac{a+b}{2}\right)dx + \frac{f''(\eta)}{2}\int_a^b \left(x - \frac{a+b}{2}\right)^2 dx = \\ &\simeq f\left(\frac{a+b}{2}\right)(b-a) + \frac{f''(\eta)}{24}(b-a)^3 = \\ &\simeq Q_0[f] + \text{errore}\end{aligned}$$

dove il calcolo fornisce una stima dell'errore proporzionale al quadrato della lunghezza dell'intervallo e al valore della seconda derivata di f.

Il grado di precisione della formula del punto medio è limitato ai polinomi di primo grado, poiché è esatta per questi. Questo è dovuto al fatto che per i polinomi di grado uno o meno, la seconda derivata è zero, eliminando così l'errore nell'approssimazione dell'integrale.

L'errore della formula di quadratura è nullo per funzioni del tipo $f(x) = c$ (dove c è una costante) o $f(x) = \alpha x + \beta$, ciò significa che la formula di quadratura è in grado di calcolare esattamente l'integrale di tali funzioni, senza alcuna discrepanza tra il valore calcolato dalla formula e il vero valore dell'integrale. Questa formula di quadratura ha grado di precisione 1.

5.2 Formula del trapezio

La formula del trapezio è uno strumento fondamentale nell'integrazione numerica per approssimare l'integrale di una funzione $f(x)$ su un intervallo $[a, b]$. Si basa sulla seguente sostituzione di f con il polinomio interpolatore di grado 1 relativo :

$$\begin{aligned}I = \int_a^b f(x)\,dx &\approx \int_a^b f(x)\sum_{i=0}^{1} f(x_i)L_i(x)\,dx = \\ &= \int_a^b f(x_0)L_0(x) + f(x_1)L_1(x)\,dx = \\ &= f(a)\int_a^b \frac{x-b}{a-b}\,dx + f(b)\int_a^b \frac{x-a}{b-a}dx = \\ &= \frac{f(a)}{a-b}\frac{-(a-b)^2}{2} + \frac{f(b)}{b-a}\frac{(b-a)^2}{2} \\ &= \frac{b-a}{2}[f(a)+f(b)] = Q_1\end{aligned}$$

Questa formula utilizza i valori della funzione agli estremi a e b per formare un trapezio, da cui il nome. La semplicità della formula la rende particolarmente utile per calcoli rapidi e per funzioni che non presentano elevata variabilità nel range di integrazione. Per valutare l'accuratezza di questa approssimazione, consideriamo l'errore associato utilizzando l'analisi di Taylor, supponendo che $f \in C^2([a,b])$:

$$\begin{aligned} I - Q_1 = \int_a^b \frac{f''(\eta_x)}{2}\omega_2(x)\,dx = \int_a^b \frac{f''(\eta_x)}{2}(x-a)(x-b)\,dx = \\ = \frac{f''(\eta)}{2}\int_a^b (x-a)(x-b)\,dx \\ = -f''(\eta)\frac{(b-a)^3}{12} \end{aligned}$$

L'errore si riduce proporzionalmente al cubo della lunghezza dell'intervallo e dipende dalla seconda derivata della funzione, riflettendo come l'approssimazione del trapezio gestisca le curvature della funzione.

La formula di quadratura del trapezio ha grado di precisione 1, significando che è esatta per polinomi fino al primo grado. Questa proprietà la rende adeguata per applicazioni dove la funzione target è approssimativamente lineare su brevi intervalli.

5.3 Formula di Cavalieri-Simpson

Il polinomio interpolatore di grado 2 relativo ai nodi $x_0 = a$, $x_1 = \frac{a+b}{2}$, e $x_2 = b$ permette di ottenere la formula di quadratura di Simpson. Questa formula è particolarmente utile per l'integrazione numerica e si esprime come:

$$Q_2[f] = \frac{b-a}{6}[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]$$

Il calcolo dell'errore per la formula di Simpson, assumendo che f sia almeno quattro volte differenziabile sull'intervallo $[a,b]$ (indicato da $f \in C^4([a,b])$), utilizza il teorema dell'errore e lo sviluppo di Taylor per mostrare che:

$$I[f] - Q_2[f] = -\left(\frac{b-a}{2}\right)^5 \frac{f^{(\mathrm{IV})}(\eta)}{90}$$

dove η è un valore nell'intervallo $[a,b]$. Questo risultato indica che l'errore è proporzionale alla quinta potenza della lunghezza dell'intervallo, riducendosi significativamente per intervalli più piccoli.

La formula di Cavalieri-Simpson ha un grado di esattezza di 3. Ciò significa che può integrare esattamente ogni polinomio fino al terzo grado. Queste formule di quadratura che utilizzano polinomi interpolatori calcolati su nodi equispaziati sono classificate come formule di Newton-Cotes. In generale, le formule di Newton-Cotes di tipo $Q_n[f]$ hanno un ordine di precisione che è $n+1$ per n pari e n per n dispari, dimostrando la loro efficacia nel calcolare integrali di funzioni polinomiali di vari gradi.

5.4 Formula del punto medio composita

L'intervallo $[a,b]$ è suddiviso in m sottointervalli uguali, ognuno con un punto medio. Gli estremi dei sottointervalli sono determinati da:

$$x_i - x_{i-1} = H \quad \text{dove } i = 0, 1, 2, \ldots, m$$

e $H = \frac{b-a}{m}$ rappresenta la lunghezza di ogni sottointervallo.

La formula del punto medio composita approssima l'integrale di $f(x)$ su $[a,b]$ sommando gli integrali approssimati su ciascun sottointervallo:

$$\begin{aligned} \int_a^b f(x)\,dx = \sum_{i=1}^{m}\int_{x_{i-1}}^{x_i} f(x)\,dx = \sum_{i=1}^{m}\left[f\left(\frac{x_{i-1}+x_i}{2}\right)(x_i - x_{i-1}) + f''(\beta_i)\frac{H^3}{24}\right] \\ \simeq H\sum_{i=1}^{m} f\left(\frac{x_i + x_{i-1}}{2}\right) + f''(\beta)m\frac{H^3}{24} \\ \simeq \frac{H}{2}(f(x_0) + (2\sum_{i=1}^{m} f(x_i)) + f(x_m)) + f''(\beta)m\frac{H^3}{24} \\ f''(\beta)m\frac{H^3}{24} = 0 \quad \text{se } H \to 0 \end{aligned}$$

Quest'ultimo termine, $f''(\beta)m\frac{H^3}{24}$, rappresenta l'errore di approssimazione, che diminuisce con la diminuzione di H, o aumentando il numero m dei sottointervalli. Questo termine di errore deriva dall'approssimazione della curvatura della funzione nell'intorno di ciascun punto medio e dimostra che, per funzioni sufficientemente differenziabili e con la seconda derivata limitata, l'errore dell'integrale composito diventa trascurabile all'aumentare di m.

In pratica, questo significa che la formula del punto medio composita diventa un'approssimazione estremamente accurata per l'integrazione numerica di funzioni lisce man mano che il numero di suddivisioni aumenta, avvicinandosi all'integrazione esatta per H che tende a zero.

5.5 Formula del trapezio composita

La formula del trapezio composita è un metodo numerico utilizzato per approssimare l'integrale definito di una funzione su un intervallo $[a, b]$. Questo metodo si basa sulla suddivisione dell'intervallo in m sottointervalli di uguale lunghezza e sull'approssimazione dell'area sotto la curva mediante trapezi. L'integrale può essere espresso come:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{m} \int_{x_{i-1}}^{x_i} f(x)\,dx$$

Applicando la formula del trapezio a ciascun sottointervallo, si ottiene:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{m} \left(\frac{f(x_i) + f(x_{i-1})}{2}(x_i - x_{i-1}) - f''(\beta_i)\frac{H^3}{12} \right)$$

dove $x_i = a + ih$ e $h = \frac{b-a}{m}$ è la larghezza di ciascun sottointervallo. Inoltre, $(x_i - x_{i-1}) \to H$, perciò:

$$\int_a^b f(x)\,dx \approx \frac{H}{2} \sum_{i=1}^{m} (f(x_i) + f(x_{i-1})) - f''(\beta)m\frac{H^3}{12}$$

$$\begin{aligned}\sum_{i=1}^{m} (f(x_i) + f(x_{i-1})) &= f(x_1) + f(x_0) + f(x_2) + f(x_1) + f(x_3) + f(x_2) + \cdots \\ &\quad + f(x_{m-1}) + f(x_{m-2}) + f(x_m) + f(x_{m-1}) = \\ &= f(x_0) + 2\sum_{i=1}^{m-1} f(x_i) + f(x_m)\end{aligned}$$

L'approssimazione diventa più accurata aumentando il numero di sottointervalli m, riducendo così la larghezza di ciascun sottointervallo h. Per determinare il numero di sottointervalli m necessario per ottenere un errore inferiore a una tolleranza ε, utilizziamo la seguente relazione per l'errore:

$$|\text{Errore}| = \left| f''(\beta)\frac{H^3}{24}m \right| = \left| f''(\beta)\frac{(b-a)^3}{24m^3}m \right| \leq M\frac{(b-a)^3}{24m^2} \leq \varepsilon$$

$$\forall m \in \mathbb{N} : m > \sqrt{\frac{M(b-a)^3}{24\varepsilon}}$$

Un'ulteriore comprensione dell'errore può essere ottenuta utilizzando lo sviluppo di Taylor. Consideriamo i due sviluppi di Taylor per $f(x+h)$ e $f(x-h)$:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)h^2}{2} + \frac{f'''(x)h^3}{3!} + \cdots$$

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)h^2}{2} - \frac{f'''(x)h^3}{3!} + \cdots$$

Sommando queste due espansioni, otteniamo:

$$f(x+h) + f(x-h) = 2f(x) + f''(x)h^2 + \mathcal{O}(h^4)$$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}\mathcal{O}(h^2)$$

Questo mostra che l'errore è dell'ordine di h^2, confermando che la precisione della formula del trapezio composita migliora con il numero di sottointervalli.

In conclusione, la formula del trapezio composita offre un metodo semplice ed efficace per l'approssimazione numerica degli integrali. La scelta del numero di sottointervalli m è cruciale per controllare l'errore dell'approssimazione. Utilizzando lo sviluppo di Taylor, possiamo ottenere una stima più precisa dell'errore e quindi determinare il valore ottimale di m per garantire una precisione desiderata.

5.6 Formula di Cavalieri-Simpson composita

La formula di Cavalieri-Simpson composita è un metodo efficace per l'approssimazione numerica di integrali definiti, estendendo la formula di Simpson a un intervallo suddiviso in m sottointervalli. Questo metodo è particolarmente utile per funzioni che sono continuamente differenziabili almeno quattro volte nell'intervallo di integrazione, dato che l'errore associato dipende dalla quarta derivata della funzione. La formula è espressa come segue:

$$\int_a^b f(x)\,dx = \sum_{i=1}^{m} \int_{x_{i-1}}^{x_i} f(x)\,dx =$$

$$= \frac{H}{6}\left[f(x_0) + 2\sum_{i=1}^{m-1} f(x_i) + 4\sum_{i=1}^{m}\left(\frac{x_{i-1}+x_i}{2}\right) + f(x_m)\right] - \frac{b-a}{180}\left(\frac{H}{2}\right)^4 f^{(\mathrm{IV})}(\beta)$$

$\left(\frac{H}{2}\right)^4$ tende a 0 per $H \to 0$

6 Risoluzione di sistemi lineari

I metodi diretti, in aritmetica esatta, forniscono la soluzione in un numero finito di passi. Un esempio classico di metodo diretto è l'eliminazione di Gauss, che trasforma la matrice del sistema in una forma triangolare superiore e risolve il sistema tramite sostituzione all'indietro. Altri metodi diretti includono la fattorizzazione LU e la decomposizione di Cholesky. I metodi iterativi, invece, forniscono una successione di soluzioni approssimate che converge alla soluzione esatta. Questi metodi sono particolarmente utili per sistemi di grandi dimensioni o per sistemi sparsi. Tra i metodi iterativi più noti vi sono il metodo di Jacobi, il metodo di Gauss-Seidel e il metodo del sottospazio di Krylov (come il metodo del gradiente coniugato).
I metodi diretti e iterativi offrono approcci complementari per la risoluzione di sistemi lineari, con i primi che garantiscono una soluzione esatta in un numero finito di operazioni e i secondi che forniscono soluzioni approssimate migliorando progressivamente l'accuratezza ad ogni iterazione.

6.1 Condizionamento e precondizionamento

Tra i concetti chiave di questo capitolo vi sono il vettore residuo, il condizionamento delle matrici e le tecniche di precondizionamento. Questi concetti aiutano a comprendere e migliorare l'accuratezza e la stabilità delle soluzioni dei sistemi lineari.
Il vettore residuo r è definito come

$$r = b - A\tilde{x} \neq 0$$

dove $\tilde{x}$ non è una soluzione del sistema $Ax = b$. L'errore relativo sulle soluzioni è espresso dalla relazione

$$\frac{\|\delta x\|}{\|x\|} \leq \mathrm{cond}(A)\frac{\|r\|}{\|b\|}$$

che indica come l'accuratezza della soluzione sia legata al condizionamento della matrice A. Il condizionamento di una matrice A è dato da

$$\mathrm{cond}(A) = \|A\|\|A^{-1}\|$$

e fornisce una misura di quanto la soluzione di un sistema lineare sia sensibile ai piccoli cambiamenti nei dati di input. Un esempio tipico di matrici con elevato condizionamento include la matrice di Hilbert e la matrice di Vandermonde, che sono note per la loro instabilità numerica.
Consideriamo una matrice diagonale $D^{(m)}$ con elementi diagonali pari a 10^{-1}. Il determinante di $D^{(m)}$ è

$$\det(D^{(n)}) \approx 10^{-n}$$

mentre il condizionamento è

$$\text{cond}_\infty(D^{(n)}) = \|D^{(n)}\|_\infty \|D^{(n)^{-1}}\|_\infty = 10^{-1} \cdot 10 = 1$$

indicando un buon condizionamento nonostante il determinante piccolo. La matrice inversa di $D^{(m)}$ è

$$D^{(m)^{-1}} = \begin{pmatrix} 10 & 0 & \cdots & 0 \\ 0 & 10 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 10 \end{pmatrix}$$

quindi

$$\det(D^{(n)^{-1}}) \approx 10^n$$

Un'osservazione importante è che

$$1 = \|I\| = \|A \cdot A^{-1}\| \leq \|A\| \|A^{-1}\| = \text{cond}(A)$$

sottolineando che il condizionamento di una matrice è sempre maggiore o uguale a uno. Per illustrare il precondizionamento, consideriamo la matrice

$$A = \begin{pmatrix} \varepsilon & 0 \\ 0 & \frac{1}{\varepsilon} \end{pmatrix}$$

con $0 < \varepsilon \ll 1$. Il suo determinante è

$$\det(A) = 1$$

e l'inversa è

$$A^{-1} = \begin{pmatrix} \frac{1}{\varepsilon} & 0 \\ 0 & \varepsilon \end{pmatrix}$$

Tuttavia, il condizionamento di A è

$$\text{cond}_1(A) = \|A\|_1 \|A^{-1}\|_1 = \max(\varepsilon, \frac{1}{\varepsilon}) \max(\frac{1}{\varepsilon}, \varepsilon) = \frac{1}{\varepsilon^2} \gg 1$$

indicando un cattivo condizionamento.
Applicando il precondizionamento con la matrice

$$C = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon^2 \end{pmatrix}$$

otteniamo

$$CAx = Cb$$

dove

$$\tilde{A} = CA = \begin{pmatrix} \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix}$$

$$\tilde{A}^{-1} = \begin{pmatrix} 1/\varepsilon & 0 \\ 0 & 1/\varepsilon \end{pmatrix}$$

Il condizionamento di $\tilde{A}$ è

$$\text{cond}_1(\tilde{A}) = \|\tilde{A}\| \|\tilde{A}^{-1}\| = \varepsilon \, \frac{1}{\varepsilon} = 1$$

migliorando significativamente la stabilità numerica del sistema.

In sintesi, il vettore residuo misura la differenza tra il termine noto e il prodotto della matrice con una soluzione approssimata, la precisione della soluzione è legata al condizionamento della matrice, e il precondizionamento è una tecnica efficace per trasformare un sistema lineare in uno equivalente ma meglio condizionato. Questi concetti sono essenziali per garantire accuratezza e stabilità nelle soluzioni numeriche dei sistemi lineari.

6.2 Metodi diretti

6.2.1 Risoluzione dei sistemi lineari con matrice diagonale

Questo tipo di sistema è particolarmente semplice da risolvere grazie alla struttura della matrice A. Consideriamo un sistema lineare:

$$Ax = b$$

dove A è una matrice diagonale. La matrice A ha la forma:

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

e il vettore b è:

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

mentre il vettore incognito x è:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Poiché A è diagonale, ogni equazione del sistema può essere risolta in modo indipendente. Ogni equazione del sistema è:

$$a_{ii}x_i = b_i \quad \text{per} \quad i = 1, \ldots, n$$

da cui possiamo ricavare direttamente:

$$x_i = \frac{b_i}{a_{ii}}$$

Risoluzione per ogni incognita:

$$\begin{cases} x_1 = \frac{b_1}{a_{11}} \\ x_2 = \frac{b_2}{a_{22}} \\ \vdots \\ x_m = \frac{b_n}{a_{nn}} \end{cases}$$

Il vettore x risulta composto dalle soluzioni x_i calcolate come sopra.
Il costo computazionale di questo algoritmo è molto basso. Poiché ogni componente x_i può essere calcolato indipendentemente dagli altri e richiede solo un'operazione di divisione, il costo totale è di n divisioni. Questo rappresenta un costo computazionale di:

In sintesi, la forma diagonale della matrice elimina la necessità di operazioni complesse come la fattorizzazione o la decomposizione, che sono necessarie per matrici più generali.

6.2.2 Risoluzione dei sistemi lineari con matrice triangolare inferiore

Questo tipo di sistema è relativamente semplice da risolvere grazie alla struttura della matrice A. Consideriamo un sistema lineare:

$$Ax = b$$

dove A è una matrice triangolare inferiore:

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 & \\ a_{21} & a_{22} & \cdots & 0 & \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{nn} \end{pmatrix}$$

e il vettore b è:

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

mentre il vettore incognito x è:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Per risolvere il sistema, utilizziamo l'algoritmo di sostituzione in avanti, che risolve il sistema un'equazione alla volta, partendo dalla prima riga fino all'ultima.

1. Prima equazione:

   $$a_{11}x_1 = b_1$$

   da cui possiamo ricavare direttamente:

   $$x_1 = \frac{b_1}{a_{11}}$$

2. Seconda equazione:

   $$a_{21}x_1 + a_{22}x_2 = b_2$$

   Sostituendo x_1 già trovato:

   $$x_2 = \frac{b_2 - a_{21}x_1}{a_{22}}$$

3. Equazioni successive:

   $$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

   sostituendo x_1 e x_2:

   $$x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}$$

4. Equazione generica per la i-esima equazione:

   $$x_1 = \frac{b_1}{a_{11}}$$

   $$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}}$$

   dove $i = 2 \dots n$

Il costo computazionale dell'algoritmo di sostituzione in avanti è relativamente basso. Richiede n divisioni e una somma e un prodotto di termini per ogni riga fino alla i-esima equazione, il che implica un costo complessivo di:

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

che rappresenta un costo computazionale di ordine $O(n^2)$.

In sintesi, la risoluzione dei sistemi lineari con matrice triangolare inferiore è efficiente grazie alla struttura della matrice. L'algoritmo di sostituzione in avanti permette di risolvere il sistema in modo sequenziale, equazione per equazione, con un costo computazionale contenuto. Questo metodo è fondamentale per la soluzione di sistemi lineari derivanti dalla fattorizzazione LU (lower-upper) di matrici generiche.

6.2.3 Risoluzione dei sistemi lineari con matrice triangolare superiore

Questo tipo di sistema è relativamente semplice da risolvere grazie alla struttura della matrice A. Consideriamo un sistema lineare $Ax = b$ dove A è una matrice triangolare superiore. La matrice A ha la forma:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & a_{nn} \end{pmatrix}$$

e il vettore b è:

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

mentre il vettore incognito x è:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Per risolvere il sistema, utilizziamo l'algoritmo di sostituzione all'indietro, che risolve il sistema un'equazione alla volta, partendo dall'ultima riga fino alla prima. L'ultima equazione è $a_{nn}x_n = b_n$ da cui possiamo ricavare direttamente $x_n = \frac{b_n}{a_{nn}}$. La penultima equazione è:

$$a_{n-1n-1}x_{n-1} + a_{n-1n}x_n = b_{n-1}$$

Sostituendo x_n già trovato, otteniamo:

$$x_{n-1} = \frac{b_{n-1} - a_{n-1n}x_n}{a_{n-1n-1}}$$

Per la i-esima equazione (dove $i = n-1, \ldots, 1$), abbiamo

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$$

Il costo computazionale dell'algoritmo di sostituzione all'indietro è relativamente basso. Richiede n divisioni e una somma di termini per ogni riga dalla i-esima equazione fino alla prima, il che implica un costo complessivo di:

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

che rappresenta un costo computazionale di ordine $O(n^2)$. Inoltre, bisogna considerare il costo dei prodotti. Per ogni riga dalla i-esima equazione fino alla prima, il costo dei prodotti è:

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Il costo totale per le somme e i prodotti è quindi:

$$2 \cdot \frac{n(n-1)}{2} = n(n-1)$$

che è anch'esso di ordine $O(n^2)$.

In sintesi, la risoluzione dei sistemi lineari con matrice triangolare superiore è efficiente grazie alla struttura della matrice. L'algoritmo di sostituzione all'indietro permette di risolvere il sistema in modo sequenziale, equazione per equazione, con un costo computazionale contenuto. Questo metodo è fondamentale per la soluzione di sistemi lineari derivanti dalla fattorizzazione LU di matrici generiche.

6.2.4 Metodo di eliminazione di Gauss

Il metodo di eliminazione di Gauss è una tecnica fondamentale per risolvere sistemi di equazioni lineari. Questo metodo trasforma un sistema di equazioni lineari in una forma triangolare superiore, rendendo il sistema più semplice da risolvere tramite sostituzione all'indietro. Consideriamo il sistema lineare:

$$\begin{cases} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 = \frac{11}{6} \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 = \frac{13}{12} \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 = \frac{47}{60} \end{cases}$$

che può essere rappresentato in forma matriciale come $Ax = b$:

$$A = \begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix}, \quad b = \begin{pmatrix} 11/6 \\ 13/12 \\ 47/60 \end{pmatrix}$$

Per eliminare il primo elemento sotto la diagonale, calcoliamo $m_{21} = \frac{a_{21}}{a_{11}} = \frac{1/2}{1} = \frac{1}{2}$ e aggiorniamo la seconda riga: Riga 2 $\leftarrow$ Riga 2 $- m_{21} \cdot$ Riga 1. Otteniamo:

$$\begin{pmatrix} 1 & 1/2 & 1/3 \\ 0 & 1/12 & 1/12 \\ 1/3 & 1/4 & 1/5 \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} 11/6 \\ 1/6 \\ 47/60 \end{pmatrix}$$

Per eliminare il secondo elemento sotto la diagonale, calcoliamo $m_{31} = \frac{a_{31}}{a_{11}} = \frac{1/3}{1} = \frac{1}{3}$ e aggiorniamo la terza riga: Riga 3 $\leftarrow$ Riga 3 $- m_{31} \cdot$ Riga 1. Otteniamo:

$$\begin{pmatrix} 1 & 1/2 & 1/3 \\ 0 & 1/12 & 1/12 \\ 0 & 1/12 & 1/20 \end{pmatrix}, \quad b^{(2)} = \begin{pmatrix} 11/6 \\ 1/6 \\ 1/180 \end{pmatrix}$$

Per eliminare l'elemento rimanente, calcoliamo $m_{32} = \frac{a_{32}}{a_{22}} = \frac{1/12}{1/12} = 1$ e aggiorniamo la terza riga: Riga 3 $\leftarrow$ Riga 3 $- m_{32} \cdot$ Riga 2. Otteniamo:

$$\begin{pmatrix} 1 & 1/2 & 1/3 \\ 0 & 1/12 & 1/12 \\ 0 & 0 & 1/180 \end{pmatrix} b^{(3)} = \begin{pmatrix} 11/6 \\ 1/6 \\ 1 \end{pmatrix}$$

Dopo aver ottenuto la forma triangolare superiore, risolviamo il sistema tramite sostituzione all'indietro:

$$\begin{cases} x_3 = 180 \cdot 1 = 180 \\ x_2 = 12\left(\frac{1}{6} - \frac{1}{12} \cdot 180\right) = 1 \\ x_1 = 1\left(\frac{11}{6} - \frac{1}{2} \cdot 1 - \frac{1}{3} \cdot 180\right) = 1 \end{cases}$$

Il metodo di eliminazione di Gauss consiste nel trasformare la matrice A in una matrice triangolare superiore U tramite operazioni elementari di riga, aggiornare il vettore b corrispondentemente e risolvere il sistema triangolare superiore risultante con la sostituzione all'indietro. Per verificare il metodo, possiamo generare una matrice $A \in \mathbb{R}^{n \times n}$ e un vettore x (ad esempio ones(n,1)) con elementi noti, calcolare il vettore termine noto $b = Ax$ e risolvere il sistema $Ax = b$ per ritrovare x.
Consideriamo il sistema $Ax = b$, dove $A \in \mathbb{R}^{n \times n}$ e $b, x \in \mathbb{R}^n$. Per ottenere una matrice triangolare superiore, applichiamo la matrice M ad A e b:

$$MAx = Mb \quad \text{con} \quad A^{(1)} = MA \quad \text{e} \quad b^{(1)} = Mb$$

In $A^{(1)}$, eliminiamo gli elementi sotto la diagonale della prima colonna:

$$A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{m2}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_m^{(1)} \end{pmatrix}$$

Per eliminare l'elemento $a_{21}^{(1)}$, calcoliamo il moltiplicatore m_{21}:

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}}$$

Aggiorniamo la seconda riga Riga 2 $\leftarrow$ Riga 2 $- m_{21} \cdot$ Riga 1:

$$a_{2j}^{(2)} = a_{2j}^{(1)} - m_{21} a_{1j}^{(1)} \quad \text{per} \quad j = 1, \ldots, n$$

$$b_2^{(2)} = b_2^{(1)} - m_{21} b_1^{(1)}$$

In generale con $i = 2 \ldots n$ e $j = 2 \ldots n$:

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$$

Il costo computazionale di questa operazione è di $(n-1)$ divisioni.

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1} a_{ij}^{(1)}$$

Il costo computazionale di questa operazione è di $(n-1)^2$ somme e $(n-1)^2$ prodotti.

$$b_i^{(2)} = b_i^{(1)} - m_{i1} b_1^{(1)}$$

Il costo computazionale di questa operazione è di $(n-1)$ somme e $(n-1)$ prodotti.
Aggiorniamo l'intera matrice e il vettore termine noto:

$$A^{(2)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}, \quad b^{(2)} = \begin{pmatrix} b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

In $A^{(1)}$, eliminiamo gli elementi sotto la diagonale della seconda colonna:

$$m_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} \quad \text{per} \quad i = 3 \ldots n$$

Il costo computazionale di questa operazione è di $(n-2)$ divisioni. Aggiorniamo la $i-$esima equazione i equazione $\leftarrow i$ equazione $- m_{i2} \cdot$ Seconda Equazione:

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2} a_{2j}^{(2)} \quad \text{per} \quad j = 3 \ldots n$$

Il costo computazionale di questa operazione è di $(n-2)^2$ somme e $(n-2)^2$ prodotti.

$$b_i^{(3)} = b_i^{(2)} - m_{i2} b_2^{(2)}$$

Il costo computazionale di questa operazione è di $(n-2)$ somme e $(n-2)$ prodotti.

$$A^{(2)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & a_{34}^{(3)} & \cdots & a_{3n}^{(3)} \\ 0 & 0 & a_{43}^{(3)} & a_{44}^{(3)} & \cdots & a_{4n}^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & a_{n4}^{(3)} & \cdots & a_{nn}^{(3)} \end{pmatrix}$$

$$m_{ij} = \frac{a_{ij}^{(k)}}{a_{jj}^{(k)}} \quad \text{per} \quad i = j+1, \ldots, m \quad \text{e} \quad j = k, \ldots, m$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ij} a_{jj}^{(k)}$$

$$b_i^{(k+1)} = b_i^{(k)} - m_{ij} b_j^{(k)}$$

L'algoritmo generale per la trasformazione della matrice in forma triangolare superiore sarà:

1. Calcolo dei moltiplicatori:
$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \quad \text{per} \quad i = k+1, \ldots, n$$
dove $a_{kk}^{(k)} \neq 0$.

2. Aggiornamento delle righe:
$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \quad \text{per} \quad j = k+1, \ldots, n$$
$$b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$$

3. Iterazione:
$$\text{for} \quad k = 1, \ldots, n-1$$

Costo computazionale totale:
Divisioni:
$$\sum_{k=1}^{n-1} (n-k) = \frac{n(n-1)}{2}$$
Somme e Prodotti:
$$\sum_{k=1}^{n-1} (n-k)^2 = \frac{n(n-1)(2n-1)}{6}$$
Costo complessivo: $O(n^3)$.

6.2.5 Fattorizzazione LU

L'algoritmo di eliminazione di Gauss può essere utilizzato per ottenere la fattorizzazione LU della matrice A. La matrice L è formata dai moltiplicatori utilizzati durante il processo, mentre la matrice U è la matrice triangolare superiore ottenuta:

$$A = LU$$

Per risolvere il sistema $Ax = b$, possiamo risolvere $LUx = b$ in due passaggi:

$$Ly = b$$

Risolvendo per y tramite sostituzione in avanti (costo $O(n^2)$).

$$Ux = y$$

Risolvendo per x tramite sostituzione all'indietro (costo $O(n^2)$).

6.2.6 Inversione di una matrice

Per calcolare l'inversa di una matrice A, sfruttiamo il fatto che $A^{-1} \cdot A = I$, dove I è la matrice identità. Questo implica che dobbiamo risolvere n sistemi lineari della forma $Ax_i = e_i$, dove e_i sono i vettori colonna della matrice identità.

$$A \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} = \begin{pmatrix} e_1 & e_2 & \cdots & e_n \end{pmatrix}$$

dove e_i sono i vettori della base canonica. L'inversa A^{-1} è data da:

$$A^{-1} = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$$

Mostriamo come risolvere n sistemi lineari con la stessa matrice A e diversi termini noti. Ogni colonna della matrice identità I viene presa come vettore e_i per formare il sistema:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & & \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

Risolvere n sistemi lineari con la stessa matrice A ma con diversi termini noti e_i può essere computazionalmente intensivo. Pertanto, conviene fattorizzare la matrice A una sola volta utilizzando la strategia LU. Dopo aver ottenuto la fattorizzazione $A = LU$, possiamo risolvere ciascun sistema in modo efficiente applicando la sostituzione in avanti e all'indietro.

$$Ly_i = e_i \quad \text{e poi} \quad Ux_i = y_i$$

Fattorizzazione di Doolittle

La fattorizzazione di Doolittle permette di decomporre una matrice A non singolare nel prodotto di due matrici, una triangolare inferiore L con elementi diagonali pari a 1 e una triangolare superiore U, espressa come $A = LU$. Questo processo è utile per risolvere sistemi lineari e calcolare l'inversa di una matrice. L'algoritmo per ottenere questa fattorizzazione si basa sul calcolo degli elementi di L e U attraverso un processo iterativo.
Per ogni riga i da 1 a n e per ogni colonna j da 1 a n, si calcolano gli elementi di U con la formula

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$$

e gli elementi di L per $j > i$ con

$$l_{ji} = \frac{a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki}}{u_{ii}}$$

per $k = 1 \dots n$.
Per una matrice A non singolare, esiste ed è unica la fattorizzazione $A = LU$. Questa fattorizzazione è ottenuta risolvendo n^2 equazioni ricavate dalle m^2 equazioni del sistema originale.
Per calcolare l'inversa di una matrice A, utilizziamo la fattorizzazione LU. Poiché $A^{-1}A = I$, possiamo risolvere n sistemi lineari $Ax_i = e_i$, dove e_i sono i vettori della base canonica. Questo processo viene semplificato fattorizzando A come LU e risolvendo i sistemi lineari tramite la sostituzione in avanti e all'indietro.
Consideriamo la matrice:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

La fattorizzazione di Doolittle produce:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Inoltre, per risolvere un sistema lineare $Ax = b$, possiamo usare la fattorizzazione LU. Dapprima, risolviamo il sistema $Ly = b$ utilizzando la sostituzione in avanti, e successivamente risolviamo $Ux = y$ utilizzando la sostituzione all'indietro. Questo approccio semplifica il calcolo delle soluzioni e riduce il costo computazionale. Quando si calcola l'inversa di una matrice, il processo implica la risoluzione di n sistemi lineari con la stessa matrice ma con termini noti diversi. Pertanto, conviene fattorizzare la matrice e applicare la strategia LU. La relazione $A^{-1}A = I$ implica che per calcolare l'inversa di una matrice di ordine n si devono risolvere n sistemi lineari. Utilizzando la fattorizzazione LU, possiamo semplificare questo processo risolvendo i sistemi con la sostituzione in avanti e all'indietro. L'efficienza della risoluzione dei sistemi lineari dipende dalla struttura della matrice A. Ad esempio, se A è una matrice simmetrica definita positiva, può essere fattorizzata come $A = BB^T$. Se A è una matrice tridiagonale, la risoluzione del sistema può essere eseguita in $O(n)$ operazioni, rendendola estremamente efficiente. In generale, la scelta del metodo di risoluzione dipende dalle proprietà della matrice e dalle specifiche esigenze computazionali del problema.

6.3 Metodi iterativi

I metodi iterativi vengono utilizzati soprattutto quando le dimensioni della matrice rendono impraticabili i metodi diretti. Un metodo iterativo lineare genera una successione di vettori $\{x^{(k)}\}$ che approssimano la soluzione x del sistema $Ax = b$. Sia $x^{(k)}$ una successione di vettori in $\mathbb{R}^n$; essa converge al vettore $x \in \mathbb{R}^n$ se esiste una norma vettoriale per cui $\lim_{k\to\infty} \|x - x^{(k)}\| = 0$. La convergenza si verifica componente per

componente, ovvero $|x_i - x_i^{(k)}| \leq \|x - x^{(k)}\|_\infty \leq c\|x - x^{(k)}\|_1 \to 0$ per $i = 1, \ldots, n$. Questo implica che la convergenza è soddisfatta componente per componente. Noi cercheremo metodi iterativi consistenti. Un metodo iterativo si dice consistente se, quando $x^{(k)}$ è esattamente la soluzione x, allora anche l'iterazione successiva $x^{(k+1)}$ rimane x. Formalmente, se $x^{(k)} = x$, allora $x^{(k+1)} = x$.
Un metodo iterativo lineare ha la forma generale:

$$x^{(k+1)} = Bx^{(k)} + q$$

dove B è una matrice e q è un vettore. Un metodo iterativo è considerato consistente se la soluzione x del sistema lineare soddisfa $x = Bx + q$.
Per dimostrare la consistenza, consideriamo la soluzione x del sistema lineare $Ax = b$. Si può dimostrare che:

$$q = (I - B)A^{-1}b$$

Supponiamo che x sia una soluzione di $Ax = b$. Allora, x può essere espresso come $x = A^{-1}b$. Sostituendo questa espressione nella forma iterativa, otteniamo:

$$x = Bx + q$$

Sostituendo $x = A^{-1}b$ nell'equazione sopra, si ha:

$$A^{-1}b = BA^{-1}b + q$$

Per soddisfare questa equazione, deve essere:

$$q = (I - B)A^{-1}b$$

Ad ogni passo iterativo, il costo computazionale principale è dato dal prodotto matrice-vettore $Bx^{(k)}$. Se B è una matrice di dimensioni $n \times n$ e $x^{(k)}$ è un vettore di dimensione n, il costo di questo prodotto è $O(n^2)$.
Un metodo iterativo diventa vantaggioso rispetto a un metodo diretto quando il numero di passi necessari per raggiungere la precisione richiesta è inferiore al numero di variabili del sistema. Formalmente, se per raggiungere la precisione desiderata $\|x^{(\overline{n})} - x\| < \varepsilon$ il numero di passi $\overline{n}$ è inferiore a n, il metodo iterativo è preferibile. Questo vantaggio è particolarmente significativo per sistemi lineari di grandi dimensioni $A \in \mathbb{R}^{n \times n}$. Nei metodi diretti, come la decomposizione LU o la fattorizzazione QR, il costo computazionale cresce rapidamente con n, tipicamente come $O(n^3)$. In contrasto, i metodi iterativi possono offrire soluzioni più efficienti in termini di tempo computazionale per sistemi molto grandi, purché convergano sufficientemente rapidamente.
Il teorema fondamentale afferma che la successione dei vettori $x^{(k)}$ ottenuta con un metodo iterativo della forma $x^{(k+1)} = Bx^{(k)} + q$ converge alla soluzione x del sistema $Ax = b$ se e solo se il raggio spettrale di B, indicato come $\rho(B)$, è minore di 1, cioè $\rho(B) < 1$.
Per dimostrare questo teorema, consideriamo l'equazione iterativa $x = Bx + q$. Sottraendo questa dall'equazione iterativa al passo $k + 1$, si ottiene:

$$x - x^{(k+1)} = B(x - x^{(k)})$$

Indichiamo $e^{(k)} = x - x^{(k)}$, allora possiamo riscrivere l'equazione come:

$$e^{(k+1)} = Be^{(k)}$$

Iterando questa relazione, otteniamo:

$$e^{(k+1)} = Be^{(k)} = BBe^{(k-1)} = B^2e^{(k-1)} = \cdots = B^{(k+1)}e^{(0)}$$

dove $e^{(0)}$ è il vettore iniziale di innesco del metodo. Affinché $e^{(k+1)}$ converga a zero quando $k \to +\infty$, è necessario che:

$$\lim_{k \to +\infty} B^{k+1}e^{(0)} = 0$$

Questo implica che il raggio spettrale di B, $\rho(B)$, deve essere minore di 1, poiché solo in questo caso la norma di B^{k+1} tende a zero.
Una condizione sufficiente affinché $\rho(B) < 1$ è che la norma di B sia minore di 1, ossia $||B|| < 1$. Questo perché se $||B|| < 1$, allora il raggio spettrale $\rho(B)$ è necessariamente minore di 1. La dimostrazione di questa affermazione si basa sul fatto che per qualsiasi vettore x,

$$||Bx|| = |\lambda|||x||$$

dove λ è un autovalore di B. Da questa relazione si ottiene che $|\lambda| \leq ||B||$, e quindi se $||B|| < 1$, anche $|\lambda| < 1$, implicando $\rho(B) < 1$.
D'altra parte, una condizione necessaria per avere $\rho(B) < 1$ è che il determinante di B, $\det(B)$, sia minore di 1. Tuttavia, questa condizione da sola potrebbe non essere sufficiente a garantire la convergenza della successione.

Costruzione di metodi iterativi lineari

Per la costruzione di metodi iterativi lineari, consideriamo la decomposizione della matrice A come $A = P - N$, dove P è una matrice invertibile e N contiene il resto della decomposizione. Riscrivendo il sistema $Ax = b$ come:

$$(P - N)x = b$$

e successivamente,

$$Px - Nx = b$$

si ottiene la forma iterativa:

$$Px = Nx + b$$

da cui

$$x = P^{-1}Nx + P^{-1}b$$

Indicando con $B = P^{-1}N$ e $q = P^{-1}b$, la formula iterativa diventa:

$$x^{(k+1)} = Bx^{(k)} + q$$

dove $x^{(0)}$ è un vettore iniziale assegnato. Questo metodo iterativo si ripete fino a convergenza, sotto la condizione che $\rho(B) < 1$.
Un'ulteriore osservazione importante è che la matrice P dovrebbe essere scelta in modo tale da essere invertibile con un basso costo computazionale, garantendo così l'efficienza del metodo iterativo. In sintesi, la convergenza dei metodi iterativi lineari dipende fortemente dalle proprietà spettrali della matrice B. Le condizioni sufficienti e necessarie forniscono criteri utili per la progettazione di algoritmi efficienti e convergenti. La scelta della matrice P è cruciale per bilanciare tra convergenza e complessità computazionale.

6.3.1 Metodo di Jacobi

Consideriamo un sistema lineare $Ax = b$ dove la matrice A può essere scomposta nella somma di una matrice diagonale D e una matrice C contenente gli elementi fuori diagonale, cioè:

$$A = D + C$$

La matrice D contiene solo gli elementi diagonali di A, mentre C contiene gli elementi fuori diagonale:

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} \quad C = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix}$$

dove $a_{ii} \neq 0$ (se succede scambiare righe o colonne). Il metodo di Jacobi itera sulla seguente formula:

$$x^{(k+1)} = Bx + q = -D^{-1}Cx^{(k)} + D^{-1}b = -D^{-1}C\left(b - Cx^{(k)}\right)$$

per $k = 0, 1, 2, \ldots$ fino a quando la convergenza è raggiunta. Esplicitando D^{-1}, otteniamo:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right) \quad \text{per } i = 1, 2, \ldots, n$$

$$B = \begin{pmatrix} 0 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \frac{a_{23}}{a_{22}} & \cdots & \frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \cdots & \cdots & 0 \end{pmatrix}$$

Convergenza del metodo di Jacobi

Il metodo di Jacobi è intrinsecamente parallelo, il che lo rende vantaggioso per il calcolo parallelo e distribuito, perché le componenti sono tutte indipendenti. La convergenza del metodo di Jacobi può essere garantita se la matrice A è diagonalmente dominante. Una matrice è detta diagonalmente dominante stretta se:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad \text{per tutte le righe } i = 1, 2, \ldots, n$$

o, analogamente, se:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ji}| \quad \text{per tutte le colonne } i = 1, 2, \ldots, n$$

In sintesi, il metodo di Jacobi richiede la scomposizione della matrice A nelle componenti D e C, e procede iterativamente a calcolare una soluzione approssimata $x^{(k)}$ fino a quando la differenza tra le iterazioni successive è sufficientemente piccola. La convergenza è garantita se la matrice A è diagonalmente dominante.

6.3.2 Metodo di Gauss-Seidel

Il metodo di Gauss-Seidel è una modifica del metodo di Jacobi che utilizza i valori aggiornati di x non appena sono disponibili. La matrice A è decomponibile come $A = D + E + F$, dove D è la matrice diagonale, E è la matrice triangolare inferiore stretta e F è la matrice triangolare superiore stretta. La decomposizione è la seguente:

$$A = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & 0 & \cdots & 0 \\ 0 & \alpha_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_{nn} \end{pmatrix} + \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \alpha_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & \alpha_{12} & \cdots & \alpha_{1n} \\ 0 & 0 & \cdots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

Il metodo iterativo di Gauss-Seidel è definito dalla seguente relazione di ricorrenza:

$$x^{(k+1)} = -(E + D)^{-1} F x^k + (E + D)^{-1} b = (D + E)^{-1}(b - F x^{(k)}) \quad \text{per} \quad k = 0, 1, 2, \ldots$$

Ogni iterazione richiede il calcolo del vettore $x^{(k+1)}$ utilizzando:

$$x_i^{(k+1)} = \frac{1}{\alpha_{ii}} \left(b_i - \sum_{j=1}^{i-1} \alpha_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} \alpha_{ij} x_j^{(k)} \right) \quad \text{per} \quad i = 1, 2, \ldots, n$$

Il metodo iterativo di Gauss-Seidel è di tipo sequenziale, ma basta memorizzare un solo vettore soluzione da aggiornare durante il passo di iterazione.

Convergenza dei metodi di Gauss-Seidel e di Jacobi

Entrambi i metodi, Jacobi e Gauss-Seidel, convergono se la matrice A è strettamente diagonale dominante.

$$|\alpha_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{m} |\alpha_{ij}| \quad \text{per} \quad i = 1, 2, \ldots, m$$

Tuttavia, un metodo può essere più veloce dell'altro. In alcune situazioni, un metodo può convergere mentre l'altro no; in altre, entrambi i metodi possono convergere o entrambi possono non convergere. La scelta del metodo dipende dalle proprietà specifiche della matrice A e dalle caratteristiche del problema da risolvere.

Criteri di Arresto

Per determinare quando interrompere le iterazioni, possiamo utilizzare due criteri di arresto comuni:

- Criterio del residuo: $\|Ax^{(k)} - b\| < \varepsilon$
- Criterio dell'incremento: $\|x^{(k+1)} - x^{(k)}\| < \varepsilon$

7 Norme

Un'applicazione $\mathbb{R}^n \to \mathbb{R}$ chiamata norma, $\|x\|$, se verifica le seguenti condizioni:

1. $\|x\| > 0 \quad \forall x \in \mathbb{R}^n$
2. $\|x\| = 0$ se e solo se $x = 0$
3. $\|\alpha x\| = |\alpha| \cdot \|x\| \quad \alpha \in \mathbb{R}$
4. $\forall x, y \in \mathbb{R}^n \quad \|x + y\| \leq \|x\| + \|y\|$

Una norma p con $1 \leq p \leq +\infty$, si indica con $\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$

Casi particolari:

- Norma euclidea: $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$
- Norma uno: $\|x\|_1 = \sum_{i=1}^{n} |x_i|$
- Norma infinito: $\|x\|_\infty = max_{i,\dots,n} |x_i|$

Norme di matrici

1. $\|A\|_1 = max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}|$: dobbiamo calcolare la somma (in valore assoluto) degli elementi di ogni colonna di A e trovare il massimo fra di esse.
2. $\|A\|_\infty = max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{ij}|$: dobbiamo calcolare la somma (in valore assoluto) degli elementi di ogni riga di A e trovare il massimo fra di esse.
3. $\|A\|_2 = \sqrt{\rho(A^T A)}$, Il simbolo ρ denota il raggio spettrale di una matrice, che è il valore assoluto massimo degli autovalori di quella matrice. $\rho(A^T A) = max|\lambda_i|$

Norma di Frobenius: $\|A\|_F = \sqrt{\sum_{i,j=1}^{n} (a_{ij})^2}$

Norma indotta: $\|A\|_p = \max_{x \in \mathbb{R}^n} \frac{\|Ax\|_p}{\|x\|_p}$, proprietà $\|Ax\|_p \leq \|A\|_p \|x\|_p$

Due norme sono equivalenti se $c_2 \|x\|_{\tilde{p}} \leq \|x\|_p \leq c_1 \|x\|_{\tilde{p}}$ con $p \neq \tilde{p}$

Matrice di Hilbert: $a_{ij} = \frac{1}{i+j-1}$