Appunti riassuntivi di Calcolo Numerico
A.A. 2010/2011

Luke Bonham
dadasignificanulla [at] gmail [dot] com

Indice

1 Premessa

Questi appunti vogliono solo essere una sintesi degli argomenti principali del corso: in alcun modo devono sostituire le fonti primarie di studio, bensì **vanno usati come un piccolo e pratico compendio**, qualora si abbia la necessità.

L B

2 Programma del corso

Il programma del corso su cui si basano questi appunti si trova al seguente indirizzo:
`http://www.dma.unina.it/~murli/didattica/2010_2011/cn/programmaCN2010-11.pdf`

3 Sistemi aritmetici a precisione finita

3.1 Errore assoluto

Dati un numero x ed una sua approssimazione $\tilde{x}$, si dice errore assoluto in $\tilde{x}$ la quantità

$$E = |x - \tilde{x}| \ .$$

Proprietà dell'errore assoluto
Se due numeri x e $\tilde{x}$ hanno la stessa parte intera e le stesse prime m cifre decimali, allora risulta

$$E = |x - \tilde{x}| < 10^{-m}$$

L'errore assoluto non tiene conto della grandezza del valore da approssimare. Un modo per fare questo è rapportare l'errore assoluto a $|x|$, ovvero scalare l'errore considerando $|x|$ come unità di misura. A tal fine si introduce il concetto di *errore relativo*.

3.2 Errore relativo

Dati un numero x ed una sua approssimazione $\tilde{x}$, si dice errore relativo in $\tilde{x}$, la quantità

$$E' = \frac{|x - \tilde{x}|}{|x|} \quad (x \neq 0) \ .$$

Proprietà dell'errore relativo
Se due numeri x e $\tilde{x}$ hanno le stesse prime m cifre significative, allora risulta

$$E' = \frac{|x - \tilde{x}|}{|x|} < 10^{-m+1}$$

Pertanto l'errore relativo misura la distanza tra due numeri in termini di cifre significative corrette, piuttosto che di cifre decimali corrette. Si noti che questa proprietà *non* implica che x e $\tilde{x}$ abbiano le stesse prime m cifre significative, e non individua il massimo valore di m per cui vale tale relazione, cioè non può essere invertita.

3.3 Il sistema aritmetico floating-point

In $F(\beta, t, emin, emax)$,

- $rmin = 0.1 \times \beta^{emin}$
- $rmax = (1 - \beta^{-t}) \times \beta^{emax}$
- $\epsilon_{mac} = u^1 = \begin{cases} 0.5\ \beta^{1-t} & \text{per l'arrotondamento} \\ \beta^{1-t} & \text{per il troncamento} \end{cases}$
- $\epsilon_x = |x| \times u$
- numeri macchina presenti in F $= 2(\beta - 1)\beta^{t-1}(emax - emin + 1) + 1$

Per ogni numero reale $x = f \times \beta^e$ si verifica una ed una sola delle tre situazioni:

1. $x \in F$ ed è quindi esattamente rappresentabile;
2. x non è rappresentabile in F;
3. x è rappresentabile, ma non esattamente, in F.

[1] Massima accuratezza relativa.

Il caso 2 si presenta se $|x| < rmin$, ed allora si dice che si è verificato un *underflow*, oppure se $|x| > rmax$, ed allora si dice che si è verificato un *overflow* ($rmin$ e $rmax$ sono chiamati rispettivamente *soglia di underflow* e *soglia di overflow*); tale situazione in pratica si ha quando $e < emin$ oppure $e > emax$.

Il risultato $\tilde{r}$ di una operazione f.p. dipende dall'algoritmo utilizzato dal sistema aritmetico f.p., cioè, in definitiva, dall'organizzazione e dal progetto (a livello hardware) dell'unità aritmetica. Si considera ottimale, da un punto di vista dinamico, un sistema aritmetico f.p. per il quale si abbia

$$\tilde{r} = x\tilde{\#}y = fl(r) = fl(x\#y) \ .$$

Ciò implica che il risultato $\tilde{r}$ dell'operazione f.p., che è un numero macchina, è esattamente la rappresentazione f.p.n. a precisione finita del risultato r della corrispondente operazione in $\mathfrak{R}$. Un tale sistema aritmetico garantisce dunque che il risultato di qualsiasi operazione f.p. differisca dal risultato dell'operazione in $\mathfrak{R}$ corrispondente (il risultato esatto), di una quantità che è il solo errore di rappresentazione di r, e viene perciò detto *sistema a massima accuratezza dinamica*.
Per ogni operazione f.p., l'esecuzione consta essenzialmente di quattro fasi:

1. confronto tra gli esponenti della rappresentazione f.p.n. degli operandi per individuare quello con l'esponente più piccolo;
2. shift delle cifre della mantissa di tale operando in modo che il relativo esponente risulti uguale a quello dell'altro operando;
3. operazione delle mantisse degli operandi;
4. normalizzazione ed arrotondamento (o troncamento) del risultato.

3.4 Errore di roundoff

Considerato un sistema aritmetico f.p. a precisione finita, con $F(\beta, t, emin, emax)$, sia x un numero reale appartenente all'insieme di rappresentabilità di F e sia $fl(x)$ la sua rappresentazione in F. Si dice *errore assoluto di roundoff* il numero

$$|fl(x) - x| \ ,$$

e si dice *errore relativo di roundoff* il numero

$$\frac{|fl(x) - x|}{|x|}$$

L'errore di roundoff si genera sia quando un numero reale, dato nella sua rappresentazione decimale, viene rappresentato in F, cioè nel passaggio dalla rappresentazione esterna a quella interna (*errore di roundoff di rappresentazione*), sia nell'esecuzione di ogni operazione f.p. (*errore di roundoff delle operazioni f.p.*).

3.5 Epsilon macchina

In un sistema aritmetico f.p. con $F(\beta, t, emin, emax)$, si dice epsilon macchina il più piccolo numero $\epsilon \in F$ tale che

$$1 \oplus \epsilon = fl(1 + \epsilon) > 1$$

L'epsilon macchina coincide con la massima accuratezza relativa u.

Nota la base di numerazione, conoscere l'epsilon macchina equivale a conoscere la precisione del sistema aritmetico.

3.6 Massima accuratezza relativa

Si dice massima accuratezza relativa di un sistema aritmetico f.p. a precisione finita con $F(\beta, t, emin, emax)$ il massimo errore che si commette nel rappresentare un numero x nel sistema F.

3.7 Criterio di arresto naturale

Si dice *criterio di arresto* di un algoritmo basato su un processo iterativo l'insieme di condizioni che, se verificate, determinano l'arresto del processo iterativo.

In un processo iterativo basato su una formula ricorrente del tipo

$$S_{n+1} = S_n + a_n \ ,$$

il criterio di arresto naturale è

$$|a_n| < \epsilon_{S_n} \ ,$$

dove

$$\epsilon_{S_n} = |S_n| \times \mu \quad \text{(epsilon macchina relativo a } S_n\text{)}$$

3.8 Buon condizionamento di un problema

Un problema si dice *ben condizionato* se l'errore relativo (assoluto) nella soluzione ha al più lo stesso ordine di grandezza dell'errore relativo (assoluto) nei dati.

Pertanto, un problema per il quale l'errore relativo (assoluto) nella soluzione ha ordine di grandezza maggiore rispetto all'errore relativo (assoluto) nei dati si dice *mal condizionato*. Detto δ l'errore nei dati e σ l'errore corrispondente nella soluzione, e posto

$$\sigma = \mu \cdot \delta \ ,$$

μ è detto indice di condizionamento del problema; risulta quindi che:

se $\mu \leq 1$ il problema è *ben condizionato*;
se $\mu > 1$ il problema è *mal condizionato*.

3.9 Stabilità di un algoritmo

Un algoritmo si dice stabile, se l'errore di roundoff sul risultato finale è dello stesso ordine di grandezza di quello dell'errore presente nei dati iniziali.

In altri termini, un algoritmo è *instabile* se, a causa della propagazione dell'errore di roundoff, il risultato differisce sostanzialmente dalla soluzione del problema numerico.

3.10 Il fenomeno della cancellazione

La cancellazione è un fenomeno tipico dei sistemi a *precisione finita*. Esso si verifica a causa del vincolo sul numero di cifre (finito) della mantissa riservato per rappresentare un numero f.p. e può determinare la perdita di cifre significative quando si esegue una sottrazione tra numeri vicini.

Ad esempio, considerato un sistema aritmetico a precisione finita $F(10, 4, -12, 12)$, eseguendo la sottrazione f.p. dei due numeri seguenti:

$$0.1234 \times 10^3 - 0.1233 \times 10^3 = 0.0001 \times 10^3 = 0.1000 \times 10^0$$

si ottiene un numero con una sola cifra significativa. In altre parole, si perdono 3 cifre significative nella rappresentazione del risultato.

In generale se

$$f : \Re \times \Re \to \Re$$

è la funzione che alla coppia di numeri reali (x, y) associa il numero reale $z = x - y$, è noto che l'indice di condizionamento del problema è dato da

$$C(f, x, y) = \frac{|x| + |y|}{|x - y|}$$

Tale quantità cresce al diminuire della distanza tra x e y. Cioè la cancellazione è l'effetto del *mal-condizionamento* della sottrazione tra due numeri f.p. vicini.

4 Calcolo matriciale

4.1 Pivoting parziale

Il pivoting parziale è una tecnica per la minimizzazione degli errori di round-off, effettuata permutando le righe della matrice in modo tale che la diagonale principale sia formata dagli elementi più grandi, in valore assoluto. Più precisamente, viene effettuato un riordino delle equazioni scambiando la riga k-esima con una delle righe successive affinché venga portato in posizione $a_{k,k}$ l'elemento dominante della colonna k-esima.

4.2 La strategia del pivoting parziale nel metodo di Gauss

Il metodo di Gauss è instabile, poichè non controlla l'amplificazione degli errori di roundoff:

- se un moltiplicatore risulta nullo, il coefficiente dell'incognita che deve essere eliminata non verrà annullato;
- se ad un certo passo k l'elemento diagonale $a_{k,k}$ è nullo, non è possibile definire i moltiplicatori $m_{i,k}$ in quanto ci si ritroverebbe ad effettuare una divisione per zero; esito analogo è dato dal caso in cui un elemento diagonale sia molto piccolo (nel sistema f.p. considerato) rispetto agli altri elementi della matrice: più è grande il il moltiplicatore $m_{i,k}$, ovvero più è piccolo l'elemento diagonale che lo ha generato, maggiore è l'amplificazione dell'errore di roundoff presente nell'elemento $a^k_{i,j}$.

Dunque per evitare errori e controllare l'amplificazione dell'errore di round-off nell'algoritmo di Gauss si utilizza il pivoting, che riorganizza ad ogni passo k la matrice A^k in modo tale che i moltiplicatori $m_{i,k}$ siano tutti, in valore assoluto, minori o uguali ad uno.

Il pivoting parziale è meno accurato, ovvero con minore riduzione dell'errore di roundoff, rispetto al pivoting totale, ma preferito a quest'ultimo per:

- errore relativo di roundoff sulla soluzione accettabile;
- minor numero di confronti e quindi minore complessità.

4.3 Fattorizzazione LU

4.3.1 Problema 1: risoluzione di più sistemi lineari con uguale matrice dei coefficienti

Molte applicazioni richiedono la soluzione di più sistemi lineari del tipo:

$$Ax_1 = b_1,\ Ax_2 = b_2,\ ...,\ Ax_m = b_m,$$

cioè sistemi lineari aventi la stessa matrice dei coefficienti ma termini noti differenti che, ad esempio, non sono dati a priori ma dipendono dalla soluzione di uno dei sistemi precedenti (ad es. b_2 dipende in qualche modo da x_1). In tal caso applicando l'algoritmo di eliminazione di Gauss e la *back-substitution* a ciascun problema, il costo computazionale sarebbe uguale a:

$$m(T_{Gauss}(n) + T_{bs}(n)) \simeq \Theta(mn^3)\ .$$

Tale costo computazionale è invece notevolmente ridotto utilizzando la fattorizzazione LU di A. Infatti, una volta calcolati i fattori L ed U, basta risolvere le coppie di sistemi:

$$\begin{array}{ll} Ly_1 = b_1, & Ux_1 = y_1; \\ Ly_2 = b_2, & Ux_2 = y_2; \\ \vdots & \vdots \\ Ly_m = b_m, & Ux_m = y_m. \end{array}$$

In sintesi per risolvere il problema in esame basta:

- calcolare *una sola volta* la fattorizzazione LU di A, con un costo computazionale di $\Theta(\frac{n^3}{3})$;
- risolvere m coppie di sistemi triangolari (uno inferiore ed uno superiore), con un costo computazionale di $\Theta(2\frac{n^2}{2})$ per ciascuna di esse.

Di conseguenza il costo computazionale totale è

$$\Theta\left(\frac{n^3}{3} + mn^2\right)$$

Nota: nella fattorizzazione LU per matrici a banda, l'eventuale utilizzo di un pivoting distruggerebbe la struttura a banda della matrice A, nel senso che l'ampiezza di banda di U diventerebbe maggiore di quella di A, mentre nulla si può dire sulla banda di L.

4.3.2 Problema 2: calcolo dell'inversa di una matrice

Per calcolare l'inversa A^{-1} di una assegnata matrice A di dimensione n, occorre risolvere n sistemi lineari:

$$Ax_i = e_i, \quad i = 1, ...,n,$$

dove x_i è la i-ma colonna di A^{-1} e e_i è l'i-mo vettore unitario. Questo problema è analogo al *Problema 1*, pertanto, utilizzando l'approccio descritto precedentemente si ha che la complessità computazionale del calcolo dell'inversa è:

$$\Theta\left(\frac{n^3}{3}\right) + 2n \cdot \Theta\left(\frac{n^2}{2}\right) = \Theta\left(\frac{4}{3}n^3\right)$$

4.3.3 Problema 3: calcolo del determinante di una matrice

Un altro problema in cui la fattorizzazione LU risulta notevolmente vantaggiosa è il calcolo del determinante di una matrice A di dimensione n. Infatti, dalla relazione

$$A = LU$$

si ricava che:

$$det(A) = det(LU) = det(L) \cdot det(U)$$

Poiché L ed U sono matrici triangolari, il loro determinante è dato dal prodotto dei rispettivi elementi diagonali. Pertanto, poiché L ha tutti gli elementi diagonali uguali ad 1 si ha che:

$$det(A) = det(U) = u_{1,1} \cdot u_{2,2} \cdots u_{n,n}$$

In conclusione, la complessità computazionale del determinante di A, utilizzando la fattorizzazione LU, è

$$\Theta\left(\frac{n^3}{3}\right)$$

molto più vantaggiosa rispetto alla normale complessità computazionale del calcolo del determinante, che è $\Theta(n!)$.

4.3.4 Il vettore *ipiv*

Tale vettore permette lo scambio *virtuale* delle righe, rappresentando un vantaggio computazionale in quanto risparmia:

- la costruzione di una matrice di permutazione per memorizzare gli scambi;
- lo scambio fisico delle righe.

4.4 Condizionamento di sistemi lineari

La quantità

$$\mu(A) = ||A|| \cdot ||A^{-1}|| \ ,$$

è detta *indice di condizionamento relativo* del sistema di equazioni $Ax = b$.

Il condizionamento di un sistema lineare è una proprietà intrinseca della matrice dei coefficienti (e non dipende dagli errori di cui è affetto il vettore dei termini noti).

Un sistema di equazioni, quindi, è *mal condizionato* se $\mu(A) >> 1$, mentre è *ben condizionato* se $\mu(A) \simeq 1$. Si è soliti considerare mal condizionato un sistema di equazioni in cui è almeno $\mu(A) > \Theta(n)$.

Teorema 1 (del condizionamento) *Sia $||\cdot||$ una norma matriciale submoltiplicativa (cioè $||AB|| \leq ||A||\cdot||B||$) e compatibile con una norma vettoriale (cioè $||Ax|| \leq ||A||\cdot||x||$, $\forall x \neq 0$). Sia inoltre il sistema $Ax = b$ (con A non singolare) e si consideri il sistema perturbato:*

$$(A + \triangle A)(x + \triangle x) = (b + \triangle b)$$

Se

$$|| \triangle A|| < \frac{1}{||A^{-1}||} \ ,$$

posto $\mu(A) = ||A|| \cdot ||A^{-1}||$ si ha:

$$\frac{|| \triangle x||}{||x||} \leq \frac{\mu(A)}{1 - \mu(A)\frac{||\triangle A||}{||A||}} \left(\frac{|| \triangle A||}{||A||} + \frac{|| \triangle b||}{||b||} \right)$$

Esempi di matrici (molto) mal condizionate sono quelle di Hilbert e di Vandermonde, in cui l'indice di condizionamento cresce almeno esponenzialmente con la dimensione n:

$$n \to \infty \quad \mu(Hilb(x)) \to \infty, \ \mu(Vand(x)) \to \infty$$

per cui sono matrici *intrattabili* numericamente.

Esempi di matrici ben condizionate sono la matrice identità e le matrici diagonali non singolari.

Il numero massimo di cifre significative corrette nell'approssimazione della soluzione di un sistema di equazioni perturbato corrisponde al valore assoluto dell'esponente dell'ordine di grandezza del secondo membro della disequazione del teorema di condizionamento.

4.5 Band storage

Per memorizzare una generica matrice a banda A, di dimensione $n \times n$, con ampiezze di banda p e q, si utilizza un array bidimensionale di dimensione $(p + q + 1) \times n$, nel modo seguente:

$$AB(q + 1 + i - j, j) = a_{i,j}, \quad max(1, j - q) \leq i \leq min(n, j + p) \ .$$

Tale schema di memorizzazione è detto *a banda* (band storage) ed ha una complessità di spazio pari a $(p+q+1)n$. È chiaro che la memorizzazione a banda è significativamente vantaggiosa, rispetto all'usuale memorizzazione mediante un array bidimensionale di dimensione $n \times n$, se p, $q << n/2$.

4.6 Fattorizzazione di Cholesky

Matrice definita positiva

Una matrice $A \in \Re^{n \times n}$, simmetrica, si dice *definita positiva* se, per ogni $x \in \Re^n$, $x \neq 0$, risulta

$$x^T Ax > 0$$

Per le matrici simmetriche definite positive, l'errore di roundoff nell'esecuzione dell'algoritmo di eliminazione di Gauss senza pivoting, in un sistema aritmetico f.p. a precisione finita, si mantiene limitato in maniera paragonabile a quanto avviene con l'applicazione del pivoting.

Tuttavia, è possibile utilizzare un algoritmo di fattorizzazione *ad hoc*.

Teorema 2 (Fattorizzazione di Cholesky) *Sia $A \in \Re^{n \times n}$ simmetrica definita positiva. Allora esiste una ed una sola matrice triangolare inferiore $L \in \Re^{n \times n}$, con $l_{k,k} > 0$ per $k = 1$, ..., n, tale che*

$$A = LL^T$$

La matrice L è detta *fattore di Cholesky* di A.

4.6.1 Algoritmo di Cholesky

```
void cholesky (double *A, int n)
{
        /* INPUT
         * n        : ordine di A
         * A[n][n]  : matrice da fattorizzare */

        int i, j, k; /* indici */

        for( j = 0; j < n; j++ )
        {
                /* calcolo elementi diagonali */
                for( k = 0; k < j - 1; k++ )
                        A[j][j] -= pow(A[j][k], 2);

                A[j][j] = sqrt (A[j][j]);

                /* calcolo elementi non diagonali
                 * del triangolo inferiore */
                for( i = j + 1; i < n; i++ )
                {
                        for( k = 0; k < j - 1; k++ )
                                A[i][j] -= A[i][k] * A[j][k];
                        A[i][j] /= A[j][j];
                }
        }
}
```

Ogni elemento di $\tilde{L}$ è limitato dalla radice quadrata dell'elemento diagonale di A che si trova sulla medesima riga, indipendentemente dalla grandezza degli elementi *pivot* che compaiono al denominatore nei calcoli e dei conseguenti *moltiplicatori*. In altri termini, gli elementi della matrice L non possono crescere in maniera tale da invalidare il calcolo. L'algoritmo di Cholesky risulta quindi stabile.

Risolvere un sistema di equazioni lineari con matrice simmetrica definita positiva, equivale alla risoluzione di un sistema perturbato, dove le perturbazioni sui coefficienti rimangono limitate.

Si possono, tuttavia, presentare delle *difficoltà numeriche*: in generale, nell'esecuzione dell'algoritmo di Cholesky si possono avere dei problemi di instabilità numerica nel caso in cui la matrice A non sia sufficientemente definita positiva, ovvero se uno o più autovalori sono prossimi a zero.

4.7 Complessità asintotiche dei metodi di risoluzione

Diagonale:

$$T_{diag}(n) = \Theta(n)$$

Back e forward substitutions:

- $T_{bs}(n) = T_{fs}(n) = n^2 = \Theta(n^2)$
- $S_{bs}(n) = S_{fs}(n) = n^2 + 2n = \Theta(n^2)$

Gauss:

- $T_{Gauss}(n) = \frac{2(n^3-n)}{3} + \frac{n^2-n}{2} = \Theta(n^2)$
- $T_{Gauss+Back}(n) = \Theta(n^3)$
- $S_{Gauss}(n) = n(n+1) = \Theta(n^2)$

Pivoting parziale: $(n-k+1)$ confronti ad ogni, generico, passo k $(k = 1, ..., n-1)$, per un totale di

$$\Theta\left(\frac{n^2}{2}\right) \text{ confronti}$$

Fattorizzazione LU con pivoting parziale per matrici dense:

$$T_{LU+bs+fs} = \Theta\left(\frac{n^3}{3} + n^2\right)$$

Fattorizzazione LU senza pivoting per matrici tridiagonali: si eseguono $2n-2$ moltiplicazioni o divisioni f.p. e $n-1$ addizioni f.p. La complessità di tempo è quindi

$$T_{LUtrid}(n) = 3n - 3 = \Theta(n) \text{ flops}$$

essendo gli elementi memorizzati direttamente nelle diagonali delle matrici, la complessità di spazio è

$$S_{LUtrid}(n) = 3n - 2 = \Theta(n)$$

Back e forward substitutions per matrici tridiagonali: nei due algoritmi, la soluzione è memorizzata sovrascrivendo il vettore dei termini noti, senza ricorrere all'uso di un'area di memoria aggiuntiva. La loro complessità di spazio è dunque $\Theta(n)$.

Nella *forward*, vengono eseguite $n-1$ moltiplicazioni e $n-1$ addizioni f.p., pertanto la complessità di tempo è $\Theta(n)$.

Nella *backward*, vengono eseguite $2n-1$ moltiplicazioni o divisioni e $n-1$ addizioni f.p., pertanto la complessità di tempo è $\Theta(n)$.

Inversa di una tridiagonale: nel caso in cui l'inversa sia determinata con il metodo di Gauss, sono necessarie, trascurando i termini costanti che non dipendono da n,

$$2n^2 + n \text{ operazioni di moltiplicazione o divisione,}$$

che in termini asintotici corrispondono a $\Theta(n^2)$.

Fattorizzazione LU senza pivoting per matrici a banda: la memorizzazione a banda è, in termini di complessità di spazio, pari a $\Theta((p+q+1)n)$ anzichè $\Theta(n^2)$, e risulta molto vantaggiosa per $p, q << n/2$.

La complessità di tempo è $\Theta(npq)$: al crescere di p e q, tale complessità si avvicina a $\Theta(n^3)$, ovvero a quella di un'algoritmo per una matrice di ordine n, pertanto l'algoritmo è significativamente vantaggioso se $p, q << n$.

Back e forward substitutions per matrici a banda: dato che L ed U sono memorizzate in place, la complessità di spazio è la stessa della fattorizzazione LU per entrambi i metodi, cioè $\Theta((p+q)n)$.

Le complessità di tempo sono:

- $T_{FSband}(n) = \Theta(np)$;
- $T_{BSband}(n) = \Theta(nq)$.

Fattorizzazione di Cholesky: Al generico passo j,

- il calcolo dell'elemento diagonale $l_{j,j}$ richiede $j-1$ moltiplicazioni, altrettante addizioni ed una radice quadrata, ovvero

$$2(j-1) \text{ flop } + 1 \text{ radice quadrata;}$$

- il calcolo di un elemento extradiagonale $l_{i,i}$, $i > j$, richiede $j-1$ moltiplicazioni, altrettante addizioni ed una divisione, e quindi il calcolo degli $n-j$ elementi extradiagonali della colonna j-ma richiede

$$(n-j)(2j-1) \text{ flop.}$$

La complessità di tempo è dunque

$$T_{Chol}(n) = \Theta\left(\frac{n^3}{6}\right)$$

Se per memorizzare la matrice A, e quindi la matrice L, si utilizza un array bidimensionale di dimensione $n \times n$, la complessità di spazio dell'algoritmo di Cholesky è:

$$S_{Chol}(n) = n^2;$$

se invece si utilizza uno schema di memorizzazione packed, la complessità di spazio diventa

$$S_{Chol}^{packed}(n) = \frac{n(n+1)}{2} = \Theta\left(\frac{n^2}{2}\right)$$

e quindi risulta dimezzata.

4.8 Calcolo matriciale in Matlab

Metodo di Gauss:

```
x = A \ b
```

Fattorizzazione LU:

```
[L, U, P] = lu (A);
y = L \ (P*b);
x = U \ y
```

Norma vettoriale:

```
norm (X, p) % p = 1, 2, 'inf'
```

Norma matriciale:

```
cond (A, p) % p = 1, 2, 'inf'
```

Indice di condizionamento relativo:

```
cond (A, p) * cond (inv(A), p) % p = 1, 2, 'inf'
```

5 Rappresentazione dei dati

5.1 Fitting

Dato un insieme finito di dati $D = \{(x_i, y_i)_{i=1,...,n}\}$ appartenenti ad un intervallo I, tale cioè che $I \supset x_i$, ogni funzione f, definita su I che descrive D, si dice *fitting* o *modello* per D; tale funzione è poi detta **interpolante** D se sono verificate delle condizioni sulla funzione e/o sulle sue derivate nei punti assegnati, cioè se sono soddisfatte le seguenti condizioni:

$$f(x_i) = y_i \quad (\text{in generale } f^{(j)}(x_i) = y_i^j,\ j \in J \subseteq N_0) \qquad \forall i = 1, ..., n\ .$$

Tali condizioni sono dette *condizioni di interpolazione* e caratterizzano il modello interpolante.

Se invece si richiede che la funzione f sia tale che [2]:

$$f = arg \quad min|f(x_i) - y_i| \qquad i = 1, ..., n$$

allora la funzione f fornisce un modello **approssimante**. Per determinare un modello per D tale che:

i) rappresenti i dati (x_i, y_i)

ii) conservi eventuali altre proprietà della correlazione tra grandezze

iii) consenta di ottenere nuove informazioni eventualmente richeste

è dunque necessario in primo luogo stabilire se f debba:

- solo approssimare;

[2] Tale problema è detto *di migliore approssimazione*.

- solo interpolare;
- interpolare ed approssimare.

Nel caso in cui si scelga un modello approssimante, è poi necessario anche poter stabilire una **misura** di quanto f si *scosti* dai punti in D.

Nell'interpolazione polinomiale, nel caso in cui i dati da rappresentare siano affetti da *errori trascurabili* si può usare il polinomio interpolante di Lagrange costruito mediante la formula di Newton, se il numero di dati da rappresentare non è troppo alto. Altrimenti si può usare la spline cubica, naturale o non. Invece, nel caso in cui i dati da rappresentare siano affetti da *errori non trascurabili*, si può usare un modello approssimante, quale la migliore approssimazione nel senso dei minimi quadrati.

5.2 Problema di interpolazione di Lagrange

Dati n nodi distinti $(x_i)_{i=1,...,n}$, ed n valori $(y_i)_{i=1,...,n}$, determinare una funzione f tale che:

$$f(x_i) = y_i \ , \quad i = 1, \ ... \ , \ n$$

Tali condizioni sono dette **condizioni di interpolazione di Lagrange**.

Il problema di interpolazione di Lagrange richiede quindi che per ciascun nodo sia assegnata una ed una sola condizione.

5.3 Problema di interpolazione di Hermite

Assegnati n nodi distinti $(x_i)_{i=1,...,n}$, n interi positivi l_1, l_2, ..., l_n, tali che:

$$\sum_{i=1}^{n} l_i = m$$

ed m valori (y_i^j) con $i = 1, ..., n$; $j = 0, ..., l_i - 1$, determinare la funzione f tale che:

$$f^{(j)}(x_i) = y_i^j \qquad i = 1, \ ... \ , \ n; \ j = 0, \ ... \ , \ l_i - 1$$

Il problema di interpolazione di Hermite richiede quindi che per ciascun nodo sia assegnata almeno una condizione; inoltre, se in un nodo è assegnata una condizione sulla derivata di un ordine q allora devono essere necessariamente assegnate in quel nodo tutte le condizioni sulle derivate di ordine inferiore, cioè le condizioni sulle derivate di ordine j con $j = 0 \ , \ ... \ , \ q - 1$.

5.4 Problema di interpolazione polinomiale di Lagrange

Assegnati due punti *distinti* del piano, $P_1 \equiv (x_1, y_1)$ e $P_2 \equiv (x_2, y_2)$, con $x_1 \neq x_2$, determinare per quali valori di m esiste ed è unico il polinomio [3]:

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

di grado al più m, tale che:

$$\begin{cases} p(x_1) = y_1 \\ p(x_2) = y_2 \end{cases}$$

[3] Cioè esiste un'unica (m + 1)-pla di valori a_0, a_1, ..., a_n.

Teorema 3 (unicità del polinomio interpolante di Lagrange) *Assegnati n nodi distinti* $(x_i)_{i=1,...,n}$ *ed n valori corrispondenti* $(y_i)_{i=1,...,n}$*, il polinomio p di grado al più m* ($p \in \Pi_m$)*, tale che:*

$$p(x_i) = y_i, \quad i = 1, \ldots, n$$

è unico se:

$$m = n - 1.$$

5.5 Unicità del polinomio interpolante di Hermite

Teorema 4 (unicità del polinomio interpolante di Hermite) *Assegnati n nodi distinti* $(x_i)_{i=1,...,n}$*, n interi positivi* $l_1, l_2, \ldots, l_n$*, tali che:*

$$\sum_{i=1}^{n} l_i = m$$

ed m valori (y_i^j) *con* $i = 1, \ldots, n$; $j = 0, \ldots, l_i - 1$*, il polinomio* $p \in \Pi_q$ *tale che:*

$$p^{(j)}(x_i) = y_i^j \qquad i = 1, \ldots, n;\ j = 0, \ldots, l_i - 1$$

è unico se

$$q \leq m - 1.$$

5.6 Formula di Lagrange

Per $p \in \Pi_{n-1}$:

$$p(x) = \sum_{i=1}^{n} y_i \prod_{j=1,\ j \neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

La complessità asintotica di tempo per richiesta dall'algoritmo per la costruzione del polinomio interpolante è

$$T(n) = \Theta(n^2) \text{ flops.}$$

Per la valutazione del polinomio interpolante così costruito, occorre valutare per un fissato valore di x, ciascun polinomio fondamentale di Lagrange e quindi al variare di x occorre effettuare di nuovo tutte le valutazioni, con una complessità asintotica di tempo di

$$T(n) = \Theta(n^2) \text{ flops.}$$

5.7 Differenze divise

Le differenze divise si utilizzano per costruire i coefficienti del polinomio interpolante espresso nella formula di Newton.

Differenza divisa di ordine n:

$$\begin{array}{lcl} y[x_i] & = & y_i \\ y[x_1, ..., x_{n+1}] & = & \frac{y[x_2,...,x_{n+1}] \ - \ y[x_1,...,x_n]}{x_{n+1} - x_1} \end{array}$$

L'algoritmo per il calcolo dei coefficienti del polinomio interpolante espresso nella formula di Lagrange calcola, in ciascuno degli $n-1$ passi, $n-i$ differenze divise, se i indica il generico passo. Complessivamente calcola:

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

differenze divise e, poichè ciascuna richiede due addizioni e una divisione, la complessità asintotica di tempo dell'algoritmo per il calcolo dei coefficienti del polinomio è

$$T(n) = \Theta(n^2) \text{ flops.}$$

Tutti i coefficienti del polinomio interpolante di Lagrange sono esprimibili in termini di differenze divise relative ai nodi di interpolazione.

Teorema 5 *Sia $p \in \Pi_{n-1}$ il polinomio interpolante di Lagrange relativo ai punti $(x_i, y_i)_{i=1,...,n}$. Se a_k è il $k-$simo coefficiente di p espresso nella formula di Newton, allora a_k è la differenza divisa di ordine k relativa ai nodi x_1, x_2, ..., x_{k+1}:*

$$a_k = y[x_1, x_2, ..., x_{k+1}]$$

cioè:

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

Costruzione della tavola delle differenze divise
La tavola avrà $(n-1)+2$ colonne. Le prime due colonne conterranno x_i, e y_i, dove $i = 1$, ..., n; le rimanenti colonne conterranno gli ordini k delle differenze divise, con $k = 1$, ..., $n-1$.

Ogni differenza divisa va calcolata, utilizzando la definizione della differenza divisa di ordine n, a partire dai due elementi ad essa adiacenti nella colonna precedente. Si procede quindi per $n-1$ passi, calcolando le differenze divise, per colonne, da sinistra verso destra, fino all'unico elemento della colonna $k = n-1$.

x_i	y_i	$k=1$	$k=2$	$\cdots$	$k=n-1$
x_1	y_1				
		$y[x_1, x_2]$			
x_2	y_2		$y[x_1, x_2, x_3]$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$y[x_1, x_2, \ \ldots \ , x_n]$
x_{n-2}	y_{n-2}				
		$y[x_{n-2}, x_{n-1}]$			
x_{n-1}	y_{n-1}		$y[x_{n-2}, x_{n-1}, x_n]$		
		$y[x_{n-1}, x_n]$			
x_n	y_n				

Tavola delle differenze divise

I coefficienti del polinomio interpolante si trovano lungo la diagonale superiore.

Se si aggiunge un nodo all'insieme dei dati, la formula di Newton per il polinomio di Lagrange $p(x)$, interpolante gli $n+1$ nodi, sarà rappresentata in funzione del polinomio $q(x)$, interpolante gli n nodi, come segue:

$$p(x) = q(x) + a_n(x - x_1) \cdot (x - x_2) \ ... \ (x - x_n)$$

con $a_n = y[x_1, x_2, \ ... \ , x_{n+1}]$.

5.8 Spline

Il polinomio interpolante, se il numero di punti di interpolazione è elevato, non fornisce in generale un modello accettabile. Infatti, al crescere del numero di punti aumenta il grado del polinomio interpolante e aumentano anche le oscillazioni del polinomio corrispondente ottenendo un modello non sempre coerente con l'andamento dei dati [4]. Pertanto, in tali condizioni, è preferibile utilizzare una *spline*.

Spline di grado m (di ordine m + 1)
Sia

$$K = \{x_0 < x_1 < x_2 < \ ... \ < x_n < x_{n+1}\}$$

con gli x_i, $i = 1, \ ... \ , n$ appartenenti all'asse reale e $x_0 = -\infty$ e $x_{n+1} = +\infty$, una funzione $s(x)$, definita su tutto l'asse reale, è una **spline di grado m** se:

1. $s(x) \equiv p_i(x) \in \Pi_m$ per $x \in [x_i, x_{i+1}], \quad i = 0, \ ... \ , n$;
2. $s(x) \in C^{m-1}((-\infty, \ +\infty))$: su tutto l'asse reale la funzione $s(x)$ è continua con le sue derivate fino all'ordine $m - 1$.

Indicato con $S_m(K)$ l'insieme delle spline di grado m costruite sull'insieme K, si ha che:

$$S_m(K) \supset \Pi_m$$

Le spline in assoluto più utilizzate sono quella di grado 1 (lineare) e quella cubica.

Problema di interpolazione mediante spline cubica
Fissati n nodi x_i, $i = 1, ..., n$ e fissati n valori corrispondenti y_i, $i = 1, ..., n$, si vuole costruire una funzione spline cubica, $s \in S_3(K)$, con

$$K = \{x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1}\},$$

dove

$$x_0 = -\infty \text{ e } x_{n+1} = +\infty$$

tale che $s(x_i) = y_i$, $i = 1, ..., n$.

Per costruire la spline interpolante relativa ad un insieme di nodi, le condizioni di regolarità non bastano ad individuarla univocamente. È necessario imporre condizioni aggiuntive

[4] Tale comportamento dipende dal grado del polinomio interpolante, che a sua volta dipende dal numero dei punti da interpolare.

che specializzino il tipo di spline da costruire (ossia atte a restringere l'insieme delle soluzioni ammissibili). Una delle spline più utilizzate nelle applicazioni è la **spline cubica naturale**[5].

Spline naturale di grado m
Una spline s di grado dispari $m = 2j - 1$, relativa all'insieme dei nodi K, è una **spline naturale** se, in ciascuno dei due intervalli

$$(-\infty, x_1),\ (x_n, +\infty)$$

coincide con un polinomio di grado minore o uguale a $j - 1$. Ciò significa che

$$s^{(h)}(x_1) = s^{(h)}(x_n) = 0, \quad h = j,\ j+1,\ \ldots,\ 2j-2.$$

La spline cubica naturale soddisfa tali condizioni con $j = 2$.

5.8.1 Un algoritmo per la costruzione e valutazione della spline naturale cubica interpolante

Il procedimento per la costruzione e valutazione, in un valore fissato $\tilde{x}$, della funzione spline cubica naturale interpolante un insieme di punti assegnati (x_i, y_i), $i = 1, \ldots, n$ consiste in:

1. costruzione della matrice A e del vettore dei termini noti $3B \triangle f$;
2. risoluzione del sistema $A\lambda = 3B \triangle f$;
3. determinazione dell'intervallo $[x_i, x_{i+1}]$ a cui $\tilde{x}$ appartiene;
4. calcolo dei coefficienti del polinomio di Hermite, nell'intervallo $[x_i, x_{i+1}]$;
5. valutazione in $\tilde{x}$.

Le complessità di tempo sono:

- risoluzione di un sistema tridiagonale: $T(n) = \Theta(n) flops$
- ricerca binaria dell'intervallo di appartenenza del punto di valutazione: $T(n) = \Theta(\log_2 n)$ confronti
- costruzione degli $n - 1$ polinomi di terzo grado: $T(n) = \Theta(n)$ flop
- valutazione (algoritmo di Horner): $T(n) = \Theta(n)$ flop

5.9 L'approssimazione polinomiale dei minimi quadrati

Questo modello viene usato per la rappresentazione di quei dati che sono affetti da un errore *non trascurabile* (derivante dagli strumenti di misura utilizzati, oppure da opportune semplificazioni). In questo caso, per evitare di esaltare l'errore presente nei dati, è più ragionevole richiedere che la funzione non debba assumere i valori assegnati ma che invece si scosti poco da questi in modo da non perdere completamente le informazioni in essi ottenute e, allo stesso

[5]La denominazione *naturale*, assegnata a questo particolare tipo di funzione, è dovuta al fatto che lo strumento spline al di fuori dei pesi che lo vincolano, e cioè all'esterno dell'intervallo $[x_1, x_n]$, tende naturalmente ad assumere la forma di linea retta.

tempo, fornire una rappresentazione attendibile. Si parla quindi di **modello approssimante**.

In base alla scelta della misura dello scostamento della funzione approssimante dai dati e ai vincoli che imponiamo sullo scostamento, si caratterizza il tipo di approssimazione. In particolare, se scegliamo come misura dello scostamento la somma dei quadrati delle distanze dei punti assegnati dal grafico della funzione approssimante, e imponiamo che tale scostamento sia il minimo possibile, allora la funzione che si ottiene viene detta **migliore approssimazione nel senso dei minimi quadrati**.

In particolare, il tipo di funzione utilizzato è il polinomio.

Teorema 6 *Assegnati n nodi distinti ed n valori corrispondenti, il sistema di equazioni normali relativo alla costruzione del polinomio $p(x) \in \Pi_m$ di migliore approssimazione nel senso dei minimi quadrati, ammette una ed una sola soluzione se e solo se i nodi sono a due a due distinti e se $n > m + 1$.*

Tenendo conto che fissati n punti il polinomio interpolante ha grado $n - 1$, bisogna escludere che il grado del polinomio approssimante, m, possa essere uguale a $n - 1$.

5.10 Interpolazione in Matlab

Legenda:

```
% n = numero nodi di interpolazione
% m = numero di punti di valutazione
% x = vettore nodi (a due a due distinti)
% y = vettore condizioni
% z = vettore punti di valutazione
```

Polinomio interpolante di Lagrange

```
% costruzione del polinomio interpolante
c = polyfit(x,y,n-1)

% valutazione del polinomio in corrispondenza degli m punti
% di valutazione z(i), componenti del vettore z
pz = polyval(c,z)
plot(x,y,'o',z,pz)
```

Spline cubica

```
% creazione spline cubica naturale
spNat = csape (x, y,'variational');
% valutazione dei punti del vettore z
values = fnval (spNat, z);
% grafico delle valutazioni
plot (z, values, 'bd')

% creazione e valutazione spline cubica non naturale
plot(z, csapi(x,y,z),'k-',x,y,'ro')
```

Migliore approssimazione nel senso dei minimi quadrati

```
% tratto da una function

% costruzione della matrice dei coefficienti
for i=1:n
A(i,1) = 1;
A(i,2) = x(i);
end

% valutazione
y = y';
coeff = A' * A;
noto = A' * y;
c = coeff\noto;

% calcolo deviazione standard
sigma = 0;
for i = 1:n
sigma += ( y(i) - polyval(c, x(i)) )^2;
end
sigma /= n;
sigma ^= 0.5;

% grafico della funzione
fprintf('\nDeviazione standard del vettore y rispetto a f(x): %f\n', sigma)
plot(x, polyval(c,x),'-r')
title('Migliore approssimazione nel senso dei minimi quadrati')
```

6 Quadratura

6.1 Formula trapezoidale

Data una figura piana non lineare R, un metodo per approssimarne l'area è quello di utilizzare un trapezio T con basi di misura $f(a)$ e $f(b)$ ed altezza $(b-a)$, cioè

$$I[f] \equiv A(R) \simeq A(T) = \frac{(b-a)}{2}\,[f(a)+f(b)] = T[f].$$

Questa formula è detta **formula trapezoidale**, ed equivale a calcolare l'area al di sotto del segmento di retta passante per la coppia di punti $(a, f(a))$, $(b, f(b))$.

6.1.1 Formula trapeziodale composita

Suddiviso l'intervallo $[a, b]$ in m sottointervalli di ampiezza

$$\tau = \frac{(b-a)}{m},$$

mediante i punti equidistanziati:

$$a = x_0 < x_1 < \ ... \ < x_m = b \quad \text{con } x_i = a + i\tau, \quad i = 0, \ ... \ , m$$

la formula di quadratura $T_m[f]$ che si ottiene applicando, in ognuno degli intervalli $[x_{j-1}, x_j]$, con $j = 1, \cdots, m$, la formula trapezoidale $T[f]$, è detta **formula trapezoidale composita**. In questo caso la distanza tra i nodi h coincide con τ, per cui, posto

$$x_i = a + ih, \quad i = 0, \ ... \ , \ m-1, \quad x_m = b,$$

si ha:

$$\int_a^b f(x)dx \simeq T_m[f] = h\left(\frac{f(x_0)}{2} + f(x_1) + \cdots + f(x_{m-1}) + \frac{f(x_m)}{2}\right).$$

6.2 Formula di quadratura

Fissata una funzione integrabile nel senso di Riemann ed un intero n, dati n punti $x_i \in [a, b]$ detti **nodi** e n valori A_i detti **pesi**, la combinazione lineare:

$$Q[f] = A_1 f(x_1) + \ ... \ + A_n f(x_n) = \sum_{i=1}^{n} A_i f(x_i)$$

prende il nome di **formula di quadratura**.

Una formula di quadratura rappresenta un'approssimazione dell'integrale $I[f]$, per cui è possibile dare anche la seguente definizione:

Errore di discretizzazione
La differenza:

$$E[f] = I[f] - Q[f] = \int_a^b f(x)dx - \sum_{i=1}^{n} A_i f(x_i)$$

è l'**errore di discretizzazione della formula di quadratura** $Q[f]$.

6.2.1 Formula di quadratura composita

Fissata una formula di quadratura

$$Q[f] = \sum_{i=1}^{n} A_i f(x_i)$$

ed un intervallo $[a, b]$, si divida tale intervallo in m sottointervalli

$$[t_{j-1}, t_j] \quad (j = 1, \ ... \ , \ m).$$

Si definisce $Q_m[f]$ **formula di quadratura composita** la formula che si ottiene applicando $Q[f]$ in ogni sottointervallo $[t_{j-1}, t_j]$ di ampiezza $\tau_j = (t_j - t_{j-1})$, cioè:

$$Q_m[f] = \sum_{j=1}^{m} Q^{(j)}[f] = \sum_{j=1}^{m} \sum_{i=1}^{n} A_i^{(j)} f(x_i^{(j)})$$

dove con $Q^{(j)}[f]$ si è indicata la formula $Q[f]$ nel j-mo intervallo $[t_{j-1}, t_j]$, con $x_i^{(j)}$ si è indicato l'i-mo nodo di $Q^{(j)}[f]$ e con $A_i^{(j)}$ l'i-mo peso di $Q^{(j)}[f]$.

6.3 Formule innestate

Due formule $Q'[f]$ e $Q''[f]$, relative ad uno stesso intervallo, tali che l'insieme dei nodi di $Q'[f]$ è contenuto nell'insieme dei nodi di $Q''[f]$, costituiscono una **coppia di formule innestate**.

Data quindi una coppia di formule innestate $Q'[f]$ e $Q''[f]$, l'errore $E''[f] = I[f] - Q''[f]$ può essere stimato mediante la differenza:

$$|E''[f]| \simeq \alpha |Q'[f] - Q''[f]| \quad \alpha > 0$$

dove α è una costante che dipende dalle formule di quadratura utilizzate.

La disponibilità di formule innestate è particolarmente utile per ridurre la complessità computazionale delle formule di quadratura (e più in generale, di un algoritmo per la quadratura). Infatti essa è determinata attraverso **il numero di valutazioni della funzione integranda** nei nodi x_i, perché il numero di operazioni relative a tale calcolo è predominante.

6.4 Algoritmi adattativi per la quadratura

Un **algoritmo adattativo** per la quadratura è un algoritmo che sceglie dinamicamente (cioè durante l'esecuzione) la distribuzione dei nodi, in maniera da adattare il partizionamento dell'intervallo di integrazione al particolare andamento della funzione integranda. Un algoritmo in cui l'insieme dei nodi è scelto secondo uno schema fissato, indipendentemente dalla funzione integranda, è detto **algoritmo non adattativo**.

Nell'ambito degli algoritmi adattativi si distinguono due strategie: la strategia **adattativa globale** e la strategia **adattativa locale**.

Dal punto di vista implementativo la strategia globale può essere realizzata mediante l'uso di una lista ordinata secondo le stime crescenti degli errori $E(\triangle_i)$. In questo modo l'intervallo da suddividere (cioè quello con massima stima dell'errore) è sempre in testa alla lista. Viceversa la strategia locale può essere realizzata mediante l'uso di una pila, costruita inserendo durante ogni suddivisione i due intervalli così ottenuti nella testa della pila, prima quello di destra poi quello di sinistra; in tal modo l'intervallo da esaminare (quello più a sinistra) si trova sempre in testa alla pila.

Entrambe le strategie presentano vantaggi e svantaggi, anche se le routine più efficienti attualmente esistenti sono basate sulla strategia globale. Quest'ultima infatti, a parità di tolleranza richiesta, utilizza generalmente un numero minore di valutazioni della funzione integranda rispetto alla strategia locale. Inoltre, se l'algoritmo termina per aver raggiunto il massimo numero di valutazioni della funzione integranda, è sempre disponibile una stima di $I[f]$. Diversamente, la strategia locale produce una stima dell'integrale della funzione solo fino all'intervallo che ha esaminato per ultimo procedendo dall'estremo inferiore dell'intervallo $[a, b]$ verso quello superiore.

6.5 La quadratura in Matlab

Strategia adattativa locale

```
quad('fun', a, b, tol)

% 'fun' = nome della funzione integranda (fun.m)
% a, b = estremi dell'intervallo
% tol = tolleranza (opzionale)
```

7 Calcolo di π

7.1 Metodo di Archimede

Il metodo di Archimede è instabile, poiché al passo i-esimo si moltiplica il termine precedente per $\mu = 2^{i-2}$ che è, quindi, anche il fattore di amplificazione dell'errore di round-off.

7.2 Metodo di Leibniz

Nel metodo di Leibniz, che utilizza uno sviluppo in serie, l'errore round-off cresce linearmente con il numero degli addendi della serie; questo metodo è quindi stabile. Tuttavia, a causa della lenta velocità di convergenza della sua formula, si ottengono risultati accurati solo per n molto grande.

7.3 Metodo di Viete

Il metodo di Viete è stabile, in quanto il fattore di amplificazione dell'errore di round-off nella formula di ricorrenza del passo i-esimo è limitato. Questo metodo è computazionalmente più vantaggioso di quello di Leibniz, a causa della lenta convergenza della formula di Leibniz.