Un aiuto all'esame di calcolo scientifico

Mattias Cibien e Daniele Canciani

17/09/2011

Indice

Sistemi Lineari

1 Algoritmo di eliminazione di Gauss

L'algoritmo di eliminazione di Gauss si basa su un procedimento iterativo che fattorizza la matrice A^{n*n} di un sistema $Ax = b$ in una matrice triangolare superiore U, e una matrice triangolare inferiore L tali che $A = LU$. La risoluzione del sistema lineare, una volta applicato l'algoritmo diventa

$$\begin{cases} Lz = b \\ Ux = z \end{cases} .$$

Consideriamo la matrice seguente:

$$G_k = \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & -\frac{v_{k+1}}{v_k} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -\frac{v_n}{v_k} & 0 & \dots & 1 \end{pmatrix}.$$

Queste matrici si chiamano **matrici elementari di Gauss** e hanno la particolarità di annullare tutte le componenti di un vettore v dalla $(k+1)$-esima in poi, lasciando inalterate le componenti precedenti. Esse sono anche facilmente invertibili (basta cambiare di segno tutti gli elementi tranne la diagonale). L'algoritmo di eliminazione di Gauss agisce in questo modo: azzera ad ogni passo gli elementi inferiori di una colonna della matrice A, rendendola infine una matrice triangolare superiore. Il procedimento di basa quindi su questo schema iterativo, per $k = 1, \dots, n-1$:

$$A^0 = A$$

$$A^k = G_k A^{k-1}$$

dove la matrice G_k è ottenuta dalla k-esima colonna di A^{k-1}, cioè:

$$G_k = \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & -l_{k+1,k} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -l_{n,k} & 0 & \dots & 1 \end{pmatrix}$$

dove $i_{ik} = \frac{a_{i,k}^{k-1}}{a_{k,k}^{k-1}}$ con $i = k+1, \dots, n$. Ad ogni passo k, il prodotto per la matrice elementare produce una matrice in cui gli elementi inferiori della colonna k

sono nulli. Siccome le matrici elementari sono, fino alla riga e alla colonna k, equivalenti all'identità la moltiplicazione riga per colonna, vengono alterati solo gli elementi dalla $(k+1)$-esima riga e dalla $(k+1)$-esima colonna, non toccando altri elementi.

Alla fine del processo A^{n-1}risulta triangolare superiore e sarà quindi la matrice U finale. Per quanto riguarda L invece, se chiamiamo $G = G_{n-1} \cdots G_2 G_1$ risulta che

$$U = A^{n-1} = (G_{n-1} \cdots G_2 G_1)A = GA$$
$$A = G^{-1}U$$
$$L = G^{-1}.$$

Questo algoritmo funziona solo se tutti gli elementi pivotali ($a_{k,k}^{k-1}$) non sono nulli, questo vale a dire che tutte le sotto-matrici principali di testa sono non singolari. Nel caso in cui almeno uno di questi sia nullo è necessario ricorrere alle tecniche di pivoting. La tecnica del **pivoting parziale** consiste, ad ogni passo k nello scambiare due righe della matrice A^{k-1} in modo da portare l'elemento massimo di quella colonna dall'elemento pivotale fino alla fine alla posizione di quest'ultimo utilizzando delle matrici di permutazione P_k (si ottiene così una fattorizzazione di tipo $PA = LU$). Essa permette di aumentare la stabilità dell'algoritmo. La risoluzione di un sistema lineare diventa: quindi

$$\begin{cases} Lz = Pb \\ Ux = z \end{cases}$$

dove $P = \prod_{i=n}^{i} P_i$, $\tilde{L} = P_1^{-1} G_1^{-1} \cdots P_{n-1}^{-1} G_{n-1}^{-1} L = P\tilde{L}$.

Esiste anche la tecnica del **pivoting totale** che, oltre a scambiare le righe, scambia anche le colonne (sempre partendo dall'elemento pivotale), aumentando ulteriormente la stabilità.

Il determinante della matrice A poi può essere calcolato calcolandodet (U), ricordando che il determinante di una matrice triangolare è uguale al prodotto degli elementi della diagonale (per questo motivo det $(L) = 1$), a meno del segno che è determinato dal determinante della matrice di permutazione.

2 Sistemi sovra-determinati

Molto spesso si ha a che fare con dei sistemi con un numero di incognite maggiore del numero di equazioni. In questo caso si dice che il sistema è sovradeterminato. Questo fatto fa si che non sempre esiste una soluzione al problema, dobbiamo quindi accontentarci di un'approssimazione.

2.1 Approssimazione ai minimi quadrati

Sia $A \in \mathbb{R}^{m*n}$ una matrice sovradimensionata (grassa), quindi con $m > n$ e $b \in \mathbb{R}^m$ un vettore colonna alto quanto le colonne di A. Il sistema $Ax = b$ non ha in generale una soluzione esatta, ma in alcuni casi sì. Geometricamente

possiamo vedere i vettori colonna $a_1, a_2, \ldots, a_n$ e il vettore colonna b come elementi dello spazio vettoriale $\mathbb{R}^m$. Inoltre il sistema $Ax = b$ può essere scritto anche in questa forma: ,

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$x_1 \cdot \begin{pmatrix} a_{11} \\ \vdots \\ a_{m1} \end{pmatrix} + \cdots + x_n \cdot \begin{pmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$

Si deduce quindi che una soluzione esatta del sistema esiste se e solo se b può essere espresso come combinazione lineare dei vettori colonna di A. Questo significa che una soluzione esatta esiste se e solo se $b \in span\{a_1, \ldots, a_n\}$ quindi se b appartiene al **sottospazio generato** dai vettori colonna di A.

Nel caso in cui non esista una soluzione esatta la migliore approssimazione possibile è quella che minimizza il **residuo,** ovvero il valore $z \in \mathbb{R}^n$ tale che:

$$\|Az - b\|_2^2 \leq \|Ax - b\|_2^2 \qquad \forall x \in \mathbb{R}^n$$

La soluzione approssimata in questo modo si chiama **approssimazione nel senso dei minimi quadrati**. Si sceglie di minimizzare il quadrato del residuo, che è un problema equivalente, perché nel calcolo della norma due è coinvolta l'estrazione di una radice quadrata, che in questo modo viene evitata. Geometricamente il residuo corrisponder alla lunghezza del vettore differenza $Az - b$, cioè la distanza tra b e il vettore Az che giace sul piano. La soluzione cercata è quella che fa in modo che la distanza tra Az e b sia minima, e geometricamente si intuisce che ciò accada quando $Az - b$ è **ortogonale** al piano.

Due vettori v e v'sono ortogonali se il loro prodotto scalare è uguale a zero. Un vettore è ortogonale ad un sottospazio se è ortogonale alla base di quello spazio. Da queste considerazioni si nota che il residuo $\|Ax - b\|_2$ è il vettore per il quale $r(x) = Ax - b$ è ortogonale ai vettori colonna di A, cioè:

$$a_j^T r(x) = 0 \quad j = 1, \ldots, n$$

$$A^T r(x) = 0$$

$$A^T Ax = A^T b \qquad sosituendo\, r(x) = Ax - b$$

Questo insieme di condizioni è detto **sistema di equazioni normali** e dalla sua soluzione si ricava la soluzione del problema.

Possiamo quindi risolvere il sistema con i seguenti passi:

1. Calcolare $M = A^T A$;
2. Calcolare $d = A^T b$;
3. Risolvere $Mx = d$.

La matrice M risulta simmetrica e quindi il sistema può essere risolto con l'algoritmo di Cholesky, nel caso in cui risulti definita positiva. Con degli errori di arrotondamento però può succedere che la matrice M può non essere definita positiva o addirittura singolare. Il condizionamento del problema è infatti $cond\,(M) = cond^2\,(A)$. Il metodo di risoluzione non è molto stabile, per questo motivo si introducono quindi i metodi di fattorizzazione **QR** e **SVD**.

Zeri di funzione

3 Metodo di bisezione

Se una funzione continua cambia segno agli estremi di un intervallo, questo intervallo sicuramente conterrà uno zero. In altre parole: se f continua in un intevallo $[a, b]$ tale che $f\,(a)\,f\,(b) < 0$ esise $c \in [a, b]$ tale che $f\,(c) = 0$.

Il metodo di bisezione funziona quindi dimezzando un tale intervallo un numero di volte sufficiente ad isolare la radice che si sta cercando. Più precisamente, si stabilisce un intervallo iniziale $I^{(0)} = \left[a^{(0)}, b^{(0)}\right]$tale che $f\left(a^{(0)}\right) f\left(b^{(0)}\right) < 0$ e ad ogni iterazioni, per ogni k:

- Si divide a metà l'intervallo: $x^{(k)} = \frac{a^{(k)}+b^{(k)}}{2}$;
- Si trova l'ntervallo per l'iterazione successiva ponendo:

$$I^{(k+1)} = \begin{cases} \left[a^{(k)}, x^{(k)}\right] & se \quad f\left(a^{(k)}\right) f\left(x^{(k)}\right) < 0 \\ \left[x^{(k)}, b^{(k)}\right] & se \quad f\left(x^{(k)}\right) f\left(b^{(k)}\right) < 0 \end{cases}$$

Idealmente si arresterebbe quando in un punto k in cui la funzione si annulla, ma dati gli errori di approssimazione potrebbe non accadere mai. Bisogna quindi trovare dei criteri di arresto. Cerchiamo quindi di quantificare l'errore che si commette arrestando a k.

Ad ogni iterazione l'intervallo viene dimezzato:

$$\left|I^{(k)}\right| = \left(\frac{1}{2}\right)^k \left|I^{(0)}\right|.$$

Sappiamo che la soluzione esatta α è contenuta sempre nell'intervallo $I^{(k)}$e quindi l'errore commesso al passo k, cioè la distanza tra $x^{(k)}$e αè sicuramente minore della metà dell'ampiezza dell'intervallo, quindi:

$$\left|e^{(k)}\right| = \left|x^{(k)} - \alpha\right| < \frac{1}{2}\left|I^{(k)}\right| = \left(\frac{1}{2}\right)^k (b - a).$$

Volendo garantire un errore minore di una certa tolleranza ε, è quindi sufficiente terminare al passo in cui $\left|I^{(k)}\right| < 2\varepsilon$. Dato che si dimezza l'ampiezza ad

ogni passo e la soluzione è certamente inclusa nell'intervallo, gli estremi dell'intervallo tendono sicuramente ad α. Per soddisfare un critero di arresto abbiamo il seguente numero di iterazioni k_{min}:

$$\left(\frac{1}{2}\right)^{k_{min}+1}(b-a) < \varepsilon$$

$$k_{min} > \log_2\left(\frac{b-a}{\varepsilon}\right) - 1.$$

Questo algoritmo però non garantisce una diminuzione dell'errore in modo uniforme.

4 Metodi di iterazione funzionale

Questi metodi costruiscono la soluzione iterativamente a partire da un'approssimazione iniziale. Consideriamo una successione definita da una funzione g nel seguente modo, partendo da un valore iniziale x_0:

$$x_{k+1} = g(x_k).$$

Se la funzione è continua e la successione è convergente, essa converge ad un valore a tale che $g(a) = a$. Questo valore si chiama **punto fisso** della funzione g. Con queste nozioni base possiamo trasfomare il nostro problema di trovare le radici di una funzione f in un problema di punto fisso, data una opportuna funzione g. Bisogna quindi risolvere l'equazione $g(x) = x$. I metodi di iterazione funzionale si basano sulla ricerca iterativa del punto fisso di una funzione, scelta in base a quella di partenza, partendo da un'approssimazione iniziale x_0. Si differenziano per la scelta della funzione g.

Bisogna scegliere bene la funzione in quanto non è sempre possibile che la successione converga.

La differenza tra il risultato dell'iterazione $k+1$ e il punto fisso a, ovvero l'errore e_{k+1}, ad ogni iterazione è uguale a:

$$e_{k+z} = x_{k+1} - a = g(x_k) - g(a)$$

Applicando il teorema del valor medio di Lagrange, sappiamo che esiste un valore $c_k \in [x_k, a]$ tale che:

$$e_{k+1} = g'(c_k)(x_k - a) = g'(c_k)\, e_k.$$

L'errore quindi varia rispetto all'iterazione precedente in base al comportamento della derivata di g in un intorno di a. In particolare, perchè la successione converga è necessario che $|g'(a)| < 1$. Se la funzione è derivabile con continuità basta assicurare che questa condizione valga per a per sapere che vale in un suo intorno quindi:

- Se $|g'(a)| > 1$ la successione diverge;

- Se $|g'(a)| = 1$ è possibile che converga (se esiste di a in cui la derivata di g è minore di uno);
- Se $0 < g'(a) < 1$ la successione è localmente convergente monotona;
- Se $-1 < g'(a) < 0$ la succession è localmente convergente in modo alternato;
- Se $g'(a) = 0$ non si può dire nulla sul tipo di convergenza ma sappiamo che la successione sarà comunque localmente convergente.

La velocità di convergenza si analizza osservando questo limite:

$$\lim_{k\to\infty} \frac{|x_{k+1} - a|}{|x_k - a|} = l$$

per una successione $\{x_k\}_{k\geq 0}$:

- Se $l = 1$ la convergenza si dice **sublineare** (dopo un certo numero di iterazioni l'errore smetterà di diminuire);
- Se $0 < l < 1$ la convergenza di dice **lineare** (il rapporto tende ad un valore finito comproeso tra zero e uno);
- Se $l = 0$ la convergenza si dice **superlineare** e in questo caso di definisce **ordine di convergenza** il più piccolo numero naturale p tale che:

$$\lim_{k\to\infty} \frac{|x_{k+1} - a|}{|x_k - a|^p} = L > 0$$

Nel caso delle successioni generate da un metodo di iterazione funzionale, si verifica che $l = |g'(a)|$, da cui si deduce che l'ordine di convergenza della successione è il più piccolo p tale che $g^{(p)}(a) \neq 0$ (se la funzione è derivabile con continuità fino all'ordine p.

4.1 Metodo di Newton

Il metodo di Newton è il metodo più usato dei metodi di iterazione funzionali. Ad ogni passo k si scegle x_{k+1} pari all'intersezione con le ascisse della tangente alla curva f nel punto $(x_k, f(x_k))$.

La retta tangente nel punto $(x_k, f(x_k))$ presenta la seguente equazione:

$$y(x) = f(x_k) + f'(x_k)(x - x_k)$$

Quindi cercando x_{k+1} come punto di intersezione di questa retta con le ascisse, ovvero $y(x_{k+1}) = 0$, troviamo lo schema generale del metodo di Newton:

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

Possiamo vedere il metodo di Newton come un metodo di iterazione funzionale considerando come funzione di iterazione:

$$g(x) = x - \frac{f(x)}{f'(x)}$$

gli zeri di questa funzione f infatti sono i punti fissi della funzione g.

4.1.1 Convergenza per radici semplici

Consideriamo una **radice semplice**. In questo caso abbiamo che $f(a) = 0$ e $f'(a) \neq 0$, possiamo quindi esprimere la derivate di g così:

$$g'(x) = \frac{f(x) f''(x)}{f'(x)^2}$$

che valutata in una radice di a di f vale zero. Avendo $|g'(a)| = 0$ possiamo conclude che il metodo di Newton converge sempre localmente in modo superlineare. Per calcolare l'ordine di convergenza dobbiamo trovare il più piccolo p per cui la derivata di g di ordine p valutata in a risulta non nulla. Calcoliamo quindi la derivata seconda di g:

$$g''(x) = \frac{f''(x)}{f'(x)}$$

da cui si conslude che la convergenza dipende dal comportmamento della derivata seconda di f:

- Se $f''(a) \neq 0$ la convergenza è quadratica;
- Altrimenti è maggiore di quadratica (per esempio l'orine è 3 se $g'''(a) \neq 0$).

Il metodo di Newton converge localmente in modo superlineare alemno quadratico.

Teorema Sia f una funzione e I_a un intorno destro o sinistro di una radice a di f tale che $f \in C^2(I_a)$. Prendendo un valore inziale $x_0 \in I_a$, la convergenza del metodo di Newton è garantita essere **monotona** se valgono entrambe le condizioni:

- $f'(x_0) \neq 0$
- $f(x_0) f''(x_0) > 0$

4.1.2 Convergenza per radici multiple

Consideriamo il caso di una radice con molteplicità $\mu > 1$. Ciò significa che tutte le derivate fino all'ordine $\mu - 1$ valutate in a sono nulle, mentre $f^{(\mu)} \neq 0$. In questo caso possiamo somporre la funzione f in questo modo:

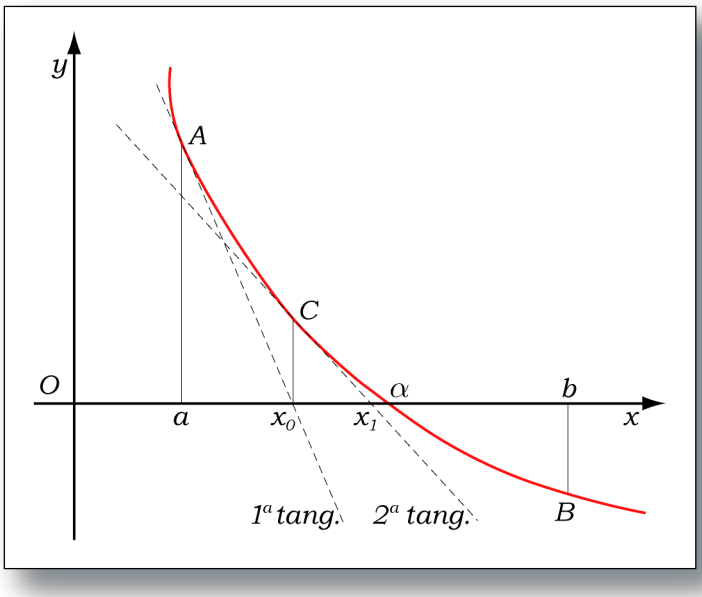
$$f(x) = (x - a)^{\mu} h(x)$$

dove $h(x)$ è una funzione continua per cui $h(a) \neq 0$. Si può esprimere quindi la derivata di g in funzione della molteplicità della radice:

$$g'(a) = 1 - \frac{1}{\mu}.$$

Da questoi deduciamo che il metodo di Newton, nel caso di radici multiple, ha convergenza solo **lineare**. Nel caso di una radice doppia ad esempio il fattore di riduzione è $\frac{1}{2}$.

Questo metodo è molto veloce nel caso di radici semplici quindi. Nel caso di radici con molteplicità elevata o nel caso in cui la derivata di una funzione non sia facilemente calcolabile è preferibile ricorrere ad altri metodi perchè ad ogni passo di calcolo è richiesto il calcolo di una derivata e della funzione stessa. oltrettuto può non essere disponibile l'espressione analitica della derivata ed è quindi necessario approssimare quest'ultima, aggiungendo costo alla computazione.



4.2 Metodo a pendenza costante

Il metodo a pendenza costante è una semplificazione del metodo di Newton che invece di calcolare il valore di $f'(x)$ ad ogni passo, utilizza un coefficiente angolare costante.

La funzione di iterazione di questo metodo si presenta quindi così:

$$g(x) = x - \frac{f(x)}{m}$$

dove m è, appunto, una costante. Per analizzare la convergenza di questo metodo sfruttiamo quanto utilizzato precendetemente, quindi studiamo la derivata di g:

$$g'(x) = 1 - \frac{f'(x)}{m}$$

perché il metodo converga è necessario che $|g'(a)| < 1$, e quindi:

$$\left|1 - \frac{f'(a)}{m}\right| < 1$$

Questa condizione è difficile da assicurare, in quanto non conoscendo a è impossibile conoscere $f'(a)$. Si può vedere, comunque, che la convergenza è assicurata da tutti gli x in un intorno di I_a tali che:

$$f'(x) \neq 0,\ mf'(x) > 0,\ |m| > \max_{x \in I_a} |f'(x)|$$

Tenendo conto di questo criterio basta porre $m = f'(x_o)$, scegliendo $x_0$tale che $|f'(x_0)| > \frac{1}{2} \max_{x \in I_a} |f'(x)|$.

Questo metodo, per quanto semplice da implementare, è di difficile uso visto il rischio che non converga nel caso di una scelta sbagliata di m e/o di x_0.

4.3 Altri metodi

Il metodo di Netwon richiede, come già accennato il calcolo della derivata ad ogni iterazione. Calcolo non sempre efficiente e in certi casi impossibile.

Dobbiamo quindi ricorrere ad altri metodi che aggirano questo problema.

Uno fra questi è il metodo delle secanti, che approssima la derivata con il rapporto incrementale dei valori delle due iterate precedenti, il procedimenti iterativo risulterà quindi:

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

$$con\, x_0\, e\, x_1\, dati.$$

In Matlab è implementato l'algoritmo di Brent, un algoritmo ibrido. Questi algoritmi cercano di prendere i pregi dei vari metodi, e usualmente eseguono nel seguente modo:

- Inizialmente si usa il metodo di bisezione per trovare un approssimazione grossonala dello zero;
- Successivamente si continua con metodo meno stabili (ricordiamo che bisezione converge sempre) ma più veloci come il metodo delle secanti.

5 Interpolazione

L'interpolazione ci permette di approssimare funzioni complicate in funzioni più semplici oppure, dati dei campioni di una funzioni trovare dei valori apporssimati della funzione non presenti nei campioni.

Dati m punti (t_i, y_i) detti **nodi di interpolazione** si vuole trovare una funzione $\varphi(t)$ tale che:

$$\varphi(t_i) = y_i$$

In pratica si vuole ricavare una **funzione di interpolazione** che passi per quegli stessi punti.

5.1 Interpolazione polinomiale

La prima possibilità di scelta di una funzione di iterpolazione è quella di scegliere un polinomio funzione φ. Questo polinomio, se consideriamo $m+1$ punti distinti, possiamo dimostrare che esiste ed è unico di grado m.

L'approccio più conveniente per costruire questo polinomio consiste nel rappresentarlo come combinazione lineare di **funzioni base** $\phi_0(t), \ldots, \phi_m(t)$, ovvero:

$$\varphi(t) = \sum_{j=0}^{m} x_j \phi_j(t)$$

Per interpolare i punti, inoltre vogliamo che $\varphi(t_i) = y_i$. I coefficienti della combinazione lineare sono quindi le incognite di un sistema lineare quadrato di dimensione $m+1$. Risulta quindi cruciale scegliere delle funzioni base che siano calcolabili in maniera efficiente e con precisione, ricordando comunque che il polinomio interpolante è unico e la scelta delle funzioni base determina solo una diversa rappresentazione dello stesso.

5.1.1 Base dei monomi

Come base più semplice prendiamo i monomi $\phi_i(t) = t^i$. Il polinomio interpolante è quindi di forma:

$$p_m(t) = x_0 + x_1 t + \ldots + x_m t^m$$

da cui si vede subito che la matrice risultante, detta **matrice di Vandermonde**, ha struttura:

$$A = \begin{pmatrix} 1 & t_0 & t_0^2 & \cdots & t_0^m \\ 1 & t_1 & t_1^2 & \cdots & t_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^m \end{pmatrix}$$

Se i nodi risultano distinti, una matrice di questo tipo è non singolare, quindi il sistema è risolvibile. Risulta però fortemente malcondizionata soprattutto per un numero di nodi elevato. Il costo computazionale è oltrtutto $O(n^3)$ usando l'algoritmo di Gauss. Non risulta quindi molto efficiente.

5.1.2 Rappresentazione di Lagrange

Nella rappresentazione di Lagrange le funzioni base sono scelte affinchè ogni $l_i(t)$ valga 0 in tutti i punti $t_j \neq t_i$, mentre $l_i(t_i) = 1$. Il polinomio interpolante si può quindi esprimere con:

$$\varphi(t) = \sum_{i=0}^{m} l_i(t)\, y_i$$

I coefficenti inconiti sono somplicemente i dati di y_i quind non è necessario calcolare nulla.

La funzione base di Lagrange è:

$$l_i(t) = \prod_{j=0, j\neq i}^{m} \frac{t - t_i}{t_i - t_j}$$

Anche se il calcolo dei coefficienti è immediato, la valutazione del polinomio fuori dai punti campionati può risultare inefficiente in quanto non si può applicare l'algoritmo di Horner.

5.1.3 Rappresentazione di Newton

La funzione base di Newton sopperisce ai problemi che presenta la rappresentazione di Lagrange e le sue funzioni base sono:

$$\phi_0(t) = 1$$

$$\phi_i(t) = \prod_{j=0}^{i-1} (t - t_j)$$

Il polinomio risultante ha quindi forma:

$$\varphi(t) = x_0 + x_1(t - t_0) + x_2(t - t_0)(t - t_1) + \ldots + x_m \prod_{j=0}^{m-1} (t - t_j)$$

I termini successivi all'i-esimo, quando il polinomio viene valutato in t_i, risultano quindi nulli. Da questa osservazione deriva l'importante caratteristica del polinomio di Newton: il sistema da risolvere è quindi composto da una matrice triangolare inferioe. Ad esempio, con tre punti:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & (t_1 - t_0) & 0 \\ 1 & (t_2 - t_0) & (t_2 - t_0)(t_2 - t_1) \end{pmatrix}$$

I coefficenti x_i possono quindi essere trovati risolvendo il sistema con l'algoritmo di sostituzione in avanti (costo $O(m^2)$). Inoltre si può valutare il polinomio usando una variante dell'algortimo di Horner. Per quanto riguarda

l'accuratezza però, date le molte sottrazioni, il buon condizionamento può risentire di un ordine sbagliato dei nodi e spesso l'ordiamento crescente non è la scelta migliore, oltretutto si rischia di andare incontro a overflow e underflow. Per rimediare a queste pecché esiste un algoritmo alternativo chaiamto **differenze divise**.

Questa formulazione ha un altro vantaggio: se devo aggiungere un nodo di interpolazione non devo ricalcolare tutti i coefficienti in quanto mi basta calcolare il valore dell'ultimo coefficiente; questa proprietà non vale per le altre formulazioni.

5.1.4 Errore e convergenza del polinomio interpolante

Sia f la funzione da interpolare. Dati $m+1$ punti compresi in un intervallo$[a,b]$, $p_m(t)$ polinomio interpolante di f e $f \in C^{m+1}[a,b]$ si può domostrare che esiste un punto $c \in [a,b]$ tale che:

$$e_m(t) = f(t) - p_m(t) = \frac{f^{m+1}(c)}{(m+1)!}(t-t_0)\cdots(t-t_m)$$

L'errore dipendi quindi gran parte dal valore dei fattori $(t-t_i)$. Questa formula però non ci consente di stimare l'errore in modo pratico perché non conosciamo la funzione f e tanto meno le sue derivate. Possiamo però estrarre da questa formula una maggiorazione dell'errore. Supponiamo di conoscere una maggiorazione della derivata di ordine $m+1$, chiamta D_{m+1} e di maggiorare ogni fattore $(t-t_i)$ con la lunghezza dell'intervallo $(b-a)$. Risulta quindi che:

$$e_m(t) \leq \frac{D_{m+1}}{(m+1)!}|b-a|^{m+1}$$

Possiamo poi raffinare questa maggiorazione se gli estremi dell'intervallo a e b corrispondono a t_0e t_m, nel qual caso, definendo h come la lunghezza dell'intervalllo $[t_i, t_{i+1}]$ più lungo, si può domostrare che:

$$e_m(t) \leq \frac{D_{m+1}}{4(m+1)}h^{m+1}$$

Il problema è che non sappiamo dice con certezza se la precisione cresce al crescere di m. Il limite con $m \to \infty$dipende da come si comportano le derivate di f. Al crescere di m, comunque, si può dimostrare che l'errore converge a zero se esitiono c e k tali che:

$$D_{m+1} \leq ck^{m+1}$$

In caso contrario l'errore può anche divergere.

5.2 Interpolazione polinomiale a tratti

Invece che interpolare i nodi con un unico polinomio di grado elevato si possono utilizzare più polinomi di grado basso raccordati con continuità tra di loro. In altre parole, il polinomio interpolante a tratti $P_{n,m}$è composto da m polinomi di grado n, uno per ogni intervallo tra gli $m+1$ nodi di interpolazione.

5.2.1 Interpolazione lineare a tratti

Il caso più semplice è il polinomio $P_{1,m}$, che interpola gli $m+1$ punti congiungendo due punti consecutivi con polinomi di primo grado (ovvero segmenti).

Sebbene questo sistema sembri grossolano al crescere del numero di nodi può essere molto più preciso di un polinomio interpolante.

L'errore totale di questo polinomio è maggiorato del massimo errore dei singoli intervalli. All'interno degli intervalli, i punti sono praticamente interpolati da un polinomio interpolante di primo grado. Da quanto abbiamo visto, l'errore di un polinomio di grado n è maggiorato in questo modo

$$e_n(t) \leq \frac{D_{n+1}}{4(mn+1)} h^{n+1}$$

Essendo $n = 1$, otteniamo che l'errore del polinomio lineare a tratti è maggiorato in questo modo:

$$e_{1,m} \leq \frac{D_2}{8} h^2$$

All'aumentare del numero di nodi, quindi, anche con una distribuzione equispaziata, l'errore tende a zero.

5.2.2 Polinomio a tratti di grado 2

Il secondo polinomio a tratti è quello di grado 2, $P_{2,m}$che interpola gli $m+1$punti congiungendo i due nodi con delle parabole.

In questo caso l'errore è maggiorato con:

$$e_{2,m} \leq \frac{D_3}{6} h^3$$

5.2.3 Polinomio di Hermite a tratti

L'ultimo polinomio a tratti che studieremo è quello di Hermite, $H_{3,m}$che risulta una funzione $C^1[a,\, b]$ imponendo nei punti di raccordo oltre che alla continuità anche la continuità della derivata. I parametri che descrivono tale polinomio sono $2m+2$ e l'errore è maggiorato con:

$$e_{3,m} \leq \frac{D_4}{16 * 4!} h^4$$

5.3 Funzioni Spline

Si definisce Spline di grado n relativa alla suddivisione Δ_m la funzione $S_{n,m} : [a,b] \to R$ tale che:

- $S_{n,m} \in C^{n-1}([a,b])$: cioè la derivata di ordine n-1 è continua;
- $S_{n,m} \in \prod_n$: cioè appartiene allo spazio dei polinomi di grado n.

Una Spline di grado n è quindi un polinomio di grado n con le derivate continue fino alla $(n-1)-esima$. Da questo si deduce che la Spline è definita da $m+n$ parametri.

D'ora in poi prenderemo in esame le Spline cubiche, $S_{3,m}$ che quindi sono definite da m+3 parametri, di cui:

- $m+1$sono i nodi di interpolazione;
- 2 parametri rimangono liberi, e danno origine a diverse famiglie di spline:
  - naturali: $S_3^{''}(t_0) = S_3^{''}(t_m) = 0$;
  - periodiche: $S_3^{'}(t_0) = S_3^{'}(t_m)$ $e\, S_3^{''}(t_0) = S_3^{''}(t_m)$;
  - vincolate: $S_3^{''}(t_0) = d_{y_0}$ $è\, S_3^{''}(t_m) = d_{y_m}$;
  - not-a-knot: $S_3^{'''}(t)$ $continua\, in\, t_1\, e\, t_{m-1}$.

La scelta della famiglia dipende dal tipo di proprietà che si vogliono garantire all'interpolante.

Teorema sulla convergenza Sia $f \in C^4([a,b])$ e sia $\triangle_m$una suddivisione di $[a,b]$ tale che $\frac{h}{|h_j|} \leq K$, $i = 0, ..., m-1$ con $h = max_{i=0,...,m-1}|h_i|$. Allora:

$$max_{a,\leq t\leq b}|f^k(t) - S_3^k(t)| \leq C_k * K * h^{4-k} * D_4$$

$$k = 0, 1, 2, 3\ con\, C_k > 0\ opportuna\, costante.$$