Università degli Studi di Venezia
Corso di Laurea in Informatica

Appunti per le lezioni di Calcolo Numerico

Prof. Flavio Sartoretto

2 giugno 2009

Indice

Capitolo 1

Introduzione

F. Sartoretto
Calcolo Numerico

Testi adottati:

- A. Quarteroni e F. Saleri. *Introduzione al Calcolo Scientifico*. Springer Verlag Italia, III edizione, 2006.
- F. Sartoretto e M. Putti. *Introduzione alla Programmazione per Elaborazioni Numeriche*. Edizioni Libreria Progetto, Padova, 2008.
- F. Sartoretto, *Appunti di Calcolo Numerico*, Dispense, WEB page del Docente, 2009.

Testi consigliati:

- G. Gambolati. *Lezioni di Metodi Numerici per Ingegneria e Scienze Applicate*. Cortina, Padova, 1994.

Prerequisiti:

- Calcolo I, con cenni sui numeri complessi.
- Calcolo II: risoluzione di equazioni differenziali ordinarie lineari a coefficienti costanti, calcolo di derivate parziali.
- Algebra delle matrici.

In questi appunti, si fa riferimento all'area WEB dedicata al corso. La directory principale viene chiamata `$CN` = `www.dsi.unive.it/~sartoret-/italian/didattica/CalcoloNumerico/`.

1.1 Notazioni

- $a \simeq b$ significa *"a" é approssimativamente uguale a "b"*.
- $f(x) = O(g(x))$ **per** $x \to a$ (si legge $f(x)$ *é di ordine "O-grande" di* $g(x)$ *per* $x \to a$), significa che la funzione

$$\left|\frac{f(x)}{g(x)}\right|$$

é limitata per $x \to a$, ossia esiste un intorno I_a di a ed esiste $M \in \Re^+$ t.c. $(\forall x \in I_a)\ |f(x)/g(x)| \leq M$.

1.1.1 Esempi

- $\sin(x) = O(1)$ per $x \to \infty$.

  Infatti $|\sin(x)/1| \leq 1$.

  Notare che *non esiste* il $\lim_{x\to\infty} \sin(x)/1$. Notare anche che $\sin(x) = O(x^k)$, $k \geq 0$, quando $x \to \infty$!

- $\sin(1/x) = O(1)$ per $x \to 0$.

  Per le stesse ragioni di prima.

- $2(n+1)n = O(n^2)$ per $n \to +\infty$. Infatti $\lim_{n\to+\infty} |2(n+1)n/n^2| = 2$.
-

Una relazione piú forte è:

$$f(x) \approx g(x) \iff$$
$$f(x) = O(g(x)) \quad e \quad g(x) = O(f(x)).$$

Si dice che $f(x)$ e $g(x)$ sono simili.

1.2 Costo Computazionale

Sia $\mathbf{A}$ una matrice $n \times n$ e $\mathbf{v}$ un vettore di n componenti.

Il prodotto (algebrico)

$$\mathbf{w} = \mathbf{A}\mathbf{v}$$

richiede n prodotti e $n-1$ somme per calcolare ogni elemento di $\mathbf{w}$. In totale richiede quindi $n(2n-1)$ operazioni floating point.

- Se consideriamo unitario il costo di una moltiplicazione e un'addizione, il suo costo è $O(2n^2 - n) = O(n^2)$, infatti

$$\lim_{n \to +\infty} \frac{2n^2 - n}{n^2} = 2.$$

- Se consideriamo unitario il costo di *una somma piú una moltiplicazione*, il costo è $O(n(n-1)) = O(n^2)$.

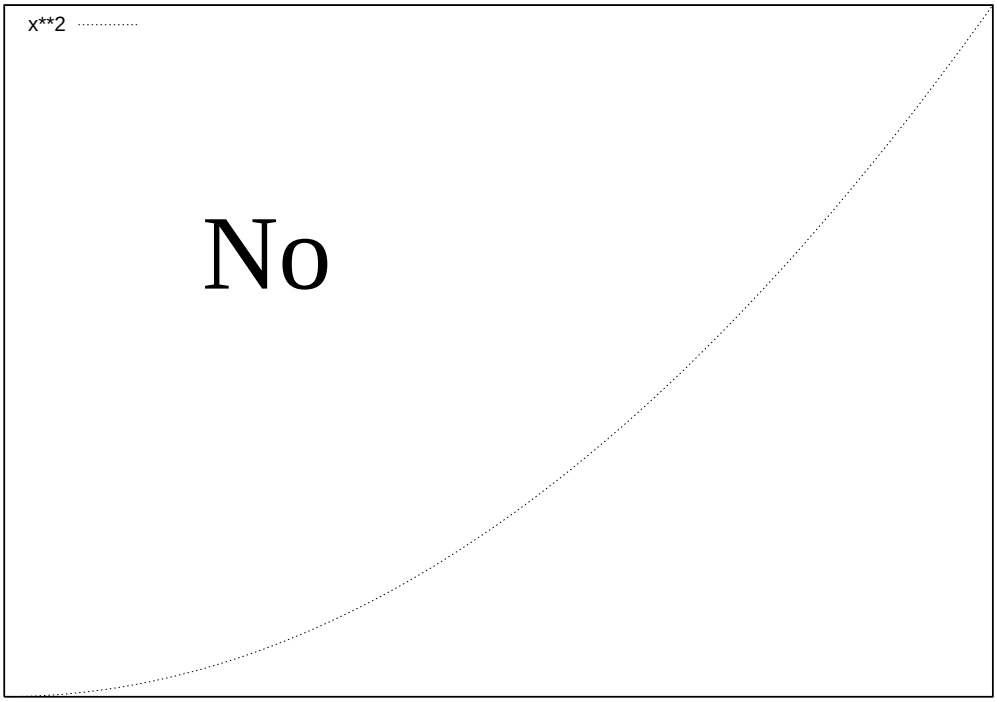
- Se consideriamo unitario il costo di un *prodotto scalare tra vettori di dimensione* n, allora il costo è $O(n)$.

1.3 Documentazione dei risultati

Non basta calcolare risultati numerici corretti, bisogna saperli anche presentare in maniera corretta.

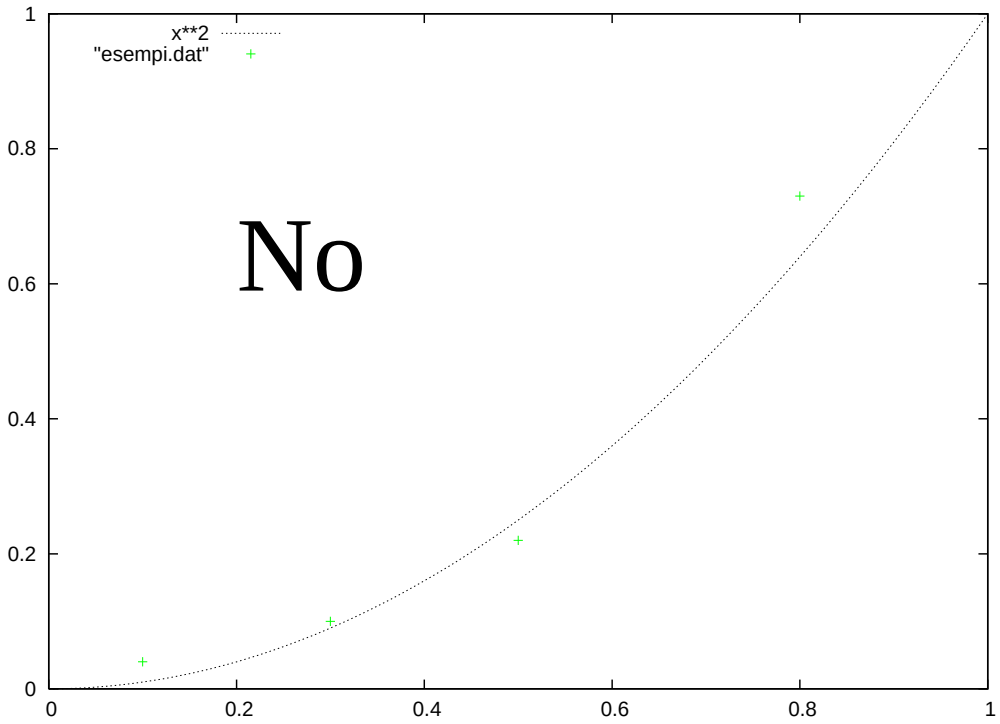
Valgono le seguenti regole:

Usare tabelle, ma con parsimonia: un tabulato con dieci pagine di risultati numerici è illeggibile! Riportare in tabella solo valori importanti e affidare a grafici la presentazione di altri risultati.

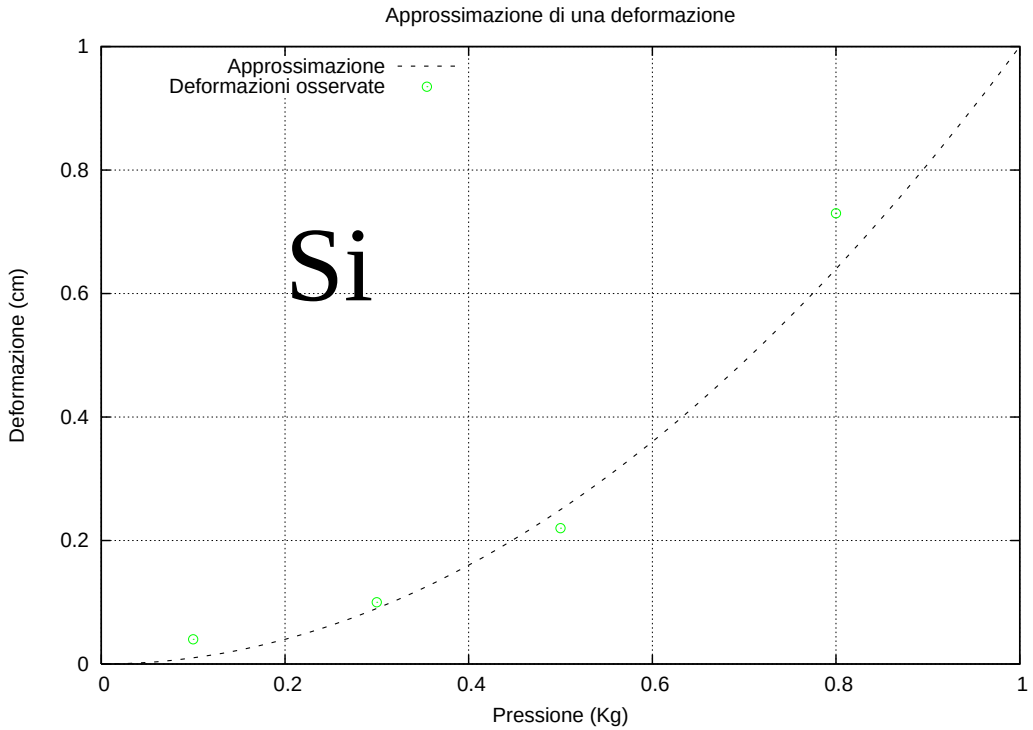


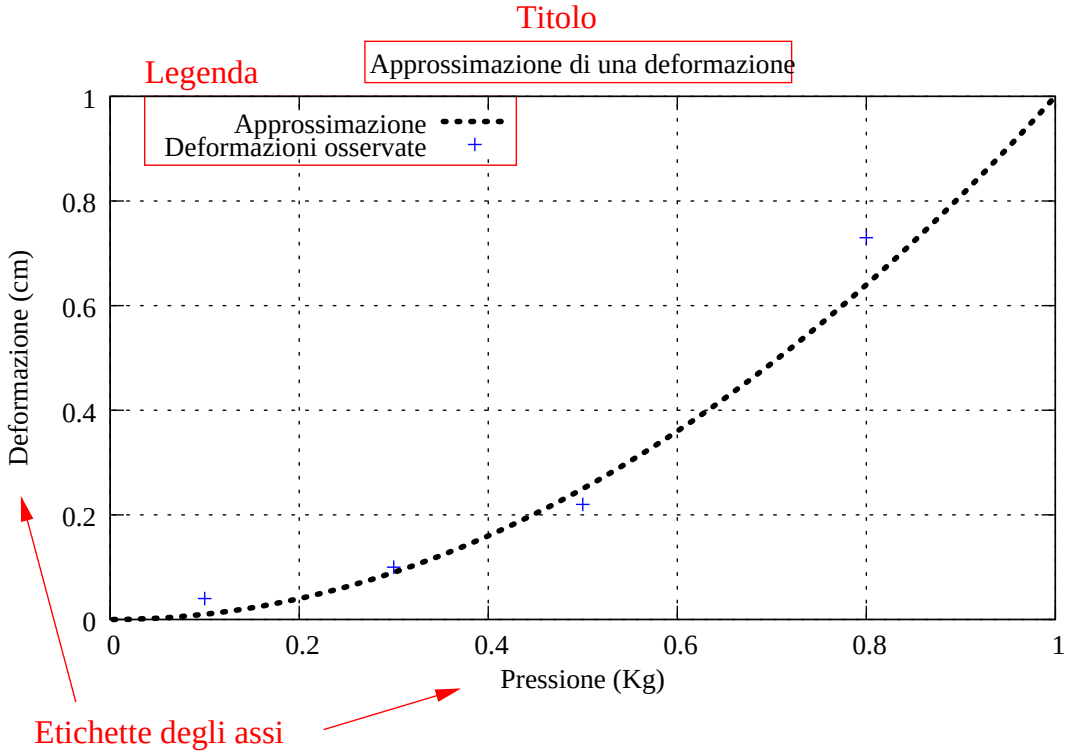
Nei grafici

- segnare sempre con un simbolo opportuno i valori calcolati, per separarli da eventuali linee approssimatrici;
- tratteggiare le linee approssimatrici;
- usare sempre la scala semilogaritmica per disegnare l'andamento di quantità che presentano forti variazioni (errori, scarti, etc.);
- indicare sempre chiaramente le scale in ascissa e ordinata;
- indicare sempre chiaramente il significato delle ascisse e delle ordinate e le eventuali unità di misura usate.

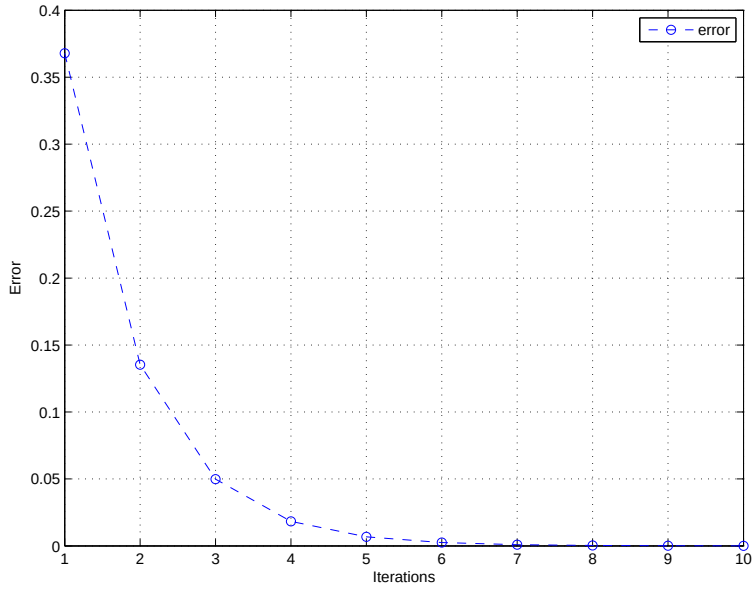


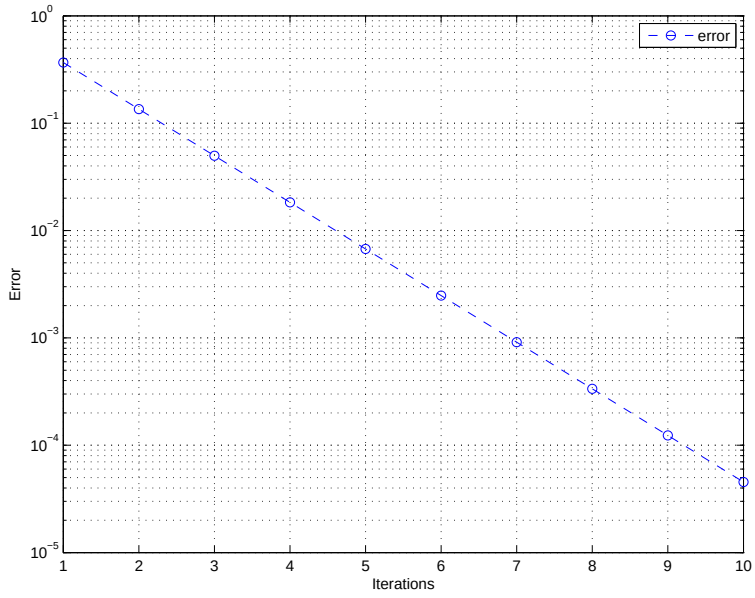
Quando vengono forniti dei dati e viene chiesto di approssimarli, riportare sempre i primi assieme ai secondi, nello stesso grafico e/o tabulato.





... uso di scale semi-logaritmiche e logaritmiche ...





1.4 Concetti di base.

Calcolo Numerico o Scientifico:

Disciplina che tenta di dare una risposta numerica ad un problema matematico usando un elaboratore digitale.

Obiettivo del Calcolo Numerico o Scientifico é

Trovare gli algoritmi che risolvono un problema matematico

- nel minor tempo,
- con il minimo impiego di risorse di calcolo,
- con la massima accuratezza possibile.

1.5 Definizioni

Un procedimento numerico si dice instabile se gli errori associati crescono fino a distruggere la soluzione.

Un problema si dice mal condizionato se una piccola variazione nei dati (può) provocare una grande variazione dei risultati.

Capitolo 2

Rappresentazione dei numeri.

2.1 Pentium FDIV flaw

Eseguendo sui primi Pentium l' operazione

```
EXA := (5-1/1000000)/(15-1/1000000);
```

Si otteneva il valore 0.333329, con un errore relativo: (EXA-0.333329)/EXA =

1.287461e-5.

Eseguendo la stessa operazione su un 486 si ottiene 0.33333329. L' errore relativo e':

(EXA-33333329/100000000)/EXA =
-1666671/499999900000000,
ossia circa -3.333343e-9.

2.2 Esempi di errori di arrotondamento.

2.2.1 Errori di accumulazione

La rappresentazione con un numero finito di cifre dá luogo ad errori. Ecco i valori ottenuti calcolando le quantità $xis = x_i^{(s)} = x_{i-1} + h$, $x_0^{(s)} = x_0$, $xip = x_i^{(p)} = x_0 + ih$, $i = 1, \ldots, n$. Il valore iniziale esatto è $x_0 = 1.3$.

```
Risultati ottenuti sul VAX con VAX FORTRAN
i      xis                    xip
0  1.29999995231628418  1.29999995231628418
1  1.34999990463256836  1.34999990463256836
2  1.39999985694885254  1.39999997615814209
```

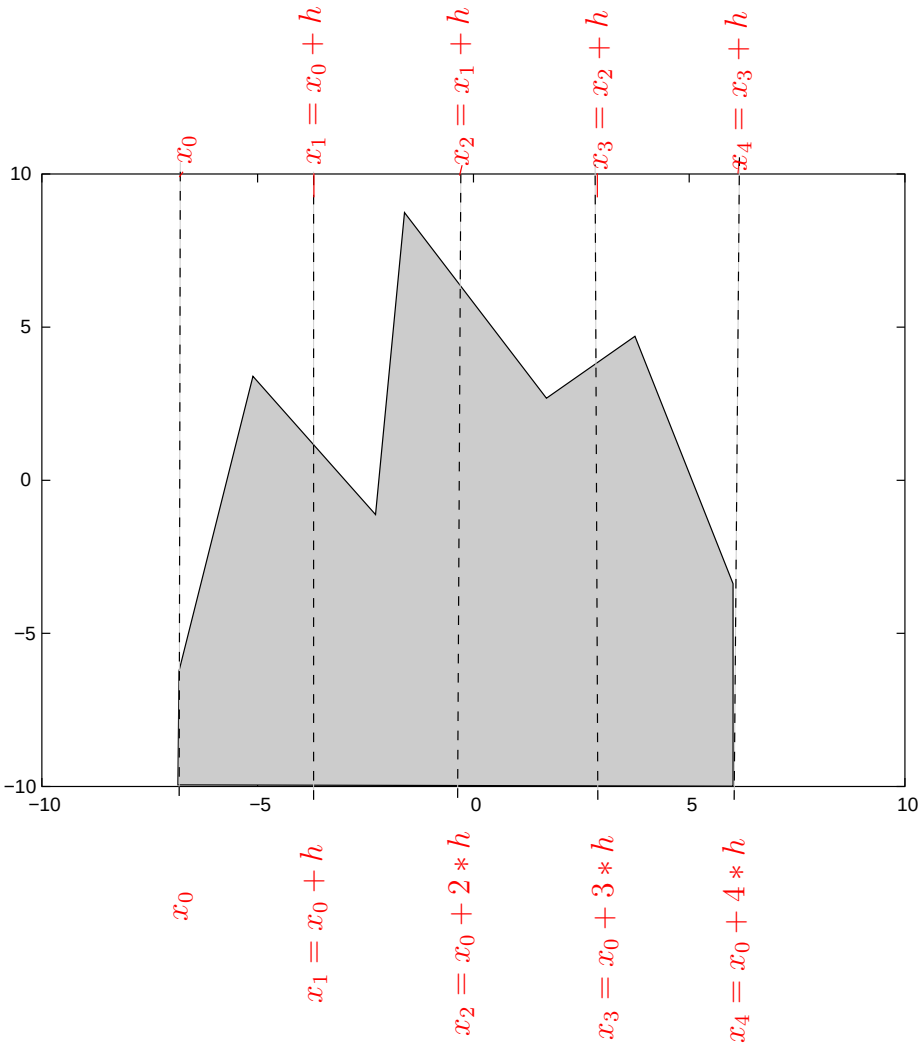


Figura 2.1: Discretizzazione uniforme di un intervallo.

```
3  1.44999980926513672   1.44999992847442627
4  1.49999976158142090   1.50000000000000000
5  1.54999971389770508   1.54999995231628418
6  1.59999966621398926   1.60000002384185791
7  1.64999961853027344   1.64999997615814209
8  1.69999957084655762   1.69999992847442627
9  1.74999952316284180   1.75000000000000000
```

2.3 Modi di arrotondamento

Consideriamo un numero floating point *normalizzato*, in un'aritmetica di base β, con *mantissa* o *significante* (*significand*) a p cifre, esponenti s, $e_{min} \leq s \leq e_{max}$,

$$x = \pm a_1.a_2 \ldots a_p \times \beta^q = m \times \beta^q$$

Ricordiamo che troncare x a $r \leq p$ cifre dà luogo al valore

$$tr_r(x) = \pm a_1.a_2 \ldots a_r \times \beta^q.$$

Due modi tipici di calcolare l' arrotondmento a r cifre di x sono i seguenti.

1. *Arrotondamento* (senza attributo) [Com91]

$$rd_r(x) = \text{sign}(m)\, tr(|m| + \frac{1}{2}\beta^{-r}) \times \beta^q.$$

2. *Arrotondamento pari* (*round to even*) [Gol91]

$$rd_r^{(e)}(x) = \begin{cases} rd_r(x) & \text{se } a_{r+1} \neq \beta/2, \\ \begin{cases} tr_r(x) & \text{se } a_r \text{ pari}, \\ rd_r(x) & \text{altrimenti.} \end{cases} & \text{altrimenti.} \end{cases}$$

I due modi differiscono quando vi è equilibrio, ossia se $\beta = 10$ quando la $(r+1)$-esima cifra è 5. In tal caso, $rd_p(x)$ è il piú vicino floating point che in valore assoluto è *piú grande* di $|x|$, mentre $rd_p^{(e)}(x)$ dipende dalla cifra a_r. Chiamiamo vicino minore, x_n, di x il piú grande floating point minore di x; chiamiamo vicino maggiore, x_g, di x il piú piccolo floating point maggiore di x. L'*arrotondamento pari* restituisce quello tra x_n e x_g che ha la cifra meno significativa pari.

Esempio: sia $\beta = 10$, $p = 3$. Se $x = 1.25$, $rd_2(x) = 1.3$, $rd_2^{(e)} = 1.2$, mentre se $x = 1.35$, $rd_2(x) = 1.4$, $rd_2^{(e)} = 1.4$. Se a_r è diverso da 5, i due modi coincidono: $x = 1.24$, $rd_2(x) = 1.2 = rd_2^{(e)}$, $x = 1.26$, $rd_2(x) = 1.3 = rd_2^{(e)}$, ecc.

Capitolo 3

Schemi ricorrenti.

Esercizio 3.0.1 Per calcolare i valori dei polinomi ortogonali $T_n^*(x)$, $n = 2, \ldots, 20$, per $x = 2$ e $x = 1/2$, si vuole utilizzare lo schema ricorrente

$$\begin{cases} T_{n+1}^*(x) &= (-2+4x)T_n^*(x) - T_{n-1}^*(x) \\ T_0^*(x) = 1, & T_1^*(x) = x \end{cases}$$

Supponiamo che T_0^* e T_1^* siano noti con errori ϵ_0 e ϵ_1 che siano uguali in modulo.

Nel caso $x = 2$ lo schema si può ritenere stabile?

Nel caso $x = 1/2$ lo schema si piò ritenere stabile?

Soluzione.

Le seguenti tabelle riassumono i risultati che si ottengono calcolando alcuni termini della formula ricorrente. Caso $x = 2$.

$$\begin{pmatrix} \epsilon_0 = \epsilon_1 & \epsilon_0 = -\epsilon_1 \\ \epsilon_0 & -\epsilon_0 \\ 5\,\epsilon_0 & -7\,\epsilon_0 \\ 29\,\epsilon_0 & -41\,\epsilon_0 \\ 169\,\epsilon_0 & -239\,\epsilon_0 \end{pmatrix}$$

Caso $x = 1/2$.

$$\begin{pmatrix} \epsilon_0 = \epsilon_1 & \epsilon_0 = -\epsilon_1 \\ \epsilon_0 & -\epsilon_0 \\ -\epsilon_0 & -\epsilon_0 \\ -\epsilon_0 & \epsilon_0 \\ \epsilon_0 & \epsilon_0 \\ \epsilon_0 & -\epsilon_0 \\ -\epsilon_0 & -\epsilon_0 \\ -\epsilon_0 & \epsilon_0 \\ \epsilon_0 & \epsilon_0 \end{pmatrix}$$

Quindi nel caso $x = 2$ lo schema **non** é stabile, lo é nel caso $x = 1/2$.

Capitolo 4

Risoluzione di equazioni non lineari.

4.1 Esercizio

Calcolare con uno scarto relativo minore o uguale a $T = 10^{-3}$ il valore di ξ t.c.

$$ln(\xi) = cos(\xi) \tag{4.1}$$

Risolvere questo problema

1. determinando un intervallo in cui cade la soluzione.
2. Eseguendo una iterazione con il metodo della bisezione, in modo da ottenere una stima $\bar{x_0}$ della soluzione,
3. eseguendo un numero opportuno di iterazioni con lo schema di Newton-Raphson, dopo aver posto $x_0 = \bar{x_0}$.

Arrotondare i risultati intermedi a 5 cifre decimali.

Soluzione: schizzando le funzioni $y = \ln(x)$ e $z = \cos(x)$ si vede facilmente (cf figura) che esiste una sola soluzione dell' equazione data, che cade ad esempio nell'intervallo $I = [1, \pi/2]$.

Trovare il valore ξ che soddisfa l' equazione equivale a trovare lo zero della funzione

$$f(x) = \ln(x) - \cos(x).$$

Applicando il metodo di bisezione all' intervallo I, posto r_i = stima della soluzione all' i-esima iterazione, $e_i = |(t_i - s_i)/(t_i + s_i)|$, otteniamo:

```
i  s_i          t_i          r_i          e_i
0 1.0000e+00 1.5708e+00 1.2854e+00 2.2203e-01
1 1.2854e+00 1.5708e+00 1.4281e+00 9.9923e-02
```

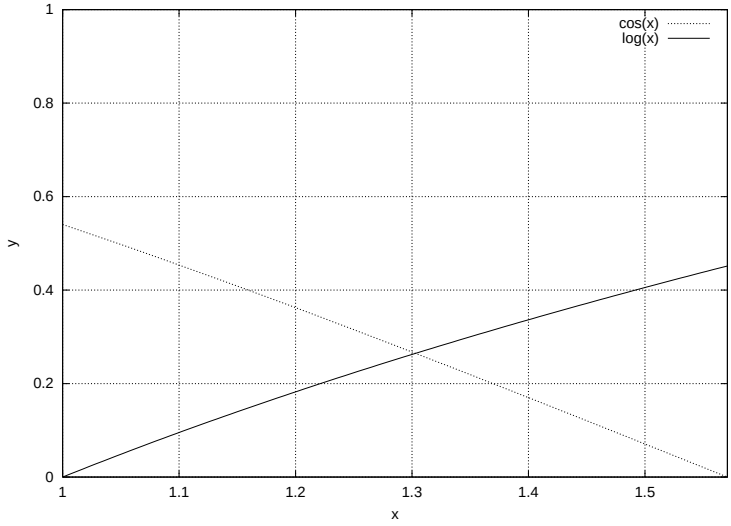


Figura 4.1: Grafico delle due funzioni.

Posto ora $x_0 = 1.4281$, $e_i = |(x_i - x_{i-1})/x_i|$ e usando lo schema di Newton-Raphson, otteniamo:

i	x_i	e_i
0	1.4281e+00	NaN
1	1.3014e+00	9.7357e-02
2	1.3030e+00	1.2279e-03
3	1.3030e+00	0

Poiché lo scarto relativo é decrescente e alla terza iterazione é minore della tolleranza richiesta, alla terza iterazione si ha l' approssimazione richiesta, che, arrotondata alla quinta cifra decimale é $x = 1.3030$.

Esercizio 4.1.1 Consideriamo lo schema del punto fisso

$$x_{n+1} = ln(1 + x_n) + 1, \quad n = 0, 1, \ldots \tag{4.2}$$
$$x_0 = 2 \tag{4.3}$$

1. Provare che questo schema converge ad un valore ξ.

2. Usando la stima a priori sull' errore, valutare il numero di iterazioni k necessario perché x_k sia affetto da un errore relativo $\epsilon_k \leq 0.003$.

3. Usando la stima a posteriori sull' errore, eseguire il numero n di iterazioni necessario perché l' errore relativo sia $\epsilon_n \leq 0.003$.

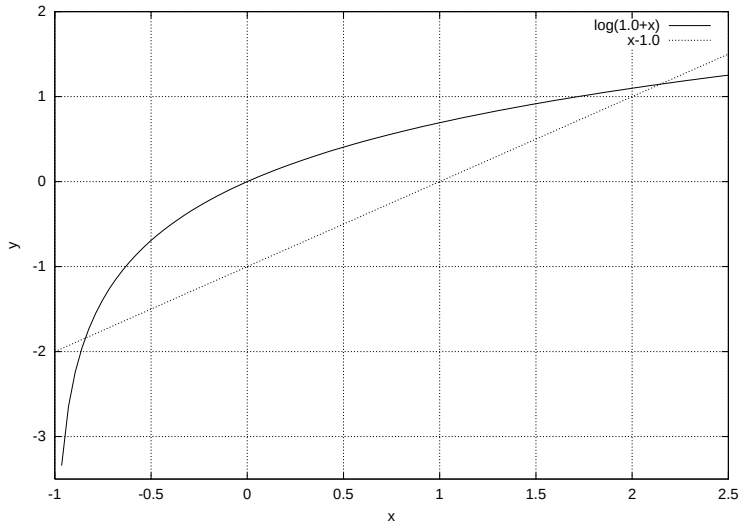


Figura 4.2: Grafico delle funzioni $y = x - 1$ e $y = \ln(1 + x)$.

Arrotondare i risultati intermedi a 5 cifre decimali.

Soluzione: Se lo schema converge, deve convergere alle soluzioni di

$$x = g(x) = ln(1 + x) + 1$$

che si vede (cf figura) sono una negativa ξ^-, e una positiva ξ.

Studiando la funzione, si puó vedere che é $2 < \xi < 2.5$. Essendo $g'(x) = 1/(1 + x)$, posto ad esempio $I = [2, 3]$ risulta

- $g(I) \subset I$
- $g(x)$ é contrattiva in I, dato che

$$m = \max_{x \in I} |g'(x)| = \frac{1}{3} < 1.$$

Dato che $x_0 \in I$, lo schema converge a ξ.

Si ha $x_0 = 2$, $x_1 = g(x_0) \simeq 2.0986$. La stima a priori sull' errore ci dice che

$$\left|\frac{\xi - x_n}{\xi}\right| \leq \left|\frac{\xi - x_n}{2}\right| \leq \frac{1}{2}\left|\frac{m^n}{1 - m}\right||x_1 - x_0| \simeq (7.3950E - 2)(1/3)^n$$

quindi deve essere

$$(7.3950E - 2)(1/3)^n \leq 0.003$$

ossia, passando ai logaritmi,

$$n \geq \frac{\ln(0.003/7.3950E-2)}{\ln(1/3)} \simeq 2.9171$$

ossia $n \geq 3$.

Utilizzando la stima a posteriori

$$\epsilon_i = \left|\frac{\xi - x_i}{\xi}\right| \leq \frac{|\xi - x_i|}{2} \leq (1/2)\frac{m}{1-m}|x_i - x_{i-1}| = \texttt{eps(i)}$$

Otteniamo i risultati

```
i  xi       eps(i)
0  2.0000     ?
1  2.0986   2.4650E-2
2  2.1310   8.0855E-3
3  2.1413   2.5959E-3
```

ed essendo $\epsilon_3 < 0.003$, ci possiamo fermare alla terza iterazione e il risultato é $\xi = 2.1413$.

Capitolo 5

Risoluzione di sistemi lineari.

5.1 Matrici e vettori

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = \mathbf{A}(m \times n)$$

$$\begin{aligned} \mathbf{A} &= (a_{ij}) = (b_{ij}) = \mathbf{B}, \\ \mathbf{C} &= (c_{ij}) = (a_{ij} + b_{ij}) = \mathbf{A} + \mathbf{B}, \\ \mathbf{P} &= (p_{ij}) = (\sum_{k=1}^{n} a_{ik} b_{kj}) = \mathbf{AB} = \mathbf{A} * \mathbf{B}. \end{aligned}$$

Il prodotto algebrico $\mathbf{AB}$ si può eseguire solo se $\mathbf{A} = \mathbf{A}(m \times n)$, $\mathbf{B} = \mathbf{B}(n \times p)$.

$$\begin{gathered} y : \mathbb{R}^n \to \mathbb{R}^n, \quad y : x \mapsto f(x), \\ z : \mathbb{R}^n \to \mathbb{R}^n, \quad z : x \mapsto g(x), \\ z = (g \circ f)(x) = g(f(x)) \end{gathered}$$

se f e g sono lineari e si fissa una base di $\mathbb{R}^n$...

$$\mathbf{y} = \mathbf{Ax}, \quad \mathbf{z} = \mathbf{By}, \quad \mathbf{z} = \mathbf{B}(\mathbf{Ax}) = \mathbf{B} \cdot \mathbf{Ax}$$

Supponiamo che le matrici siano quadrate: $m = n$.

Matrice trasposta di $\mathbf{A}$: $\mathbf{A}^T = \mathbf{A}^T_{ij} = \mathbf{A}_{ji}$. Se $\mathbf{A} = \mathbf{A}^T$ si dice che è simmetrica.

Matrice coniugata di $\mathbf{A} \in \mathbb{C}^n$: $\mathbf{A}^* = \mathbf{A}^*_{ij} = \overline{\mathbf{A}_{ij}}$.

Matrice coniugata e trasposta (trasposta coniugata) di $\mathbf{A} \in \mathbb{C}^n$: $\mathbf{A}^+ = \mathbf{A}^+_{ij} = \overline{(\mathbf{A}_{ji})} = \overline{\mathbf{A}}_{ji}$. Se $\mathbf{A} = \mathbf{A}^+$ si dice che è Hermitiana.

Matrice inversa, $\mathbf{X}$, di una matrice $\mathbf{A} = \mathbf{A}(n \times n)$:

$$\mathbf{AX} = \mathbf{XA} = \mathbf{I}, \quad \mathbf{X} = \mathbf{A}^{-1}.$$

$\mathbf{A}^{-1}$ esiste sse la matrice è non–singolare, ossia

$$det(\mathbf{A}) \neq 0$$

ovvero sse le colonne di $\mathbf{A}$ sono linearmente indipendenti, sse le righe di $\mathbf{A}$ sono linearmente indipendenti.

Il Determinante

$$det(\mathbf{A}) = \sum_{j=1}^{n} \Delta_{ij} a_{ij},$$

$$\Delta_{ij} = (-1)^{i+j} det(\mathbf{A}_{ij}),$$
$$\mathbf{A}_{ij} = \mathbf{A} \backslash \{\text{riga } i, \text{colonna } j\}.$$

$\mathbf{A}_{ij}$ = minore di $\mathbf{A}$ relativo all'elemento a_{ij}.

5.1.1 Autovettori e autovalori

$$\mathbf{A} \in \mathbb{C}^{n^2}, \lambda \in \mathbb{C}, 0 \neq \mathbf{u} \in \mathbb{C}^n$$

Definizione 5.1.1

$$\mathbf{Au} = \lambda \mathbf{u}$$

- λ è un autovalore (eigenvalue) di $\mathbf{A}$, $\lambda \in \lambda(\mathbf{A})$ = spettro di $\mathbf{A}$,
- $\mathbf{u}$ è un autovettore (eigenvector) di $\mathbf{A}$, $\mathbf{u} \in V(\mathbf{A})$ = autospazio di $\mathbf{A}$,

$$\mathbf{Au} = \lambda\mathbf{u} \iff p(\lambda) = det(\mathbf{A} - \lambda\mathbf{I}) = 0.$$

$p(\lambda)$ è il polinomio caratteristico di $\mathbf{A}$.

Definizione 5.1.2 $\mathbf{A}$ è simile a $\mathbf{B}$ sse

$$\mathbf{A} = \mathbf{S}^{-1}\mathbf{BS}.$$

$\lambda(\mathbf{A}) = \lambda(\mathbf{B})$.

Definizione 5.1.3 Raggio spettrale di una matrice,

$$\rho(\mathbf{A}) = \max_i |\lambda_i|, \quad \lambda_i \in \lambda(\mathbf{A}).$$

5.1.2 Norme

Norme di vettori

$$\|\cdot\| : \mathbf{v} \in \mathbb{C}^n \to \mathbb{R}^{\geq 0}$$

1. $\|\mathbf{v}\| \geq 0$, $\|\mathbf{v}\| = 0$ sse $\mathbf{v} = 0$;
2. $\|c\,\mathbf{v}\| = |c|\|\mathbf{v}\|$, $c \in \mathbb{C}$;
3. $\|\mathbf{v} + \mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\|$ (disuguaglianza triangolare).

Esempi:

- norma 2 o Euclidea:

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} \mathbf{v}_i^2} = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{\mathbf{v}^T\mathbf{v}}.$$

- norma p:

$$\|\mathbf{v}\|_p = (\sum_{i=1}^{n} |\mathbf{v}_i|^p)^{1/p}$$

se $p = \infty$,

$$\|\mathbf{v}\|_\infty = \max_{i=1}^{n} |\mathbf{v}_i|$$

Norme di matrici

$$\|\cdot\| : \mathbf{M} \in \mathbb{C}^{n^2} \rightarrow \mathbb{R}^{\geq 0}$$

1., 2., 3. +

4. $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$ (sub–moltiplicatività).

Esempio. Norma di Frobenius:

$$\|\mathbf{M}\|_F = \sqrt{\sum_{i,j=1}^{n} \mathbf{M}_{ij}^2} = \sqrt{Tr(\mathbf{M}\mathbf{M}^+)} = \sqrt{Tr(\mathbf{M}^+\mathbf{M})}.$$

Traccia di una matrice: $Tr(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}$.

Definizione 5.1.4 Una norma di matrice $\|\cdot\|_M$ è naturale, oppure compatibile con, oppure subordinata a, una norma di vettore $\|\cdot\|_v$ sse

$$\|\mathbf{A}\|_M = \sup_{\mathbf{v}\neq 0} \frac{\|\mathbf{A}\mathbf{v}\|_v}{\|\mathbf{v}\|_v}.$$

Norme subordinate:

- norma-1 (massima sulle colonne):

$$\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|,$$

- norma-infinito (massima sulle righe):

$$\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|,$$

- norma Euclidea (o di Hilbert):

$$\|\mathbf{A}\|_E = \sqrt{\rho(\mathbf{A}\mathbf{A}^+)}.$$

Teorema 5.1.1 *Se* $\|\mathbf{A}\|_M$ *è compatibile con* $\|\mathbf{v}\|_v$, *allora* $\|\mathbf{A}\|_M \geq \rho(\mathbf{A})$.

Matrici Speciali: vedi tabella 5.1.

Tipo matrice	Proprietá	Elementi in	Autovalori	Autovettori
Hermitiana	$H^+ = H$	$\mathcal{C}$	reali	ortonorm.*
Hermitiana e SPD	$H^+ = H$ $u^+ H\ u > 0$	$\mathcal{C}$	reali ≥ 0	ortonorm.*
Simmetrica	$H^T = H$	$\Re$	reali	ortonorm.*
Simmetrica e SPD	$H^T = H$ $u^+ H\ u > 0$	$\Re$	reali ≥ 0	ortonorm.*
Antisimmetrica	$H^+ = -H$	$\mathcal{C}$	nulli o Re $= 0$	ortonorm.*
Antisimmetrica e reale	$H^+ = -H$	$\Re$	nulli o Re $= 0$ λ_i, $\overline{\lambda_i}$	ortonorm.*
Unitaria	$U^+ U = I = U\ U^+$ $U^+ = U^{-1}$	$\mathcal{C}$	$\lvert\lambda\rvert = 1$	?
Ortogonale	$U^T\ U = I = U\ U^T$ $U^T = U^{-1}$	$\Re$	$\lvert\lambda\rvert = 1$ $\lambda = e^{\pm\ i\phi}$	?

Tabella 5.1: Alcune matrici speciali. * = Gli autovettori *possono essere scelti in modo da* essere ortonormali.

5.2 Sistemi lineari

Dato un sistema lineare

$$\begin{cases} 3x_1 - 2x_2 + x_3 &= 4 \\ x_1 + x_3 &= -7 \\ 2x_1 + 5x_2 - 3x_3 &= 2 \end{cases}$$

$$\mathbf{A} = \begin{pmatrix} 3 & -2 & 1 \\ 1 & 0 & 1 \\ 2 & 5 & -3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix},$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

La forma compatta è:

$$\boxed{\mathbf{Ax} = \mathbf{b}.}$$

5.3 Decomposizione a valori singolari

Mediante matrici ortogonali è possibile diagonalizzare una qualsiasi matrice $A \in \mathbb{R}^{m\times n}$.[1] Vale il seguente

Teorema 5.3.1 *(SVD) Sia $A \in \mathbb{R}^{m\times n}$. Esistono due matrici ortogonali $U \in \mathbb{R}^{m\times m}$ e $V \in \mathbb{R}^{n\times n}$ tali che*

$$U^T AV = \Sigma = \mathrm{diag}(\sigma_i). \tag{5.1}$$

$\Sigma \in \mathbb{R}^{m\times n}$ è una matrice diagonale con elementi σ_i tali che se r è il rango di A,

$$\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0. \tag{5.2}$$

Per la dimostrazione, si veda [Gv89]. Le quantitá σ_i vengono chiamate i *valori singolari di A*. $A^T A$ è una matrice simmetrica semi-definita positiva, in quanto per ogni vettore $x \in \mathbb{R}^n$

$$x^T \left(A^T A\right) x = \left(x^T A^T\right)(Ax) = (xA)^T (Ax) = \|Ax\|_2^2 \geq 0. \tag{5.3}$$

[1] Sezione sviluppata da M. Lizza

I suoi autovalori $\lambda_i = \lambda_i(A^TA)$ sono reali non negativi. Risultato analogo vale per la matrice AA^T. Le colonne u_i di U sono dette *vettori singolari sinistri*, le colonne v_i di V *vettori singolari destri* e rappresentano, rispettivamente, gli autovettori di AA^T e di A^TA. Risulta $\lambda_i = \sigma_i^2$. Infatti, $AA^TU = (U\Sigma V^T)(U\Sigma V^T)^TU = U\Sigma V^TV\Sigma U^TU = U\Sigma^2$ e $A^TAV = (U\Sigma V^T)^T(U\Sigma V^T)V = V\Sigma U^TU\Sigma V^TV = V\Sigma^2$.

Corollario 5.3.1 *Ogni matrice $A \in \mathbb{R}^{m\times n}$ ammette una fattorizzazione*

$$A = U\Sigma V^T, \quad \Sigma = \mathrm{diag}(\sigma_i), \tag{5.4}$$

dove $U \in \mathbb{R}^{m\times m}$ e $V \in \mathbb{R}^{n\times n}$ sono matrici ortogonali e $\Sigma \in \mathbb{R}^{m\times n}$ è una matrice diagonale tale che $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$.

Si puó dimostrare che [Gv89]

$$\|A\|_F = \left(\sigma_1^2 + \ldots + \sigma_r^2\right)^{1/2}, \quad \|A\|_2 = \sigma_1.$$

Infatti $\|A\|_F^2 = \mathrm{traccia}(A^TA) = \mathrm{traccia}(V\Sigma^2V^T) = \sum_i \lambda_i(V\Sigma^2V^T) = \sum_i \lambda_i(\Sigma^2) = \sum_i \sigma_i^2$. Inoltre $\|A\|_2^2 = \rho(A^TA) = \sigma_1^2$.

La SVD di una matrice ha interessanti proprietá, alcune condivise da altre fattorizzazioni, come la fattorizzazione QR con pivoting sulle colonne, altre peculiari della SVD.

5.3.1 Un esempio

Consideriamo la matrice quadrata

$$A = \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{vmatrix}$$

Il suo rango è 2. La decomposizione a valori singolari $A = U\Sigma V^T$ è:

$$U = \begin{vmatrix} -0.6280 & 0.3251 & -0.7071 \\ -0.4597 & -0.8881 & 0 \\ -0.6280 & 0.3251 & 0.7071 \end{vmatrix}$$

$$\Sigma = \begin{vmatrix} 2.1753 & 0 & 0 \\ 0 & 1.1260 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

$$V = \begin{vmatrix} -0.5774 & 0.5774 & -0.5774 \\ -0.2113 & -0.7887 & -0.5774 \\ -0.7887 & -0.2113 & 0.5774 \end{vmatrix}$$

Le matrici U e V sono ortonormali. Osserviamo che Σ ha due elementi non nulli, pari al rango di A. Costruiamo la matrice

$$A_1 = U_1\Sigma_1 V_1^T = |2.1753| \cdot \begin{vmatrix} -0.6280 \\ -0.4597 \\ -0.6280 \end{vmatrix} \cdot \begin{vmatrix} -0.5774 & 0.5774 & -0.5774 \end{vmatrix}$$

Risulta

$$\|A - A_1\|_2 = 1.1260 = \sigma_2.$$

Come vedremo piú avanti, questo risultato ha portata generale.

5.3.2 Spazi vettoriali

Dalle equazioni (5.1) e (5.4) si ottiene

$$\begin{cases} AV = U\Sigma \\ A^T U = V\Sigma^T \end{cases},$$

ossia

$$\begin{cases} Av_i = \sigma_i u_i \\ A^T u_i = \sigma_i v_i \end{cases}, \quad i = 1, \ldots, \min\{m, n\}. \tag{5.5}$$

Dalla (5.5) si ricava che

$$\begin{aligned} span\,\{v_{r+1}, \ldots, v_n\} &= \ker(A), & span\,\{u_{r+1}, \ldots, u_m\} &= \mathrm{range}(A)^\perp, \\ span\,\{u_1, \ldots, u_r\} &= \mathrm{range}(A), & span\,\{v_1, \ldots, v_r\} &= \ker(A)^\perp, \end{aligned}$$

dove $\ker(A) = \{x \text{ t.c. } Ax = 0\}$, $\mathrm{range}(A) = \{y \text{ t.c. } y = Ax\}$, $S^\perp = \{y \text{ t.c. } x^T y = 0\, \forall x \in S\}$.

5.3.3 Rango di matrici

La nozione di rango è di fondamentale importanza nell'algebra lineare. Determinare numericamente il rango non è un problema banale. Piccole perturbazioni lo modificano sensibilmente. Vale il seguente

Teorema 5.3.2 (Eckart-Young) *Sia $A = U\Sigma V^T$. Se $k < r = \mathrm{rank}(A)$ e*

$$A_k = \sum_{i=1}^{k} \sigma_i \cdot u_i \cdot {v_i}^T, \tag{5.6}$$

allora

$$\min_{\mathrm{rank}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}\,. \tag{5.7}$$

Per la dimostrazione, si veda [Gv89].

In altri termini, A_k è la migliore approssimazione di rango k della matrice A. Se $\sigma_k > \sigma_{k+1}$ essa è unica. Il teorema di Eckart-Young vale anche per la norma di Frobenius $\|\cdot\|_F$.

È possibile stimare l'errore relativo commesso quando la matrice A, di rango $r > k$, viene sostituita con una matrice di rango k:

$$e_k = \frac{\|A - A_k\|_2}{\|A\|_2} = \frac{\sigma_{k+1}}{\sigma_1} \tag{5.8}$$

Definizione 5.3.1 *Sia $\epsilon > 0$. L' ϵ-rango, r_ϵ, della matrice A è:*

$$r_\epsilon = \min_{\|A-B\|_2 \leq \epsilon} \text{rank}(B).$$

Dal Teorema 5.3.2 segue che r_ϵ è pari al numero di valori singolari strettamente maggiori di ϵ, ossia

$$\sigma_1 \geq \ldots \geq \sigma_{r_\epsilon} > \epsilon > \sigma_{r_\epsilon+1} = \ldots = \sigma_n = 0\,.$$

Le relazioni (5.6) valgono anche quando si mette A_r al posto di A, $r = \text{rank}(A)$.

Vediamo alcune applicazioni della decomposizione a valori singolari.

5.3.4 Sistemi sottodeterminati

Sia $A \in \mathbb{R}^{m\times n}$, $m < n$, di rango r, e sia $b \in \mathbb{R}^m$. Il sistema lineare $Ax = b$ è *sottodeterminato*: ha piú incognite che equazioni, perciò ha infinite soluzioni, oppure nessuna. Sostituendo ad A la sua SVD e ponendo $y = V^T x$, $c = U^T b$, si ottiene il sistema equivalente

$$\Sigma y = c,$$

la cui soluzione y ha componenti

$$\begin{cases} \sigma_j\, y_j = c_j, & j \leq m, \sigma_j \neq 0, \\ 0\, y_j = c_j, & j \leq m, \sigma_j = 0. \end{cases}$$

Il secondo insieme di equazioni è vuoto se $r = m$. Quindi il sistema ha soluzione se e solo se $c_j = 0$ quando $\sigma_j = 0$. Se esistono, le soluzioni y sono

$$\begin{cases} y_j = c_j/\sigma_j, & j \leq m, \sigma_j \neq 0, \\ y_j = \text{arbitrario}, & j \leq m, \sigma_j = 0, \text{ opp. } m \leq j \leq n. \end{cases}$$

Infine, le soluzioni del sistema lineare originario sono i vettori $x = Vy$.

5.3.5 Sistemi sovradeterminati

Sia $A \in \mathbb{R}^{m\times n}$, $m > n$, di rango r, e sia $b \in \mathbb{R}^m$. Il sistema lineare $Ax = b$ è *sovradeterminato*: ha piú equazioni che incognite. Procedendo come nel caso precedente, abbiamo:

$$\begin{cases} \sigma_j\, y_j = c_j, & j \le n, \sigma_j \neq 0, \\ 0\, y_j = c_j, & j \le n, \sigma_j = 0, \\ 0 = c_j, & n < j \le m. \end{cases}$$

Se $c_j = 0$ quando $\sigma_j = 0$, oppure $j > n$, esistono soluzioni y e sono

$$\begin{cases} y_j = c_j/\sigma_j, & j \le n, \sigma_j \neq 0, \\ y_j = \text{arbitrario}, & j \le n, \sigma_j = 0. \end{cases}$$

Data la norma euclidea di vettore, $\|\cdot\|_2$, calcoliamo una soluzione "alternativa" minimizzando la quantitá

$$\|r\|_2{}^2 = \|Ax - b\|_2{}^2.$$

$\|r\|_2^2$ è minima sse $\partial\|r\|_2^2/\partial x_i = 0$, $i = 1, \ldots, n$. Queste relazioni equivalgono al sistema (vedi la sezione 8.1)

$$A^T Ax = A^T b, \tag{5.9}$$

che è detto *sistema delle equazioni normali*. L'accuratezza della soluzione numerica risente molto del condizionamento di A, che peggiora in $A^T A$. Utilizzando la SVD è possibile risolvere il problema di minimizzazione senza passare per il sistema lineare (5.9). Sia $A = U\Sigma V^T$. Le matrici ortogonali preservano la norma euclidea, ossia se H è ortogonale, $\|Hv\|_2^2 = v^T H^T Hv = v^T v = \|v\|_2^2$.

$$\|r\|_2{}^2 = \|U^T \left(AVV^T x - b\right)\|_2{}^2 = \|\Sigma V^T x - U^T b\|_2{}^2$$

da cui, ponendo $y = V^T x$ e $c = U^T b$, otteniamo

$$\|r\|_2{}^2 = \|\Sigma y - c\|_2{}^2.$$

Quindi $\|r\|_2^2 = 0$ se le componenti delle soluzioni y sono

$$\begin{cases} y_j = c_j/\sigma_j, & j \le n, \sigma_j \neq 0, \\ y_j = \text{arbitrario}, & j \le n, \sigma_j = 0. \end{cases}$$

Le soluzioni di (5.9) si ottengono dalla relazione $x = Vy$. Se A non ha rango massimo, esistono infinite soluzioni. Di solito si pone $y_j = 0$ per ogni j tale che $\sigma_j = 0$, ottenendo cosí la soluzione x di norma minima, dato che $\sum_i y_i^2 = \|y\|_2^2 = \|Vy\|_2^2 = \|x\|_2^2$.

5.3.6 Filtraggio del rumore

Molti problemi nel campo dell' elaborazione del segnale utilizzano un modello di tipo vettoriale, in cui le misurazioni vengono organizzate in una matrice $A \in \mathbb{R}^{m \times n}$ [Car91]. Le colonne $a_i \in \mathbb{R}^m$ sono $a_i = s_i + r_i$, dove s_i e r_i rappresentano rispettivamente la componente del segnale e quella del rumore. Il modello assume che l'errore possa essere separato dal dato, e che la componente rumore risieda in un sottospazio ortogonale a quello del segnale.

In assenza di rumore si puø calcolare il rango di A. Purtroppo esso viene di solito sovrastimato a causa degli errori (non sistematici) di misurazione che "disturbano" i dati della matrice. Supponiamo che s_i risieda in uno spazio vettoriale di dimensione k. Calcoliamo la SVD di A. Dalle equazioni (5.6) risulta $s_i \in span\{u_1, \ldots, u_k\}, \quad r_i \in span\{u_{k+1}, \ldots, u_n\}$. Possiamo costruire la matrice A_k e stimare le due componenti (segnale e rumore) sfruttando il Teorema (5.3.2).

5.3.7 Compressione di immagini

Una immagine puó essere rappresentata mediante una matrice I *(bitmap)* di dimensioni $m \times n$ [FvDFH90]. Supponiamo $m > n$. Il generico elemento, $I_{i,j}$, definisce il colore dell'elemento di immagine (*pixel*). Per le immagini in bianco e nero $I_{i,j}$ è un valore di luminositá (tipicamente compreso nell'intervallo $[\,0, 255\,]_{\mathbb{N}}$ cosicché puø essere memorizzato in un *byte*). Figure anche semplici richiedono grandi matrici per essere rappresentate. Per memorizzare e/o trasmettere una matrice–immagine bisogna comprimerla. Gli algoritmi piu efficienti per la compressione di immagini sono quelli che comportano una certa perdita di dettaglio (*lossy algorithms*).

Utilizzando la *decomposizione a valori singolari (SVD)* [Gv89] è possibile definire una rappresentazione di dimensione ridotta che preserva la maggior parte della struttura del modello. Una volta calcolata la decomposizione a valori singolari $I = U\Sigma V^T$, si costruiscono le approssimazioni $I_k = U_k \Sigma_k {V_k}^T$. Sia $U = [u_1 \ldots u_m]$, $V = [v_1 \ldots v_n]$, $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$. Poniamo $\Sigma_k = diag(\sigma_1, \ldots, \sigma_k)$, $U_k = [u_1 \ldots u_k]$, $V_k = [v_1 \ldots v_k]$. Calcoliamo la matrice ridotta di rango k

$$I_k = U_k \Sigma_k {V_k}^T. \tag{5.10}$$

Ci chiediamo se queste approssimazioni possono rimuovere i dettagli presenti mantenendo l'"essenza" dell'immagine.

Esaminiamo due immagini. I dati relativi sono riportati nella Tabella 5.2. Nelle Tabelle 5.3 e 5.4 sono riportati i risultati delle prove effettuate. Il parametro di maggior interesse è il numero di complessivo di valori (detti anche

Immagine	Dimensione	Pixel	Rango
earth.eps	250×257	64250	250
gatlin.eps	480×640	320720	480

Tabella 5.2: Caratteristiche delle immagini considerate

k	Celle
125	79000
62	35278
32	17248
15	7830
10	5170

Tabella 5.3: Risultati quantitativi di compressione *(earth.eps)*

k	Celle
240	326400
120	148800
60	70800
30	34500
15	17025

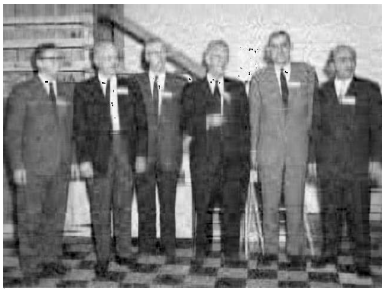
Tabella 5.4: Risultati quantitativi di compressione *(gatlin.eps)*

celle o *pixel*), necessari per memorizzare le matrici U_k, Σ_k e ${V_k}^T$. Nella Tabella 5.5 vengono mostrate alcune delle immagini ottenute con le approssimazioni. Ad esempio si vede che per k=30 si riduce di nove volte l'occupazione di memoria in *gatlin* (cf. tabelle 5.4). e 5.5) e si ha un'approssimazione di qualitá accettabile. Per dimensioni minori, la qualitá dell'immagine peggiora considerevolmente ma è ancora possibile farsi un'idea dell'aspetto "globale", soprattutto per quanto riguarda le regioni in cui compaiono linee orizzontali e/o verticali.

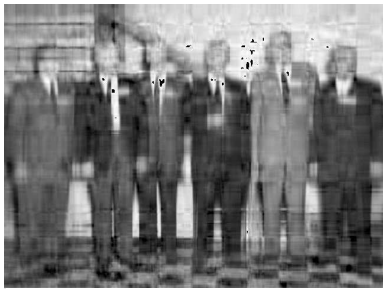
Il fattore di compressione non è paragonabile a quello offerto da algoritmi di compressione lossy come **JPEG** [Mv96]. E' confrontabile con quello offerto da **GIF** [Mv96], che perø non è lossy. Abbiamo comunque un interessante esempio di come la decomposizione a valori singolari possa essere teoricamente utilizzata per comprimere immagini.

gatlin.eps

$k = 60$



$k = 30$



$k = 15$

Tabella 5.5: Risultati qualitativi di compressione *(gatlin.eps)*

5.4 Esercizio sull' eliminazione di Gauss.

Esercizio.

Risolvere il sistema lineare

$$\begin{aligned} 10^{-2}x + y &= 1 \\ x + y &= 2 \end{aligned}$$

con il metodo di eliminazione di Gauss senza pivoting,

a) in aritmetica esatta, ottenendo cosíla soluzione $\mathbf{s}$.

b) In aritmetica a tre cifre decimali con virgola mobile normalizzata.

c) Come nel caso precedente, ma scambiando la prima riga con la seconda.

Nei casi b) e c) calcolare inoltre

$$\mathbf{e}_r = \frac{\|\mathbf{e}\|_2}{\|\mathbf{s}\|_2} \tag{5.11}$$

Caso a) I risultati sono:

```
  matrice A        vett. b
  0.01    1          1
  1       1          2
Eliminazione di Gauss
  0.01    1          1
  0     -99        -98
```

La soluzione è quindi: $\{x = 100/99, y = 98/99\}$.

Caso b) Si ottiene

```
      Matrice A         Vettore b
  1.00e-002  1.00e+000  1.00e+000
  1.00e+000  1.00e+000  2.00e+000
Eliminazione di Gauss
  1.00e-002  1.00e+000  1.00e+000
          0 -9.90e+001 -9.80e+001
Calcolo di y
  1.00e-002  1.00e+000  1.00e+000
          0  1.00e+000  9.90e-001
Calcolo di x
  1.00e+000          0  1.00e+000
          0  1.00e+000  9.90e-001
```

La soluzione è quindi: $\{x = 1.00E{+}0, y = 9.90E{-}1\}$. $\mathbf{e}_r \simeq 7.1420e{-}3$.

Caso c) Si ottiene

```
      Matrice A         Vettore b
  1.00e+000  1.00e+000  2.00e+000
  1.00e-002  1.00e+000  1.00e+000
Eliminazione di Gauss
  1.00e+000  1.00e+000  2.00e+000
          0  9.90e-001  9.80e-001
Calcolo di y
  1.00e+000  1.00e+000  2.00e+000
          0  1.00e+000  9.90e-001
Calcolo di x
  1.00e+000          0  1.01e+000
          0  1.00e+000  9.90e-001
```

La soluzione è quindi: $\{x = 1.01E{+}0, y = 9.90E{-}1\}$. $\mathbf{e}_r \simeq 1.4143e{-}4$.

Come si vede, nel caso c) il risultato é piú accurato e corrisponde ad una eliminazione con pivoting.

Capitolo 6

Costo Computazionale

Lo scopo della programmazione numerica è quello di fornire programmi accurati ed efficienti per risolvere problemi matematici.[1] L' efficienza di un codice viene valutata tramite il suo *costo computazionale*.

Il costo computazionale è volto, in ultima analisi, a "prevedere" il tempo di esecuzione degli algoritmi, senza eseguirli materialmente. Per algoritmi numerici, la stima viene data usualmente in senso asintotico [CLRS05]. Se il costo "esatto" di un algoritmo A è una complicata espressione $C(A) = f(n)$, funzione del numero di dati, n, e $f(n) = O(n)$, si dice che $C(A) = O(n)$.[2] Ad esempio, se

$$C(A) = \frac{a\,n^3 + b\,n}{g\,n + h} + d\,n^{1/2} + e\,\log(n),$$

allora $C(A) = O(n^2)$ (verificarlo per esercizio).

6.1 Operazioni

La stima piú semplice del costo computazionale di un algoritmo numerico, consiste nel valutare il numero di operazioni floating point che verranno eseguite, in funzione della quantità di dati da elaborare. In tal modo si trascurano i costi di acquisizione dei dati e produzione dei risultati, che in alcuni programmi numerici possono essere non trascurabili, ma l' idea è che in un programma *numerico*, i costi maggiori sono quelli dovuti alle computazioni,

[1] Sezione sviluppata con la collaborazione di Andrea Albarelli, Stefano Fedato, Luca Visentin.

[2] Ricordate che $f(x) = O(g(x))$ per $x \to \infty$ (si legge $f(x)$ *é di ordine "O-grande" di* $g(x)$ *per* $x \to \infty$ [Ber94]), significa che la funzione $|f(x)/g(x)|$ é limitata per $x \to \infty$, ossia esistono due costanti $M, N > 0$ t.c. $|f(x)| \leq M|g(x)|$, quando $x > N$.

Tabella 6.1: Brano di programma *prosca* che esegue il prodotto scalare di due vettori.

```
1 begin
2   s := 0;
3   for i := 1, n do
4     s := s + v(i) * w(i);
5   endfor
6 end
```

ossia in ultima analisi alle operazioni floating point (FLOP, floating point operation).

Consideriamo il brano di programma *prosca* in tabella 6.1, che calcola il prodotto scalare dei vettori **v** e **w**. Per semplicità chiamiamo "B" questo brano. Se si considera unitario il costo di una qualsiasi operazione floating point, il costo computazionale può essere stimato in base alla dimensione n dei due vettori, ed è $C(B) \simeq 2 \cdot n$ (l' algoritmo esegue n somme e n moltiplicazioni). Se si considera unitario il costo di una somma (e una sottrazione) e maggiore il costo $c > 1$ di un prodotto, il costo computazionale è $C(B) \simeq n + n \cdot c$. Se invece si considera unitario il costo di un' *operazione affine*, ossia del tipo [Nie03] $z = a * x + y$, il costo è $C(B) \simeq n$. Se si considera unitario il costo del prodotto scalare di due vettori di (valore indicativo) $m = 128$ componenti, il costo è $C(B) \simeq n/m$. Quale di questi è il costo "vero"? La risposta è che tutte queste stime sono significative, ma una è preferibile alle altre a seconda dell' architettura della macchina sulla quale si esegue l' algoritmo. Il fatto è che fino a pochi anni fa una somma veniva eseguita in un tempo molto inferiore a quello necessario per una moltiplicazione, che a sua volta era inferiore a quello speso per una divisione. Tuttavia era, ed è, pratica comune semplificare le stime assumendo che tutte le operazioni abbiano lo stesso costo. Oggi i processori hanno unità apposite per eseguire operazioni floating point, spesso di tipo *pipeline* [Sta04]. Inoltre, gli algoritmi di moltiplicazione e divisione sono stati molto migliorati. Per queste e altre ragioni, il costo di tutte le operazioni floating point può essere considerato, in pratica, uguale. Esistono inoltre processori che eseguono un' operazione affine ad ogni passo e processori che hanno un' *architettura vettoriale*, ossia che eseguono come "operazione atomica" il prodotto scalare tra due vettori di una certa dimensione. Perciò, se la macchina esegue operazioni floating point una alla volta e il costo delle operazioni è piú o meno lo stesso, per valutare il costo dell' algoritmo "B" useremo la stima $C(B) \simeq 2 \cdot n$, se invece l' architettura è vettoriale, useremo la stima $C(B) \simeq n/m$, e cosí via.

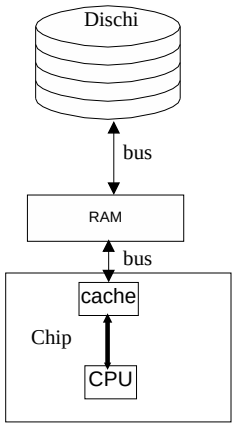


Figura 6.1: Rappresentazione schematica della gerarchia di memoria in un processore.

6.2 Accessi alla memoria

In tutte queste stime non abbiamo tenuto conto del tempo di accesso alla memoria, che è stato implicitamente pensato come trascurabile rispetto al tempo di computazione. Questa assunzione non è oggi realistica, anzi le CPU elaborano dati in un tempo molto piú basso di quello necessario per acquisire le informazioni dalla memoria [PH05].

La figura 6.1 mostra la rappresentazione schematica della gerarchia di memoria in un processore. La RAM è la memoria centrale dell' elaboratore. L' accesso ai dati su disco è molto piú lento che a quelli nella RAM, ma quando si spegne l' elaboratore i dati nella RAM vanno persi, ecco perché tutte le informazioni debbono sempre essere memorizzate su disco. Putroppo il costo per bit memorizzato nella RAM è molto piú alto che nei dischi, quindi la quantità di dati memorizzabili nel disco è molto piú grande che nella RAM; solo le informazioni che debbono essere elaborate in un dato istante vengono copiate nella RAM. Una cache è una memoria tampone ad accesso rapido, che ha dimensione inferiore alla RAM. Le cache di livello 1 (L1) sono inserite nel chip in cui risiede anche la CPU. Lo schema in figura 6.1 è una semplificazione: in realtà la gerarchia è piú complessa e vi è piú di una memoria cache anche nei processori piú comuni (vedi ad esempio la figura 6.2). Per chiarirci la necessità delle memorie cache, facciamo un esempio. Consideriamo un moderno processore a 64 bit, che lavora a 3 GHz, ossia esegue 3×10^9 operazioni al secondo. Semplificando un po', possiamo pensare che, se gli operandi sono disponibili, possa essere eseguita un' operazione floating point ad ogni cliclo, quindi 3×10^9 al secondo. Tuttavia per eseguire un' operazione

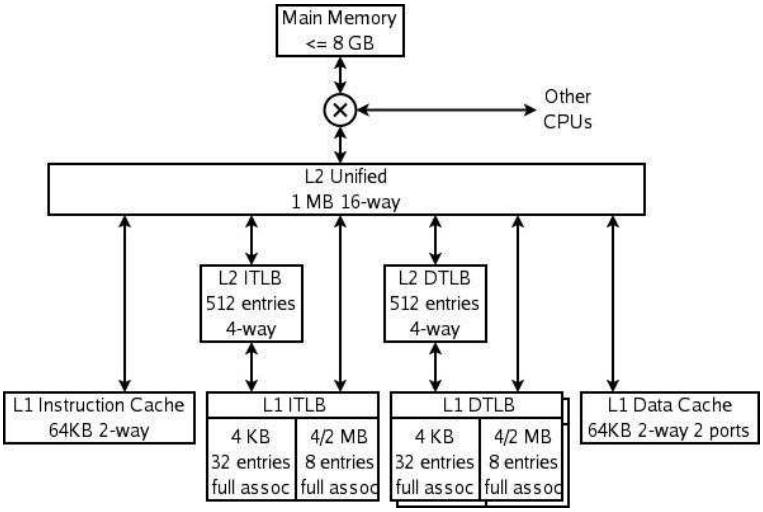


Figura 6.2: Livelli di cache in un processore AMD Athlon 64.

bisogna procurarsi i valori da elaborare, tramite i *bus*, prendendoli (in linea di principio) dalla memoria centrale, o RAM. La RAM non risiede sul chip in cui si trova la CPU. Per scambiare informazioni tra i due, bisogna usare un canale di collegamento, detto *bus*. Un moderno bus può lavorare a 333 MHz, ossia fornire 333×10^6 operandi al secondo, prelevandoli dalla memoria. A sua volta la RAM ha una velocità di accesso di 433 MHz, superiore a quella del bus. Un' operazione floating point richiede due operandi e fornisce un risultato che va inviato alla memoria. Ora: 2 operandi + 1 risultato = bisogna scambiare con la memoria 3 valori ad ogni operazione. A causa del bus, si otterrebbe quindi una performance di $333/3 = 111MHz \simeq 0.1$ GFLOPs (GFLOPs = Giga–FLOP/secondo, 10^9 operazioni al secondo), decisamente scadente!

Le cache sono inserite nel chip assieme alla CPU, la quale può accedere ai dati in essa contenuti ad una frequenza molto maggiore di quella dei bus. Ad esempio una cache di primo livello (L1) ha una frequenza di accesso tipica di 3 GHz. Se i dati risiedono in cache, si può quindi ottenere una performance di 3 GHz / 3 valori = 1 GFLOPs, molto piú soddisfacente. La dimensione della cache è piú piccola di quella della RAM, per problemi tecnologici [Sta04]. Ad esempio, a fronte di una RAM di 1 GB (1 GB $\simeq 10^9$ Byte, 1 Byte = 8 bit), le cache L1 di un comune processore possono contenere 64 KB (1 KB $\simeq 10^3$ Byte) ciascuna. Per rendere piú efficiente lo scambio, vengono messe anche cache di livelli superiori, ad esempio cache L2 da 1 MB. La figura 6.2 mostra la gerarchia delle cache di un processore AMD Athlon 64 di tipo K8. Notate

ad esempio alla base del disegno, a sinistra la cache di livello 1 (L1) per le istruzioni; a destra la cache di livello 1 per i dati. I processori piú potenti hanno cache L2 da 64 MB e L3 da 128 MB.

Le gerarchie di cache debbono essere gestite da algoritmi che, senza l' intervento dell' utente, si incaricano di far in modo che ad ogni istante dell' elaborazione, siano disponibili nella cache L1 i dati che servono alla CPU. Questi algoritimi sono di vari tipi, ogni tipo ha un nome, ad esempio *2 way associative*, oppure *full associative*, oppure *4–way*, etc. (vedi figura 6.2). Per approfondire l' argomento, si consultino [PH05, Sta04]. Naturalmente accade che le informazioni necessarie alla CPU non siano disponibili nella cache L1, in tal caso occorre trasferirli dalla cache L2; se mancano anche da questa, dalla cache L3. Ogni volta che un dato che serve non è presente in una cache, si parla di una situazione di *cache miss*. Le "cache miss" rallentano notevolmente la velocità di elaborazione. Se i dati non si trovano in alcuna cache, occorre recuperarli dalla RAM e se non si trovano nepppure lí, dal disco. Tutti i dati sono sempre memorizzati anche nel disco, perchè allo spegnimento dellla macchina i dati nelle cache e nella RAM vanno persi.

Il costo dell' operazione di copiatura dei valori da RAM a Cache e registri e viceversa non è stato considerato nel brano B. Supponiamo che il costo di ogni FLOP sia unitario e il costo *medio* di una operazione di copiatura RAM–cache–registri, χ, sia $C(\chi) = \phi \gg 1$. Allora $C(B) \simeq 2n + 4\,\phi\,n$, se ad ogni iterazione bisogna prendere i valori in `v(i)`, `w(i)` ed `s`, e poi scrivere il risultato nella variabile `s`: 4 accessi alla memoria per ognuno degli n elementi dei vettori.

Le stime attuali di costo computazionale *non* tengono conto del costo di accesso ai vari livelli di memoria, perché dipende fortemente dall' architettura e non è facile calcolarlo accuratamente. Purtroppo questo fa sí che le stime di costo computazionale non rispecchino i tempi di esecuzione dei codici.

6.3 Esempi

Supponiamo di voler eseguire un algoritmo numerico su un laptop IBM X31, dotato di processore Intel(R) Pentium(R) M a 1400MHz, con cache di 1024 KB. Usiamo il sistema operativo Fedora Core 3, con kernel Linux 2.6.9-1.667. Iniziamo scrivendo codici in ambiente GNU Octave, version 2.1.57. (vedi `http://www.octave.org`). L' ambiente Octave permette di programmare algoritmi numerici ad alto livello, ma non è adatto a elaborazioni su grandi insiemi di dati. Per brevità chiameremo "S" l' *esecutore* costituito da questa macchina, con questo sistema operativo e l' ambiente Octave.

Stimiamo il costo di un' operazione floating point. Non è sufficiente ese-

```
function [csom,cdiv] = flopTime(Nmax);
% tempi di esecuzione di operazioni floating point
  v = rand(1,Nmax); w = rand(1,Nmax); z = rand(1,Nmax);
  for oper=1:2,
    sprintf('oper=%9d',oper)
    cstv = []; count = 0; nv = [];
    n = 64
    while (n < Nmax),
      count = count + 1;
      n = 2 * n;
      if (n > Nmax), break, end
      if (oper == 1),
        [ti,ui,si] = cputime(); % octave
        for i=1:n,
          z(i) = v(i) + w(i);
        end
        [tf,uf,sf] = cputime(); % octave
      else
        [ti,ui,si] = cputime(); % octave
        for i=1:n,
          z(i) = v(i) / w(i);
        end
        [tf,uf,sf] = cputime(); % octave
      end
      tot = tf - ti; usr = uf - ui; sys = sf - si;
      cst = usr / n; cstv = [cstv cst];
      nv = [nv n];
    end
    if (oper == 1),
      csomv = cstv; csom = cst;
    else
      cdivv = cstv; cdiv = cst;
    end
  end
  loglog(nv, csomv, nv,cdivv); axis([1 Nmax 1.0e-7 1])
  return
```

Tabella 6.2: Algoritmo Octave per la stima dei tempi di esecuzione di operazioni floating point.

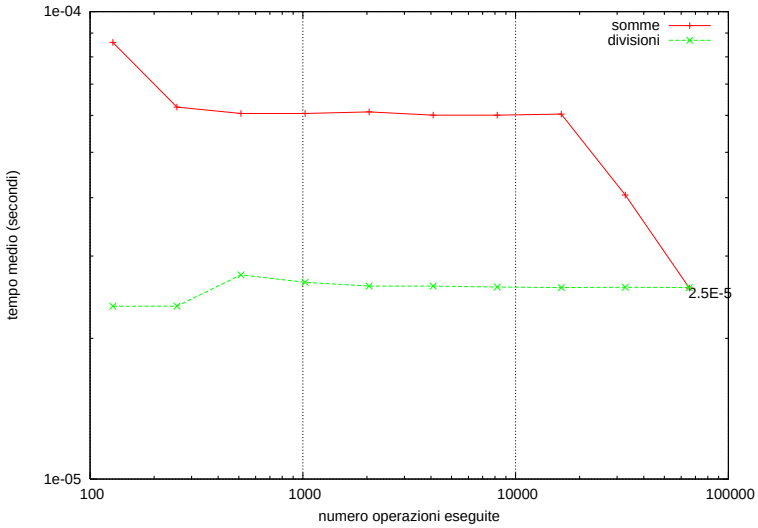


Figura 6.3: Tempo medio di esecuzione di n somme o n divisioni sul sistema "S", in Octave.

guire una sola operazione per stimare il tempo di esecuzione, perché i moderni sistemi operativi sono di tipo *time-sharing* [PH05]. La misura del tempo di CPU impiegato da un processo può essere influenzata dalla presenza di altri processi concorrenti, in particolar modo quando il tempo di esecuzione del processo che ci interessa è piccolo. La figura 6.3 mostra il tempo medio dell' esecuzione di una somma e una divisione in Octave, quando vengono eseguite $100 \leq n \leq 100000$ operazioni dello stesso tipo, usando il codice riportato in tabella 6.2. Come si vede dalla figura 6.3, i tempi diventano praticamente uguali quando il numero di operazioni è elevato. Assumiamo quindi che il tempo di esecuzione di un' operazione floating point sia $C_{\odot} \simeq$ 2.5e-5 secondi.

La figura 6.4 mostra i risultati se si usa il programma Fortran riportato nelle tabelle 6.3 e 6.4, compilato con `g77`, a sua volta basato su `gcc` versione 3.4.4. Questa volta il tempo medio di esecuzione di una somma è minore di quello di una divisione. Il calcolo del costo computazionale dovrebbe tenerne conto. Nel seguito, ci limiteremo a considerare algoritmi in Octave, per semplicità, e assumeremo uguali i tempi di esecuzione di tutte le operazioni floating point.

Sia dato un sistema lineare $Ax = b$; la matrice dei coefficienti, A sia strettamente diagonalmente dominante. Consideriamo l' algoritmo di eliminazione di Gauss senza pivoting, che calcola la matrice triangolare superiore U e il vettore $c = L^{-1}b$, tali che il sistema $LUx = b$ è equivalente a quello dato, ossia ha la stessa soluzione, e L è una matrice triangolare inferio-

```
      program flopTime
c tempi di operazioni floating point
      implicit none
c
      integer Nmax
      parameter (Nmax=10000000)
      integer oper, i
      real v(1:Nmax), w(1:Nmax), z(1:Nmax), drand
c inizializzazione casuale dei vettori v e w
      do i=1,Nmax
         v(i) = rand()
         w(i) = rand()
      end do
c calcolo dei tempi di esecuzione
      do oper=1,2
         write(6,1)'n','secondi'
         if (oper .eq. 1) then
            write(6,1)'# somme'
         else
            write(6,1)'# divisioni'
         end if
         call flopTsub(Nmax,v,w,z,oper)
      end do
c
 1    format(1x,a12,1x,a13)
 2    format(1x,i12,1x,e13.6)
      stop
      end
```

Tabella 6.3: Programma Fortran per la stima dei tempi di esecuzione di operazioni floating point.

```
      subroutine flopTsub(Nmax,v,w,z,oper)
c subroutine per il calcolo di tempi
C parametri di scambio: Nmax,v,w,z,oper in input;
      implicit none
c
      integer Nmax,n, oper, count, i
      real v(1),w(1),z(1)
      real cst, ui, uf, usr
      logical finito
c
      count = 0
      n = 2048
      finito = .false.
      do while ((n .lt. Nmax) .and. .not.(finito))
         count = count + 1
         n = 2 * n
         if (n .gt. Nmax) then
            finito = .true.
         else
            if (oper .eq. 1) then
               call second(ui)
               do i=1,n
                  z(i) = v(i) + w(i)
               end do
               call second(uf)
            else
               call second(ui)
               do i=1,n
                  z(i) = v(i) / w(i)
               end do
               call second(uf)
            end if
            usr = uf - ui
            cst = usr / n
            write(6,2)n,cst
         endif
      end do
      return
 2    format(1x,i12,1x,e13.6)
      end
```

Tabella 6.4: Sottorogramma Fortran per la stima dei tempi di esecuzione di operazioni floating point.

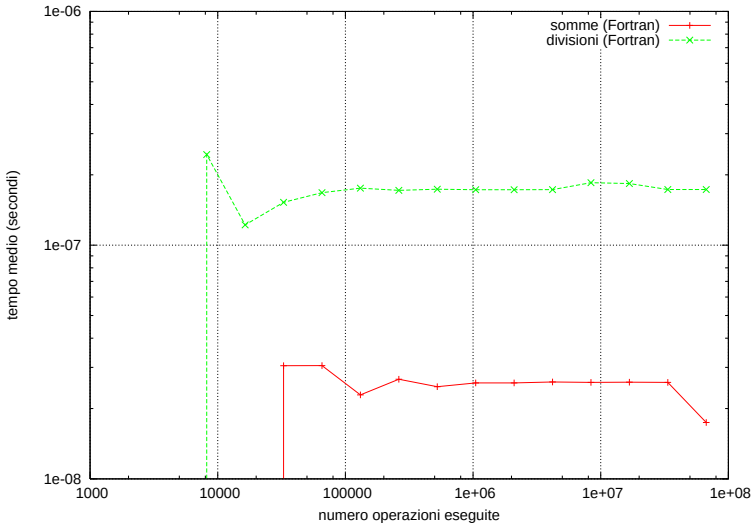


Figura 6.4: Tempo medio di esecuzione di somme o divisioni in Fortran.

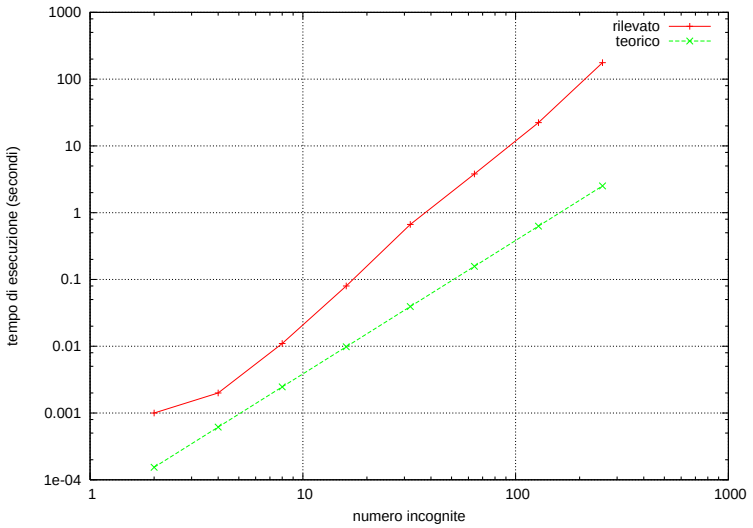


Figura 6.5: Tempo di esecuzione dell' algoritmo di eliminazione di Gauss su una matrice di ordine N.

```
function [tot,usr,sys] = gaussTime(n);
% metodo di eliminazione di Gauss, senza pivoting
% su matrice casuale
  A = rand(n);
  % modifico A per renderla strettamente diag. dom.
  for i=1:n,
    som = 0;
    for j=1:n,
      som = som + abs(A(i,j));
    end
    A(i,i) = 2 * som;
  end
  b = ((1:n)*0+1)';
  U = A;
  c = b;
  [ti,ui,si] = cputime(); % octave
  for i=1:n,
    fact = U(i,i);
    for j=i:n
      U(i,j) = U(i,j) / fact;
    end
    c(i) = c(i) / fact;
    for j=(i+1):n,
      fact = -U(j,i);
      for k=i:n
        U(j,k) = U(j,k) + fact * U(i,k);
      end
      c(j) = c(j) + fact * c(i);
    end
  end
  [tf,uf,sf] = cputime(); % octave
  tot = tf - ti;
  usr = uf - ui;
  sys = sf - si;
  return
```

Tabella 6.5: Eliminazione di Gauss senza pivoting.

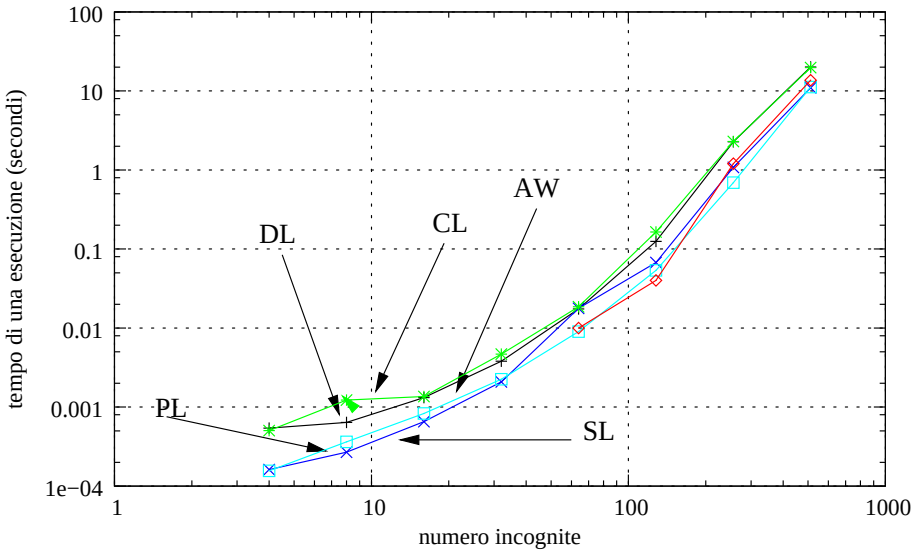


Figura 6.6: Tempo di una esecuzione dell' algoritmo di eliminazione di Gauss su varie macchine.

re [Gam94]. Sia N l' ordine della matrice; il costo asintotico dell' algoritmo di Gauss [Dem97] è $O(3N^2/2)$. La figura 6.5 mostra i tempi di esecuzione rilevati in *un solo* run del programma riportato nella tabella 6.5. Viene anche mostrata la curva $(3N^2/2)\,C_{\odot}$, che rappresenta il tempo teorico di esecuzione (asintotico) dell' algoritmo. Le due curve si distanziano all' aumentare dell' ordine della matrice: il tempo di recupero dei dati dalla memoria altera i tempi di esecuzione, allontanando i valori teorici da quelli rilevati.

La figura 6.6 mostra i tempi di una esecuzione su alcune macchine, le cui caratteristiche sono quelle fornite dai comandi "`cat /proc/cpuinfo`" e "`uname -a`" quando si usa Linux, recuperate sotto Windows con comandi opportuni. Per comodità nel seguito individueremo la macchine con una sigla, formata da due lettere:

- DL = model name: AMD Duron(tm); cpu MHz: 1000.214; cache size: 64 KB; ram size: 1024 MB; operating system: Suse Linux 10;
- SL = model name: AMD Sempron(tm)2400; cpu MHz: 1665.32; cache size: 256 KB; ram size: 1024 MB; operating system: Fedora Linux Core 4;
- CL = model name: Intel(R)Celeron(R)CPU 1.70GHz; cpu MHz: 1693.94; cache size: 128 KB; ram size: 1024 MB; operating system: Suse Linux 10;

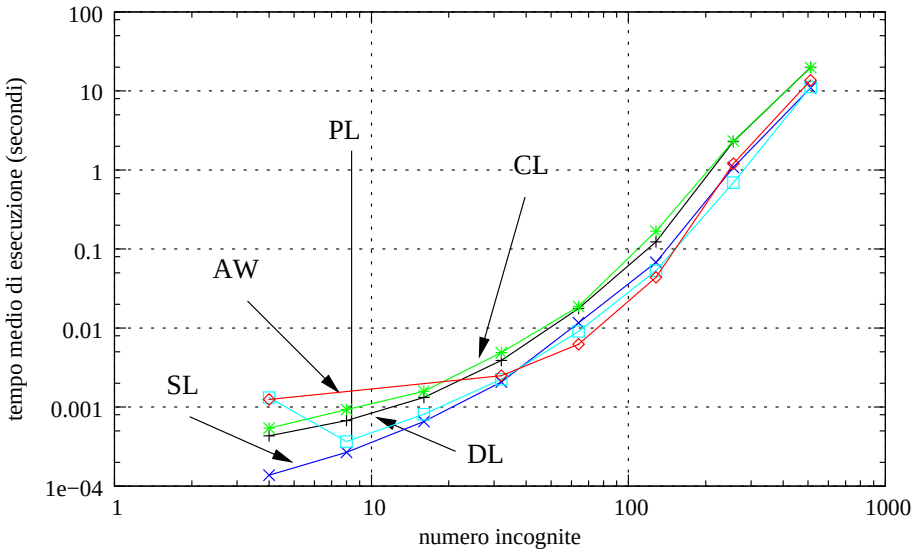


Figura 6.7: Tempo medio di esecuzione dell' algoritmo di eliminazione di Gauss su varie macchine.

- PL = model name: Dual Intel(R)Pentium(R)III CPU family 1400 MHz; cpu MHz: 1390.948; cache size: 512 KB; ram size: 1024 MB; operating system: Debian Linux;
- AW = model name: AMD Athlon XP 2000; cpu MHz: 1666; cache size: n.a.; ram size: 480 MB; operating system: Windows XP SP2.

Le curve dei tempi riportate nella figura 6.6 hanno un andamento abbastanza variabile quando la dimensione del sistema da risolvere è piccola. Ciò è sempre dovuto al fatto che nei moderni sistemi operativi time–sharing, i diversi carichi del sistema influenzano le misure dei tempi di esecuzione. Notate la maggiore variabilità delle curve nel Celeron, quando $10 < n < 100$, probabilmente dovuta alla minor quantità di cache a disposizione. Quando la dimensione è "grande", le variazioni puntuali sono smussate.

La figura 6.7 mostra i tempi medi su 8 esecuzioni. Le oscillazioni sono leggermente smussate.

Esercizio 6.3.1 Verificare che il costo computazionale del brano di codice che esegue l' eliminazione di Gauss senza pivoting è $O(3N^2/2)$.

6.4 Costo del metodo di Jacobi

Consideriamo il metodo di Jacobi (J) per risolvere il sistema lineare di ordine n

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{6.1}$$

Stimiamo il costo computazionale $C[J]$ di tale metodo, inteso come numero di operazioni floating point da effettuare, ponendo $C[\oplus] = C[\otimes] = C[\oslash] = 1$, dove $\oplus$, $\otimes$, $\oslash$ sono la somma, moltiplicazione e divisione floating point.

Lo schema di Jacobi può essere messo nella forma [Gam94, QS06]

$$\mathbf{x}_{k+1} = \mathbf{E}\mathbf{x}_k + D^{-1}\mathbf{b} = \mathbf{E}\mathbf{x}_k + \mathbf{c}, \tag{6.2}$$

La matrice di iterazione è $E = -D^{-1}(L + U)$.

Il costo di una iterazione consta di un prodotto matrice–vettore P e di una somma di vettori S. Assumendo di eseguire il prodotto P con l'algoritmo standard, il costo è di n^2 moltiplicazioni e $n^2 - n$ addizioni. Il costo di una somma di vettori è di $n - 1$ addizioni. Quindi il costo di una iterazione è $C[J(n)] = O(2n^2)]$.

Supponiamo di eseguire m iterazioni per ottenere un' approssimazione $\mathbf{x}_m$ tale che $\|\mathbf{e}_m\|/\|\mathbf{e}_0\| < \tau$, essendo τ una tolleranza prefissata, $\mathbf{e}_i$ l'errore alla i-esima iterazione. Allora $C[J(n)] = m \cdot O(2n^2)$.

Sia $\|\cdot\|$ la norma euclidea di vettori. Assumiamo che la matrice $\mathbf{A}$ sia *simmetrica*. Allora possiamo stimare m [Gam94],

$$m \geq \frac{\log \tau}{\log \rho(\mathbf{E})} = m_0.$$

Esempio. Consideriamo la matrice tridiagonale $\tilde{\mathbf{A}}$ che ha gli elementi tutti nulli, tranne

$$\tilde{\mathbf{A}}_{ij} = \begin{cases} 8, & \text{se } se \quad i = j, \\ -1, & se \quad |i - j| = 1 \end{cases}, \quad i, j = 1, \ldots, n.$$

Gli autovalori di $\tilde{\mathbf{A}}$ sono [Gam94]

$$\lambda_{\tilde{\mathbf{A}}}(k) = 8 - \frac{\sin((n-1)k\pi/(n+1))}{\sin(nk\pi/(n+1))} = 8 - \mu(k),$$
$$k = 1, \ldots, n.$$

Poniamo $\mathbf{A} = \tilde{\mathbf{A}} + 10^{-6}$. Grazie alla disuguaglianza di Wielandt-Hoffman [Par80], si ha

$$\max_j \left| \lambda_j(\tilde{\mathbf{A}}) - \lambda_j(\mathbf{A}) \right| < 10^{-6}.$$

Perciò $\rho(\mathbf{E}) = \rho(-\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})) = \rho(-\mathbf{D}^{-1}(\mathbf{A}-\mathbf{D})) = \rho((1/8)\mathbf{A} - I) \simeq \max_k |(1/8)\mu(k)| = (1/8)\max_k |\mu(k)| = (1/8)\mu(n)$. Ora

$$\lim_{n\to\infty} (1/8)|\mu(n)| = 0.25. \tag{6.3}$$

Una stima asintotica è quindi

$$m_0(\tau) = \log\tau / \log 0.25.$$

Capitolo 7

Interpolazione.

7.1 L' algoritmo di Neville.

Siano $x_0, x_1, < \ldots, x_n$ ascisse date e sia $I = [a, b]$, $a = \min_{i=0,\ldots,n}\{x_i\}$, $b = \max_{i=0,\ldots,n}\{x_i\}$, il piú piccolo intervallo che contiene le ascisse date.

L' algoritmo di Neville serve per calcolare in un punto dato $\bar{x} \in I$ i valori $P_0(\bar{x}), \ldots, P_n(\bar{x})$ dei polinomi

Polinomio	grado	punti interpolati
$P_0(\bar{x})$	0	(x_0, y_0)
$P_1(\bar{x})$	1	$(x_0, y_0), (x_1, y_1)$
$\ldots$	$\ldots$	$\ldots$
$P_n(\bar{x})$	n	$(x_0, y_0), \ldots, (x_n, y_n)$

La tecnica consiste nel costruire la tabella

$$\begin{array}{llllll} x_0 & y_0 = T_{00}(\bar{x}) & & & \\ x_1 & y_1 = T_{10}(\bar{x}) & T_{11}(\bar{x}) & & \\ x_2 & y_2 = T_{20}(\bar{x}) & T_{21}(\bar{x}) & T_{22} & \\ \ldots & \ldots & \ldots & \ldots & \\ x_n & y_n = T_{n0}(\bar{x}) & T_{n1}(\bar{x}) & \ldots & T_{nn} \end{array}$$

Gli elementi della tabella si calcolano tramite le relazioni

$$\begin{aligned} T_{i,0} &:= y_i, \quad 1 \leq k \leq i, \quad i = 0, 1, \ldots, n \\ T_{i,k} &:= \frac{1}{x_i - x_{i-k}} det \begin{pmatrix} T_{i,k-1} & \bar{x} - x_i \\ T_{i-1,k-1} & \bar{x} - x_{i-k} \end{pmatrix}. \end{aligned}$$

Si puó dimostrare che $P_i(\bar{x}) = T_{ii}(\bar{x})$.

L' algoritmo di Aitken (riportato nel testo di G. Gambolati) é una variante dell' algoritmo di Neville.

7.2 Operatori lineari.

Siano S, Z spazi di funzioni.

Un operatore é un funzionale

$$P : S \to Z. \tag{7.1}$$

Sia $\mathcal{D}(P) \subseteq S$ il dominio di P, ossia l' insieme delle funzioni per le quali P é definito e sia $\mathcal{R}(P) \subseteq Z$ la sua immagine o codominio.

Esempio: $D : C^0 \to C^0$ definito come

$$Df = \frac{df}{dx} = f' \tag{7.2}$$

é un operatore. Risulta $\mathcal{D}(D) = C^1$.

P, P_1, P_2, P_3 siano operatori t.c. $\mathcal{D}(P) \subseteq \mathcal{D}(P_1) \cap \mathcal{D}(P_2) \cap \mathcal{D}(P_3) = S \neq \emptyset$, $\mathcal{R}(P_1) \subseteq \mathcal{D}(P_2)$ e sia $f \in S$, α uno scalare. Le operazioni tra operatori si definiscono come

$$(P_1 + P_2)f = P_1 f + P_2 f \tag{7.3}$$
$$(P_2 P_1)f = P_2(P_1 f) \tag{7.4}$$
$$(\alpha P)f = \alpha(Pf) \tag{7.5}$$

Indichiamo l' identitá con 1: $1f = f$, $\forall f \in \mathcal{D}(P)$.

Se esiste un operatore P_D^{-1} t.c.

$$PP_D^{-1} = 1, \tag{7.6}$$

P_D^{-1} viene chiamato inverso destro di P.

Se esiste un operatore P_S^{-1} t.c.

$$P_S^{-1}P = 1, \tag{7.7}$$

P_S^{-1} viene chiamato inverso sinistro di P.

Si puó dimostrare che se esistono P_D^{-1} e P_S^{-1}, allora

$$P_S^{-1} = P_D^{-1} = P^{-1} \tag{7.8}$$

Definizione 7.2.1 Un operatore L é detto lineare se

- $\mathcal{D}(L)$ é uno spazio lineare,
- per tutti gli $f, g \in \mathcal{D}(L)$ e per tutti gli scalari α, β risulta

$$L(\alpha f + \beta g) = \alpha Lf + \beta Lg.$$

7.3 Interpolazione trigonometrica e FFT

$$f : [0, 2\pi] \to \mathbb{R}, \quad f(0) = f(2\pi)$$

$$f(x_k) = y_k, \quad x_k = \frac{2\pi\, k}{(N+1)}, \; k = 0, \ldots, N.$$

$$\begin{aligned} \tilde{f}(x_k) &= \frac{a_0}{2} + \sum_{k=1}^{M} (a_k \cos(kx) + b_k \sin(kx)), \\ M &= N/2 \quad (N \text{ pari}). \end{aligned}$$

$$\tilde{f}(x_k) = f(x_k), \quad k = 0, \ldots, N.$$

$$\tilde{f}(x) = \sum_{k=-M}^{M} c_k e^{ikx}, \quad i^2 = -1.$$

$$a_k = c_k + c_{-k}, \; b_k = i(c_k - c_{-k}), \quad k = 0, \ldots, M.$$

$\tilde{f}$ è detta trasformata reale discreta di Fourier (DFT, Discrete Fourier Transform).

Condizioni di interpolazione:

$$\begin{aligned} \tilde{f}(x_j) &= \sum_{k=-M}^{M} c_k e^{ikx_j} = \sum_{k=-M}^{M} c_k e^{ikjh} = f(x_j), \\ x_j &= x_0 + jh = jh, \quad k = 0, \ldots, N. \end{aligned}$$

$$\mathbf{Tc} = \mathbf{f},$$

$$\mathbf{T} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ e^{i(-M)h} & e^{i(-M+1)h} & \ldots & e^{i(+M)h} \\ e^{i(-M)2h} & e^{i(-M+1)2h} & \ldots & e^{i(+M)2h} \\ \ldots & \ldots & \ldots & \ldots \\ e^{i(-M)Nh} & e^{i(-M+1)Nh} & \ldots & e^{i(+M)Nh} \end{pmatrix}$$

$$\mathbf{c} = \begin{pmatrix} c_{-M} \\ c_{-M+1} \\ \ldots \\ c_{+M} \end{pmatrix} \quad \mathbf{f} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \ldots \\ f(x_N) \end{pmatrix}$$

$\mathbf{T}$ matrice di Toeplitz: $T_{jk} = \phi(j-k)$.

$$c_m = \frac{1}{N+1} \sum_{j=1}^{N} f(x_j) e^{-imjh}.$$

$$\mathbf{f} = \mathbf{T}\mathbf{c} = O((N+1)^2) \to O(N \log_2 N)$$

Trasformata Rapida di Fourier, Fast Fourier Transform, FFT (DFFT).

Analogamente:

$$\mathbf{c} = \mathbf{T}^{-1}\mathbf{f}$$

Inverse Fast Fourier Transform (IFFT).

Data una funzione con molte componenti in frequenza (molte armoniche). Supponiamo che la componente di frequenza massima abbia frequenza μ, allora bisogna campionare con frequenza almeno 2μ (Teorema di Shannon). Aliasing : campionando con punti "troppo lontani", si ha una cattiva approssimazione.

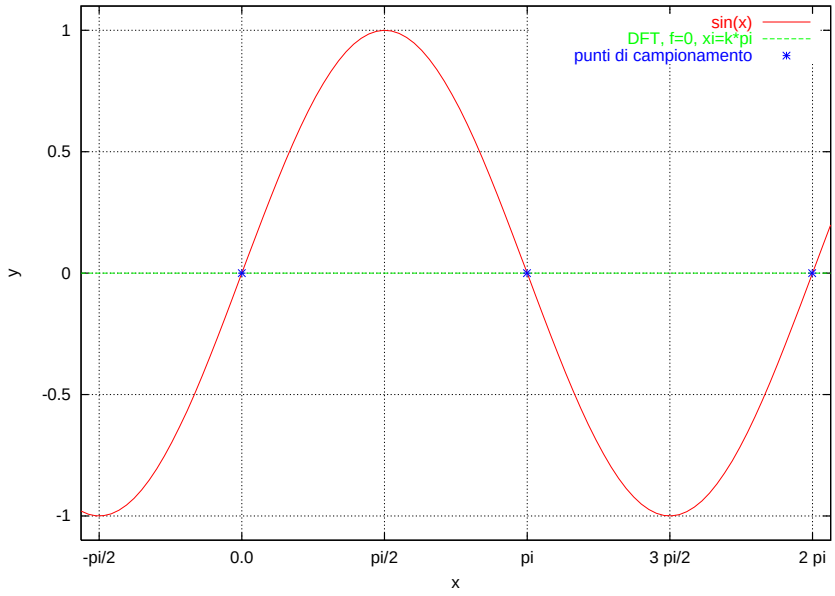


Figura 7.1: Aliasing provocato dal campionamento sui punti $x_i = k\pi$. DFT = Discrete Fourier Transform.

... vedi anche
`http://www.kostic.niu.edu/aliasing.htm` Ultimo accesso: 23 marzo 2006.

Nyquist frequency may be defined in different ways, namely:

1. maximum signal frequency that could be reliably measured to avoid aliasing = sampling frequency / 2.

2. minimum required sampling frequency to avoid aliasing = 2 * signal frequency.

Intervallo qualsiasi:

$$f : [a, b] \to \mathbb{R}, \quad f(a) = f(b)$$

$$\begin{aligned} z &= \frac{2\pi(x-a)}{(b-a)}, \quad z : [a, b] \to [0, 2\pi] \\ x &= \frac{(b-a)z}{2\pi} + a, \quad x : [0, 2\pi] \to [a, b]. \end{aligned}$$

Al posto di $\cos(k\,x)$, $\sin(k\,x)$, useremo

$$\cos\left(k\,\frac{2\pi(x-a)}{b-a}\right), \quad \sin\left(k\,\frac{2\pi(x-a)}{b-a}\right)$$

7.4 Calcolo di splines

Sia Δ una partizione dell'intervallo $[a, b]$, ossia $\Delta = \{a = x_0 < x_1 < \ldots < x_n = b\}$. Sia Y un insieme di valori reali $Y = \{y_0, \ldots, y_n\}$.

Definizione: Una funzione reale $S_\Delta : [a, b] \to \mathbb{R}$ è una *funzione spline (cubica) relativa a* Δ se

- S_Δ è almeno due volte derivabile in $[a, b]$, ossia $S_\Delta \in \mathcal{C}^2[a, b]$.
- in ogni sottointervallo $V_i = [x_i, x_{i+1}]$, $i = 0, \ldots, n-1$, S_Δ coincide con un polinomio di terzo grado:

$$S_\Delta|_{V_i} \in \mathcal{P}_3.$$

Supponiamo che Δ e Y siano fissati. e scriviamo $S(x)$ al posto di $S_\Delta(x)$.

Consideriamo i seguenti tre tipi di splines:

naturale : $S''(a) = S''(b) = 0$;

periodica : $S^{(i)}(a) = S^{(i)}(b)$, $i = 0, 1, 2$;

vincolata : $S'(a) = y'_a, \quad S'(b) = y'_b$.

Vogliamo calcolare le spline dei tre tipi, che interpolano i valori Y, ossia tali che $S(x_i) = y_i$, $i = 0, \ldots, n$.

Definiamo i *momenti* $M_j = S''(x_j)$, $j = 0, \ldots, n$, poniamo poi $h_j = x_j - x_{j-1}$, $\Delta y_j = y_j - y_{j-1}$.

Il polinomio

$$S_j(x) = S(x)|_{[x_j, x_{j+1}]}$$

si puø scrivere nel seguente modo [SB80]

$$\begin{aligned} S_j(x) = \quad & M_j \tfrac{(x_{j+1}-x)^3}{6\,h_{j+1}} + \\ M_{j+1}\frac{(x-x_j)^3}{6\,h_{j+1}} + \quad & A_j(x-x_j) + B_j. \end{aligned}$$

A_j e B_j sono date da

$$\begin{aligned} B_j = \quad & y_j - M_j \tfrac{h_{j+1}^2}{6}, \\ A_j = \quad & \tfrac{\Delta y_{j+1}}{h_{j+1}} - \tfrac{h_{j+1}}{6}(M_{j+1} - M_j). \end{aligned}$$

I momenti si possono calcolare risolvendo il sistema tridiagonale *lineare* $Tm = d$, dove

$$T = \begin{pmatrix} 2 & \lambda_0 & & & & \\ \mu_1 & 2 & \lambda_1 & & & \\ & \mu_2 & 2 & \lambda_2 & & \\ & & \ldots & \ldots & \ldots & \\ & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & \mu_n & 2 \end{pmatrix}$$

$$m = (M_0, \ldots, M_n)^T, \quad d = (d_0, \ldots, d_n)^T.$$

Le definizioni delle componenti sono

$$\lambda_j = \frac{h_{j+1}}{h_j + h_{j+1}}, \qquad \mu_j = 1 - \lambda_j, \tag{7.9}$$

$$d_j = \frac{6}{h_j + h_{j+1}} \qquad \left(\frac{\Delta y_{j+1}}{h_{j+1}} - \frac{\Delta y_j}{h_j} \right), \tag{7.10}$$

$$j = 1, \ldots, n-1. \tag{7.11}$$

Il sistema è tridiagonale: conviene risolverlo con l'algoritmo di Thomas.
Servono delle condizioni aggiuntive, che per splines
naturali, sono $\lambda_0 = d_0 = \mu_n = d_n = 0$;
vincolate, sono $\lambda_0 = \mu_n = 1$,

$$d_0 = \frac{6}{h_1}\left(\frac{\Delta y_1}{h_1} - y_0' \right), \quad d_n = -\frac{6}{h_n}\left(\frac{\Delta y_n}{h_n} - y_n' \right).$$

Le splines **periodiche** si ottengono risolvendo il sistema $T'm' = d'$, dove

$$T' = \begin{pmatrix} 2 & \lambda_1 & & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & & \\ & \mu_3 & 2 & \lambda_3 & & \\ & & \ldots & \ldots & \ldots & \\ & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & & \mu_n & 2 \end{pmatrix}$$

$$m' = (M_1, \ldots, M_n)^T, \quad d' = (d_1, \ldots, d_n)^T.$$

Valgono ancora le relazioni (7.9), (7.10), (7.11), e

$$\lambda_n = \frac{h_1}{h_1 + h_n}, \quad \mu_n = 1 - \lambda_n,$$

$$d_n = \frac{6}{h_1 + h_n} \left(\frac{y_1 - y_n}{h_1} - \frac{y_n - y_{n-1}}{h_n} \right).$$

Capitolo 8

Approssimazione.

8.1 Approssimazione di funzioni.

Il problema dell' approssimazione di funzioni si può schematizzare nel seguente modo.

E' data una funzione $f(x)$ che vogliamo analizzare nell'intervallo $I = [a, b]$. Sia $S = \{a_0\phi_0(x) + \ldots + a_n\phi_n(x)\}$, a_i sono scalari, lo spazio lineare generato dalle funzioni $\phi_0(x), \ldots, \phi_n(x)$. Si vogliono calcolare i parametri $c_0, \ldots, c_n$, della funzione $\tilde{f}(x) = c_0\phi_0(x) + \ldots + c_n\phi_n(x)$ tale che

$$d(f, \tilde{f}) = \min_{g \in S} d(f, g) \tag{8.1}$$

dove $d(\cdot, \cdot)$ è una opportuna misura di "distanza tra funzioni.

La scelta delle funzioni $\phi_i(x)$ è importante. Spesso si pone $\phi_i(x) = x^i$, in modo che S è lo spazio dei polinomi di grado minore o uguale a n e in tal caso si parla di **approssimazioni polinomiali**.

Tuttavia se è noto che la funzione da approssimare ha certe caratteristiche, sarà conveniente scegliere le funzioni $\phi_i(x)$ che meglio rappresentano queste caratteristiche. Ad esempio in alcuni casi si prende $\phi_k(x) = e^{kx}$.

La scelta della "misura di distanza è anch'essa importante e varia a seconda del tipo di modello da cui scaturisce f. Siano $x_0, \ldots, x_n$ valori dati e sia

$$I = [a, b], \quad a = min\{x_0 \ldots, x_n\}, \quad b = max\{x_0 \ldots, x_n\},$$

il piú piccolo intervallo che li contiene. Se f è nota nei punti $x_0, \ldots, x_n$, ossia sono noti i valori $y_i = f(x_i)$, $i = 1, \ldots, n$, una scelta é

$$d_1(f, g) = \sum_{i=0}^{n} |f(x_i) - g(x_i)|. \tag{8.2}$$

In tal caso, se $\phi_i(x) = x^i$ e S è lo spazio dei polinomi di grado minore o uguale a n, si parla di problema della **interpolazione polinomiale**. Si dimostra che il polinomio $\tilde{f}(x)$ che rende minima la distanza (8.2) è unico ed è il polinomio per cui $d_1(f, \tilde{f}) = 0$, ossia $\tilde{f}(x_i) = y_i$, $i = 1, \ldots, n$.

Vengono definite anche distanze in cui entrano le derivate della funzione, ad esempio

$$d_1^{(1)}(f, g) = \sum_{i=0}^{n} (|f(x_i) - g(x_i)| + |f'(x_i) - g'(x_i)|). \tag{8.3}$$

e in tal caso si parla di interpolazione di Hermite.

Se si usa la distanza d_1 ma al posto dei polinomi si usano funzioni razionali del tipo

$$R_{n,m}(x) = \frac{P_n(x)}{Q_m(x)} = \frac{a_0 + \ldots + a_n x^n}{b_0 + \ldots + b_m x^m} \tag{8.4}$$

abbiamo la cosiddetta **approssimazione di Padé**. In questo caso non è detto che vi sia sempre soluzione al problema, ossia che esista una funzione $R_{n,m}(x)$ tale che

$$R_{n,m}(x_i) = y_i, \quad i = 0, 1, \ldots, n + m \tag{8.5}$$

Se questo problema ha soluzione, la funzione che si ottiene é particolarmente efficace nel rappresentare funzioni che hanno delle singolaritá, che sono invece mal approssimate da polinomi.

Un altro caso importante è quello della **interpolazione polinomiale a tratti**: mostreremo che non è conveniente interpolare punti con polinomi di grado elevato. Se i punti sono molti, conviene interpolarli (ossia calcolare la funzione che rende uguale a zero la distanza (8.2)) con polinomi *continui a tratti*, ossia funzioni che sono polinomi opportunamente "raccordati per dare vita a una funzione definita in tutto l' intervallo I.

Se $d = d_2$, dove

$$d_2(f, g) = \sum_{i=0}^{n} (f(x_i) - g(x_i))^2, \tag{8.6}$$

il problema di calcolare la funzione g che minimizza d_2 in uno spazio di dimensione minore di n, viene detto **approssimazione discreta ai minimi quadrati**. Se S è uno spazio di polinomi, viene detta approssimazione **polinomiale** discreta ai minimi quadrati. Notare che se $dim(S) = n$, allora la soluzione è quella per la quale $d_2(f, g) = 0$, ossia il polinomio *interpolatore*!

Se $d = d_2$, ma

$$S = \{1, \cos(x), \sin(x), \cos(2x), \sin(2x), \ldots, \cos(kx), \sin(kx)\},$$

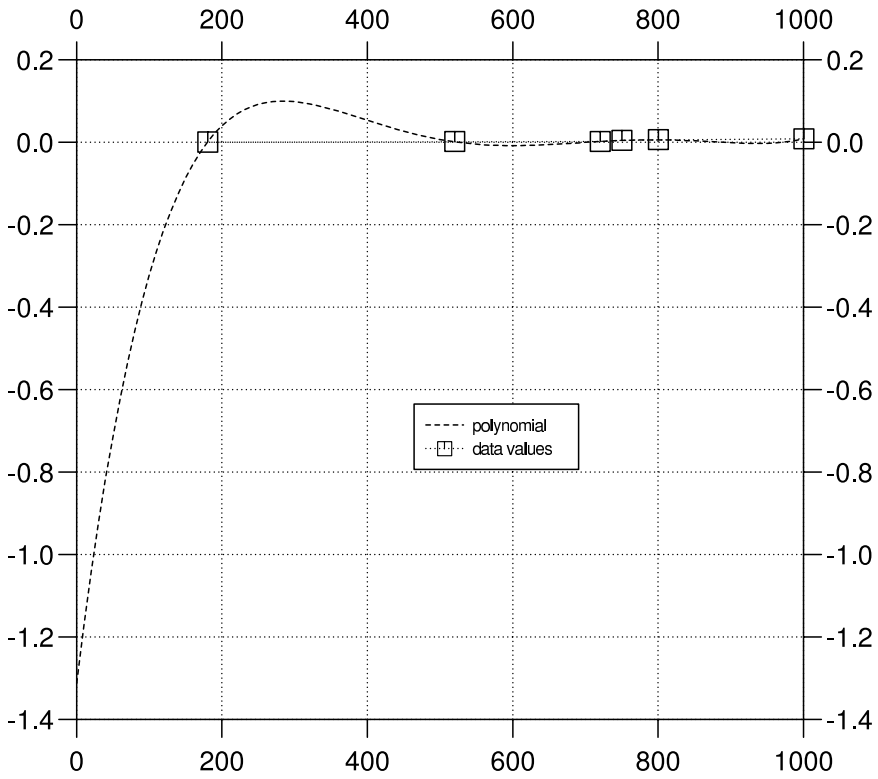


Figura 8.1: I polinomi interpolatori di grado elevato presentano forti oscillazioni.

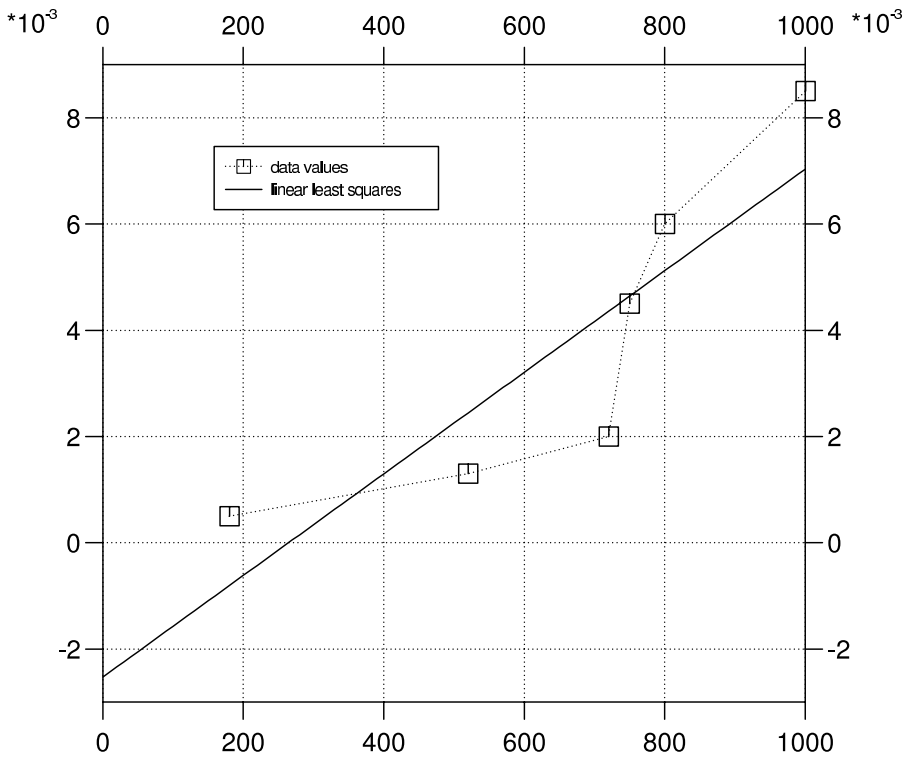


Figura 8.2: Esempio di approssimazione ai minimi quadrati.

si parla di approssimazione **trigonometrica** discreta ai minimi quadrati.

Se $d = d_2^{(c)}$, dove

$$d_2^{(c)} = \int_a^b (f(x) - g(x))^2 dx \tag{8.7}$$

si parla di approssimazione **continua** ai minimi quadrati.

Infine se $d = d_\infty$ dove

$$d_\infty(f, g) = \sup_{x \in [a,b]} |f(x) - g(x)|, \tag{8.8}$$

si parla di **approssimazione uniforme** o **approssimazione di Chebychev** in $[a, b]$. Se S è uno spazio di polinomi, si parla di approssimazione **polinomiale** uniforme e la soluzione è detta *polinomio di Chebychev*.

Il problema dell' approssimazione di Chebychev è in generale piú difficile da risolvere di quello dell' approsimazione ai minimi quadrati, sia nel caso continuo che in quello discreto. In teoria l' approssimazione di Chebychev sarebbe preferibile perché fornisce una stima *buona su tutto l' intervallo*, mentre la stima con la distanza $d_2^{(c)}$ può fornire una soluzione $\tilde{f}_2^{(c)}(x)$ che differisce molto da $f(x)$ in un ristretto sottointervallo di $[a, b]$.

Capitolo 9

Integrazione numerica.

Esercizio 9.0.1 Calcolare le approssimazioni del valore di

$$I = \int_0^1 x^2 dx \tag{9.1}$$

che si ottengono usando gli schemi di integrazione

1. dei trapezi semplice,
2. dei trapezi composta, con $n = 2$ suddivisioni.
3. di Gauss-Legendre a tre punti.

Calcolare le stime dei valori assoluti degli errori commessi, chiamiamoli rispettivamente E_1, E_2 e E_3, usando le stime teoriche e confrontarli con gli errori effettivamente generati.

Soluzione.
Usando la regola dei trapezi semplice abbiamo:

$$I \simeq 1\frac{0+1}{2} = 1/2.$$

Usando la regola dei trapezi composta, con $n = 2$ risulta:

$$I \simeq \frac{1}{2}(\frac{0+1}{2} + \frac{1}{4}) = 3/8.$$

Poiché l' intervallo di integrazione non é $[-1, 1]$, non possiamo applicare direttamente la formula di Gauss-Legendre, tuttavia ponendo $x = \frac{1}{2}y + \frac{1}{2}$, abbiamo

$$I = \frac{1}{8}\int_{-1}^1 (y+1)^2 dx.$$

n	pesi	ascisse
1	2	0
2	1	±0.57735 02692
3	5/9	±0.77459 66692
	8/9	0
4	0.34875 48451	±0.86113 63116
	0.65214 51549	±0.33998 10436
5	0.23692 68851	±0.90617 98459
	0.47862 86705	±0.53846 93101
	0.56888 88889	0

Tabella 9.1: Pesi ed ascisse per lo schema di integrazione di Gauss-Legendre.

Consultando una tabella con i pesi e le ascisse di Gauss-Legendre e usando lo schema a 3 punti otteniamo

$$I = \frac{1}{8}(0.5556(-0.7746+1)^2 + 0.8889(0+1) + 0.5556(0.7746+1)^2) \simeq 0.3333.$$

Per quanto riguarda gli errori, risulta

$$E_1 \leq \left| \frac{(b-a)^3}{12} f''(\xi) \right| = 1/6,$$

$$E_2 \leq \left| \frac{(b-a)^3}{122^2} f''(\xi) \right| = 1/24,$$

e infine

$$E_3 = 0$$

perché $f^{(5)}(x) = 0$.

I valori esatti degli errori sono uguali a quelli stimati.

9.1 Estrapolazione di Romberg

$$T(h) = \frac{b-a}{n} \left(\frac{f_0 + f_n}{2} + \sum_{i=1}^{n} f_i \right) \tag{9.2}$$

$$f \in \mathcal{C}^{m+2}[a,b]$$

$$T(h) = \int_a^b f(x)dx + \tau_1 h^2 + \tau_2 h^4 + \ldots + \tau_m h^{2m} + \alpha_{m+1}(h)h^{2m+2}$$

- $|\alpha_{m+1}(h)| \leq M_{m+1}$ per ogni $h = (b-a)/n$.
- τ_k non dipende da h.

$$T(0) = \int_a^b f(x)dx \tag{9.3}$$

$$h_0 = b - a, \quad h_1 = h_0/n_1, \quad h_2 = h_0/n_2, \ldots \tag{9.4}$$

$n_1 < n_2 < n_3 < \ldots$

$$T_{i,0} = T(h_i), \quad i = 1, \ldots, m$$

$$\tilde{T}_{m,m}(h) = a_0 + a_1 h^2 + a_2 h^4 + \ldots + a_m h^{2m}$$

polinomio interpolatore

$$\tilde{T}_{m,m}(h_i) = T(h_i), \quad i = 0, \ldots, m$$

Estrapolare: $\tilde{T}_{m,m}(0) \simeq T(0)$.
Integrazione di Romberg: $h_i := \frac{b-a}{2^i}$.
Neville:

$$\begin{array}{llll} h_0 & T(h_0) = T_{0,0} & & \\ h_1 & T(h_1) = T_{1,0} & T_{1,1} & \\ h_2 & T(h_2) = T_{2,0} & T_{2,1} & T_{2,2} \\ & \ldots & & \end{array}$$

$$\begin{aligned} T_{i,0} &= T(h_i), \\ T_{i,k} &= T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left[\frac{h_{i-k}}{h_i}\right]^2 - 1} \end{aligned}$$

Può essere che estrapolazione = "vecchie formule". Esempio:
$h_0 = b - a$, $h_1 = (b-a)/2$.
$T_{0,0} = \frac{b-a}{2}(f(a) + f(b)) \leftarrow$ trapezi
$T_{1,1} = \frac{b-a}{2}\frac{1}{3}(f(a) + 4f((a+b)/2) + f(b)) \leftarrow$ Simpson

9.2 Integrazione di Gauss

$$I(f) := \int_a^b f(x)\omega(x)dx$$

- $\omega(x) \geq 0$ misurabile su $[a, b]$.
- Esistono tutti i momenti $\mu_k = \int_a^b x^k \omega(x)dx$, $k = 0, 1, \ldots$.
- $\int_a^b \omega(x)dx > 0$.

Calcolare

$$I(f) \simeq \tilde{I}(f) := \sum_{i=1}^{n} w_i f(x_i)$$

... in modo che l' uguaglianza sia esatta fino al maggior grado possibile.
$\mathcal{P}_j :=$ { polinomi di grado $\leq$ j }
$\bar{\mathcal{P}}_j :=$ { polinomi monoici di grado $\leq$ j }
Prodotto scalare

$$(f, g) = \int_a^b f(x)g(x)\omega(x)dx$$

$L^2[a, b] =$ insieme delle funzioni "a quadrato sommabile.
$(f, g) = 0 :$ f e g sono ortogonali.

Teorema. Per ogni $k \in \mathcal{N}$, esistono polinomi $p_j \in \mathcal{P}_k$, $j = 0, 1, \ldots, n$, tali che

$$(p_i, p_j) = \delta_{ij}$$

Questi polinomi sono:

$$p_0(x) = 1, \quad p_{-1}(x) = 0,$$
$$p_{i+1}(x) = (x - \delta_{i+1})p_i(x) - \gamma_{i+1}^2 p_{i-1}(x), \quad i \geq 0$$

- $\delta_{i+1} := (x\, p_i, p_i)/(p_i, p_i)$, $i \geq 0$.
- $\gamma_{i+1}^2 := \begin{cases} 0, & \text{se } i = 0, \\ (p_i, p_i)/(p_{i-1}, p_{i-1}), & i \geq 1 \end{cases}$

Dato che se $p \in \mathcal{P}_j$ allora $p = \sum_{i=0}^{j} c_i\, p_i$, risulta
$(p, p_k) = 0$ per ogni $p \in \mathcal{P}_{k-1}$.

Teorema. Le radici x_i, $i = 1, \ldots, n$ di p_n sono

- reali e semplici,
- contenute nell' intervallo $[a, b]$.

Teorema. La matrice

$$A := \begin{pmatrix} p_0(t_1) & \ldots & p_0(t_n) \\ p_{n-1}(t_1) & \ldots & p_{n-1}(t_n) \end{pmatrix}$$

è non singolare se i punti t_i sono distinti.

Quindi il sistema lineare

$$A^T \begin{pmatrix} \_0 \\ c_1 \\ \ldots \\ c_{n-1} \end{pmatrix} = f_i$$

ha una e una sola soluzione, ossia esiste $p(x)$ t.c. $p(x) = \sum_{i=0}^{n-1} c_i f_i(x)$, $p(t_i) = f_i$.

A viene chiamata *Matrice di Haar* e

$det(a) \neq 0$ viene chiamata *condizione di Haar*.

$p_0, \ldots, p_n$ soddisfano la condizione di Haar: si dice che formano un *sistema di Chebychev*.

Teorema. Siano $p_0, \ldots, p_n$ polinomi ortogonali. Siano $x_1, \ldots, x_n$ le radici di p_n e sia $w = (w_1, \ldots, w_n)^T$ la soluzione del sistema lineare

$$\begin{pmatrix} p_0(x_1) & \ldots & p_0(x_n) \\ p_{n-1}(x_1) & \ldots & p_{n-1}(x_n) \end{pmatrix} \begin{pmatrix} w_1 \\ \ldots \\ w_n \end{pmatrix} \begin{pmatrix} (p_0, p_0) \\ 0 \\ 0 \end{pmatrix} \tag{9.5}$$

allora i valori w_i (detti *pesi*) sono maggiori di zero e

$$\begin{array}{ll} \int_a^b f(x)\omega(x)dx & = \sum_{i=1}^n w_i p(x_i) \\ & \text{per ogni } p \in \mathcal{P}_{2n-1}. \end{array} \tag{9.6}$$

Viceversa: se w_i, x_i soddisfano (9.6) allora $p_n(x_i) = 0$ e $w = (w_1, \ldots, w_n)$ soddisfa (9.5).

NON esistono $2n$ numeri w_i, x_i tali che (9.6) valga per ogni $p \in \mathcal{P}_{2n}$.

Esempio 9.2.1 $\omega(x) = 1$, intervallo $= [-1, 1]$:

polinomi di Legendre

formula ricorrente:

$$p_{-1} = 0, \quad p_0 = 1$$

$$p_{n+1}(x) = \frac{2n+1}{n+1}\, x\, p_n(x) - \frac{n}{n+1}\, p_{n-1}(x)$$

definizione alternativa:

$$p_k(x) = \frac{k!}{(2k)!}\frac{d^k}{dx^k}(x^2-1)^k$$

Come si calcolano i w_i e gli x_i?

$$J = \begin{pmatrix} \delta_1 & \gamma_1 & 0 & & & \\ \gamma_1 & \delta_2 & \gamma_2 & 0 & & \\ & & \cdots & & & \\ 0 & & & \gamma_{n-2} & \delta_{n-1} & \gamma_{n-1} \\ & 0 & & 0 & \gamma_n & \delta_n \end{pmatrix} \tag{9.7}$$

$$det(J - xI) = p_n(x)$$

$$p_n(x) = 0 \text{ sse } x \in \lambda(J)$$

I w_i si possono calcolare a partire da $V(J)$.

Teorema. Sia $f \in \mathcal{C}^{2n}[a, b]$,
esiste $\xi \in [a, b]$ tale che

$$\int_a^b f(x)\omega(x)dx - \sum_{i=1}^n w_i f(x_i) = \frac{f^{(2n)}(\xi)}{(2n)!}(p_n, p_n)$$

Capitolo 10

Equazioni differenziali ordinarie.

10.1 Cenni sulla risolubilitá delle ODE.

Si chiama *problema di Cauchy* il problema di trovare una funzione y continua e derivabile nell' intervallo $I_0 = [x_0, x_1]$ tale che

$$y'(x) = f(x, y(x)), \quad x \in I_0 \tag{10.1}$$
$$y(x_0) = y_0 \tag{10.2}$$

La condizione (10.2) si chiama **condizione di Cauchy** e anche **condizione iniziale**.

Una funzione y che verifica la (10.1) viene chiamata un **integrale** dell' equazione differenziale.

La equazione (10.1) viene detta **del primo ordine** perché in essa interviene solo la derivata prima di y.

Vale il

Teorema (Cauchy-Lipschitz): se la funzione f é continua su $I_0 \times \Re$ ed esiste una costante reale positiva L tale che

$$|f(x, y) - f(x, z)| \leq L|y - z|, \tag{10.3}$$

per tutte le coppie (x, y), $(x, z) \in I_0 \times \Re$, allora il problema (10.1)+(10.2) ammette una ed una sola soluzione.

Per poter risolvere numericamente un problema differenziale, non basta che la soluzione esista e sia unica, bisogna anche che essa sia ben condizionata. Per studiare il condizionamento del problema, dobbiamo studiare il

comportamento della soluzione quando i dati cambiano. A questo proposito, sia $\psi(x)$ una funzione continua in I_0, β ed ϵ numeri reali. Come problema *perturbato* consideriamo il seguente:

trovare una funzione $y_\epsilon \in C^1(I_0)$ tale che

$$y'_\epsilon(x) = f(x, y_\epsilon(x)) + \epsilon\psi(x), \quad x \in I_0 \tag{10.4}$$
$$y_\epsilon(x_0) = y_0 + \epsilon\beta \tag{10.5}$$

Vale la

Proposizione: se valgono le condizioni del teorema di Cauchy-Lipschitz, allora il problema (10.4)+(10.5) ammette una ed una sola soluzione e per tutti gli $x \in I_0$ risulta

$$|y(x) - y_\epsilon(x)| \leq |\epsilon\beta| e^{L|x-x_0|} + \int_{x_0}^{x} |\epsilon\psi(s)| ds \tag{10.6}$$

La relazione (10.6) implica la *dipendenza continua* della soluzione dai dati, una caratteristica molto importante dal punto di vista della risolubilitá numerica.

Si dice che un problema é **ben posto** se la sua soluzione *esiste*, *é unica* e *dipende con continuitá dai dati*. Un problema differenziale é numericamente risolubile se é non solo ben posto, ma anche *ben condizionato*. La relazione (10.6) mostra che il problema in esame é ben condizionato quando la quantitá

$$\max_{x \in I_0} e^{L|x-x_0|} \tag{10.7}$$

non é "troppo grande.

Consideriamo infine il problema della *regolaritá*, ossia del grado di derivabilitá, della soluzione in dipendenza dalla regolaritá dei dati. L' ordine di convergenza dei metodi numerici dipende infatti dalla regolaritá della soluzione. Vale il

Teorema: sia $y \in C^1(I_0)$ una soluzione del problema (10.1)+(10.2). Se f é derivabile p volte nel punto $(x, y(x))$, allora nello stesso punto la soluzione y é derivabile $(p+1)$ volte e si ha

$$y^{(p+1)}(x) = f^{(p)}(x, y(x)).$$

Molte interessanti considerazioni sulle ODE e sui metodi per la loro integrazione numerica si trovano in [EEHJ96].

Esercizio 10.1.1 Si vuole integrare numericamente il problema differenziale

$$\begin{aligned} y' + y &= 0, \quad x \in [0,1] && (10.8)\\ y(0) &= y_0 = 1 && (10.9) \end{aligned}$$

Calcolare le soluzioni approssimate $\bar{y}^{(1)}$ e $\bar{y}^{(2)}$ che si ottengono usando rispettivamente gli schemi alle differenze:

$$(1+h)\bar{y}_{n+1} - \bar{y}_n = 0 \tag{10.10}$$

$$\bar{y}_{n+2} - 4\bar{y}_{n+1} + (3-2h)\bar{y}_n = 0 \tag{10.11}$$

con passo $h = 1/2$.

Calcolare le norme 1 dei vettori degli errori relativi che si commettono utilizzando i due schemi.

Soluzione.

Tramite le relazioni ricorrenti, é immediato calcolare le approssimazioni richieste. L' unico problema é il valore di $\bar{y}_1^{(2)}$. Poniamo $\bar{y}_1^{(2)} := \bar{y}_1^{(1)}$. Otteniamo cosí:

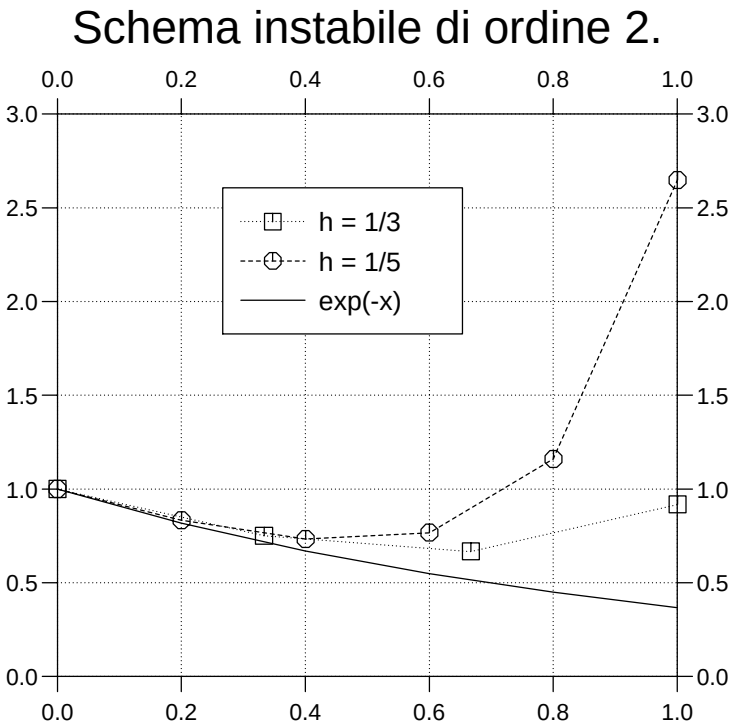
x	exp(-x)	y1	y2
0	1.0000	1.0000	1.0000
0.5000	0.6065	0.6667	0.6667
1.0000	0.3679	0.4444	0.6667

dove y1 := $\bar{y}^{(1)}$, y2 := $\bar{y}^{(2)}$.

La soluzione esatta del problema é $y(x) = exp(-x)$. Le norme 1 dei vettori degli errori relativi

$$\left\| e^{(j)} \right\|_1 = \sum_{i=0}^{2} \left| \frac{y(x_i) - \bar{y}_i^{(j)}}{y(x_i)} \right|$$

sono quindi: $\left\| e^{(1)} \right\|_1 \simeq 0.3072$, $\left\| e^{(2)} \right\|_1 \simeq 0.9114$.



Come si vede, l' errore relativo generato con lo schema del secondo ordine é piú elevato, per il fatto che la soluzione ha una componente spuria che cresce rapidamente. E' interessante vedere che cosa accade quando si usa tale schema e si diminuisce il passo di integrazione. In figura si puó vedere il comportamento della soluzione esatta e di quelle approssimate ottenute con passi $h = 1/3$ e $h = 1/5$: le soluzioni dell' equazione alle differenze crescono molto rapidamente e non convergono alla soluzione esatta.

Esercizio 10.1.2 Approssimare la soluzione del problema differenziale

$$y' + y = x^2, \quad x \in [0, 1] \tag{10.12}$$
$$y(0) = y_0 = 3 \tag{10.13}$$

utilizzando lo schema di Eulero.

Analizzare la stabilitá e la convergenza dello schema alle differenze che si ottiene.

Soluzione.

Lo schema di Eulero é

$$\bar{y}_{n+1} = \bar{y}_n + hf(x_n, y_n).$$

In questo caso abbiamo il problema:

$$y' = f(x, y) = -y + x^2,$$

l' equazione alle differenze risultante é

$$\bar{y}_{n+1} = \bar{y}_n + h(-\bar{y}_n + x_n^2),$$

ossia, dato che $x_n = x_0 + nh = nh$:

$$\bar{y}_{n+1} - (1 - h)\bar{y}_n = n^2 h^3.$$

La soluzione generale di questa equazione alle differenze é

$$\bar{y}_n^* = c(1 - h)^n + (hn)^2 - 2(hn) + 2 - h,$$

che, tenendo conto delle condizioni iniziali, si particolarizza in

$$\bar{y}_n = (1 + h)(1 - h)^n + (hn)^2 - 2(hn) + 2 - h.$$

La soluzione del problema differenziale é invece

$$y = e^{-x} + x^2 - 2x + 2.$$

E' facile vedere che per $h \to 0$ e $n \to \infty$ in modo che $nh = x$= cost., si ha

$$\lim_{h\to 0}(1 + h)(1 - h)^n = e^{-x}, \quad \lim_{h\to 0}\left[(hn)^2 - 2(hn) + 2 - h\right] = x^2 - 2x + 2,$$

ossia lo schema é convergente:

$$\lim_{h\to 0} \bar{y}_n = y(x).$$

Per quanto riguarda la stabilitá, se $\bar{y}_k$ é affetto da un errore ϵ_k, abbiamo:

$$\bar{y}_{n+1} + \epsilon_{n+1} - (1 - h)(\bar{y}_n + \epsilon_n) = n^2 h^3,$$

da cui l' errore soddisfa alla relazione

$$\epsilon_{n+1} - (1 - h)\epsilon_n = 0.$$

Quindi

$$\epsilon_n = \epsilon_0(1 - h)^n,$$

e lo schema é stabile a condizione che $|1 - h| \leq 1$, cioé $0 \leq h \leq 2$.

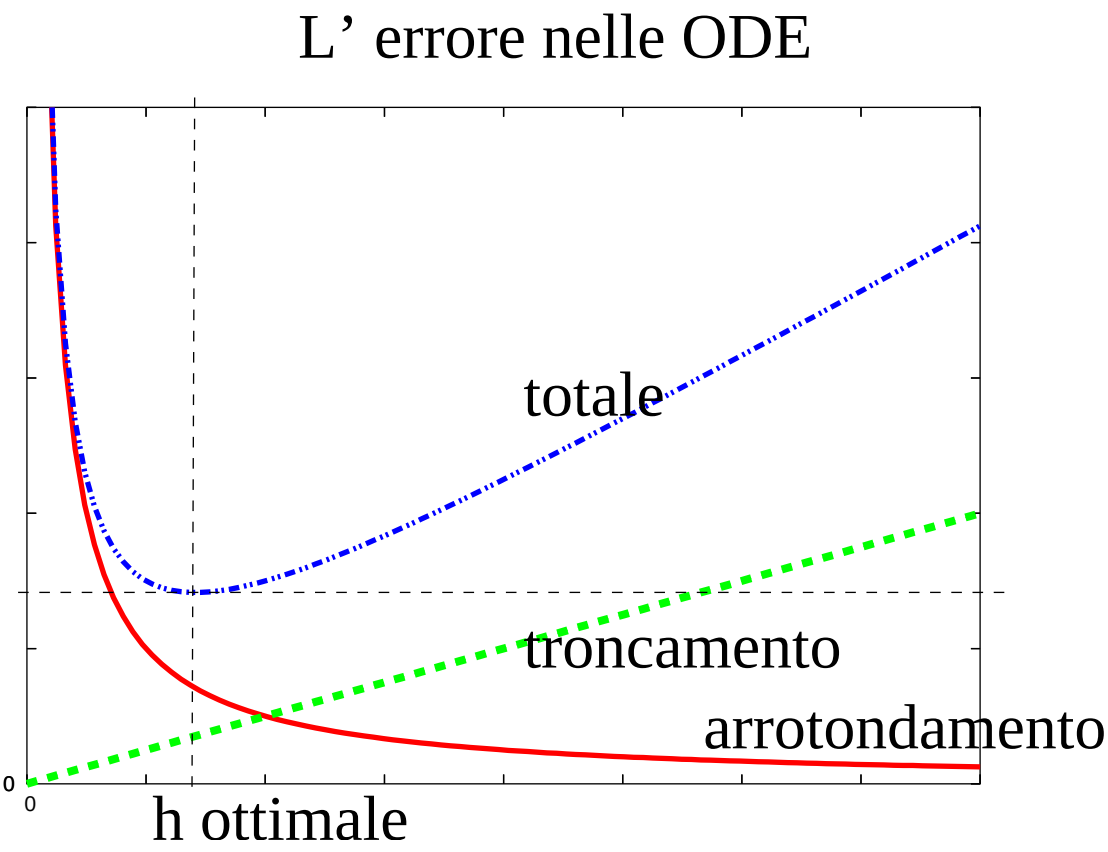


Figura 10.1: L’ errore nelle ODE.

10.2 Errori

L' errore di troncamento tende a zero con il passo di discretizzazione, h.

L' errore di arrotondamento cresce quando h diventa piccolo.

L' errore totale, che è una composizione dei due, è minimo per un certo valore di h, detto *ottimale*.

10.3 Sistemi di ODE.

Supponiamo di avere un sistema di m equazioni differenziali ordinarie nelle incognite $y_1(t)$, ..., $y_m(t)$

$$\left\{ \begin{array}{ll} y_1' & = f_1(t, y_1, \ldots, y_m) \\ \ldots & = \ldots \\ y_m' & = f_m(t, y_1, \ldots, y_m) \end{array} \right. \tag{10.14}$$

il dominio sia $t \in (t_0, T]$, le condizioni iniziali

$$y_1(t_0) = y_1^{(0)},\ y_2(t_0) = y_2^{(0)},\ \ldots,\ y_m(t_0) = y_m^{(0)}. \tag{10.15}$$

Per risolverlo, si puø applicare a ciascuna equazione uno dei metodi studiati per le singole equazioni. Ad esempio, applicando il metodo di Eulero esplicito, otteniamo le relazioni ricorrenti:

$$\left\{ \begin{array}{ll} \tilde{y}_1^{(n+1)} & = \tilde{y}_1^{(n)} + h f_1(t, \tilde{y}_1^{(n)}, \ldots, \tilde{y}_m^{(n)}) \\ \ldots & = \ldots \\ \tilde{y}_m^{(n+1)} & = \tilde{y}_m^{(n)} + h f_m(t, \tilde{y}_1^{(n)}, \ldots, \tilde{y}_m^{(n)}) \end{array} \right. \tag{10.16}$$

Esempio 10.3.1 Risolvere il sistema di ODE:

$$\left\{ \begin{array}{ll} y_1' & = +C_1 y_1(1 - b_1 y_1 - d_2 y_2), \\ y_2' & = -C_2 y_2(1 - b_2 y_2 - d_1 y_1), \end{array} \right. \tag{10.17}$$

dove $C_1 = C_2 = 1$, $b_1 = b_2 = 0$, $d_1 = d_2 = 1$, $t \in [0, 10]$, $y_1(0) = y_2(0) = 2$.

Si ottiene lo schema:

$$\left\{ \begin{array}{ll} \tilde{y}_1^{(n+1)} & = +\tilde{y}_1^{(n)} + h\tilde{y}_1^{(n)}(1 - \tilde{y}_2^{(n)}), \\ \tilde{y}_2^{(n+1)} & = -\tilde{y}_2^{(n)} + h\tilde{y}_2^{(n)}(1 - \tilde{y}_1^{(n)}). \\ \tilde{y}_1^{(0)} & = 2, \quad \tilde{y}_2^{(0)} = 2. \end{array} \right. \tag{10.18}$$

10.4 Runge-Kutta per sistemi di ODE.

Ecco un esempio di applicazione di uno schema del tipo Runge-Kutta (lo schema di Heun) per risolvere un sistema di equazioni differenziali.

Risolvere il problema differenziale:

$$y'' + 2y' + y = x + 1, \quad x \in [0, 1] \tag{10.19}$$

$$y(0) = 0, \quad y'(0) = 0. \tag{10.20}$$

La soluzione di questo problema è $y = e^{-x} + x - 1$.

L' equazione (10.19) è equivalente al sistema di equazioni

$$\begin{cases} y' &= z = f_1(x, y, z) \\ z' &= -2z - y + x + 1 = f_2(x, y, z) \end{cases} \tag{10.21}$$

Lo schema di Heun si scrive:

$$\bar{y}_{n+1} = \bar{y}_n + \frac{1}{2}k_1 + \frac{1}{2}k_2 \tag{10.22}$$

$$k_1 = hf(x_n, \bar{y}_n) \tag{10.23}$$

$$k_2 = hf(x_n + h, \bar{y}_n + k_1). \tag{10.24}$$

La sua ovvia generalizzazione ad un sistema di due equazioni in due incognite è

$$\bar{y}_{n+1} = \bar{y}_n + \frac{1}{2}k_{y,1} + \frac{1}{2}k_{y,2} \tag{10.25}$$

$$\bar{z}_{n+1} = \bar{z}_n + \frac{1}{2}k_{z,1} + \frac{1}{2}k_{z,2} \tag{10.26}$$

$$k_{y,1} = hf_1(x_n, \bar{y}_n, \bar{z}_n) \tag{10.27}$$

$$k_{z,1} = hf_2(x_n, \bar{y}_n, \bar{z}_n) \tag{10.28}$$

$$k_{y,2} = hf_1(x_n + h, \bar{y}_n + k_{y,1}, \bar{z}_n + k_{z,1}) \tag{10.29}$$

$$k_{z,2} = hf_2(x_n + h, \bar{y}_n + k_{y,1}, \bar{z}_n + k_{z,1}) \tag{10.30}$$

dove i termini $k_{y,*}, k_{z,*}$ indicano *incrementi riguardanti y oppure z*.

Applicando questa generalizzazione al sistema (10.21) otteniamo le relazioni

$$\bar{y}_{n+1} = \bar{y}_n + \frac{1}{2}k_{y,1} + \frac{1}{2}k_{y,2} \tag{10.31}$$

$$\bar{z}_{n+1} = \bar{z}_n + \frac{1}{2}k_{z,1} + \frac{1}{2}k_{z,2} \tag{10.32}$$

$$k_{y,1} = h\bar{z}_n, \tag{10.33}$$

$$k_{z,1} = h(-2\bar{z}_n - \bar{y}_n + x_n + 1) \tag{10.34}$$

$$k_{y,2} = h(\bar{z}_n + k_{z,1}) \tag{10.35}$$

$$k_{z,2} = h(-2(\bar{z}_n + k_{z,1}) - (\bar{y}_n + k_{y,1}) + (x_n + h) + 1) \tag{10.36}$$

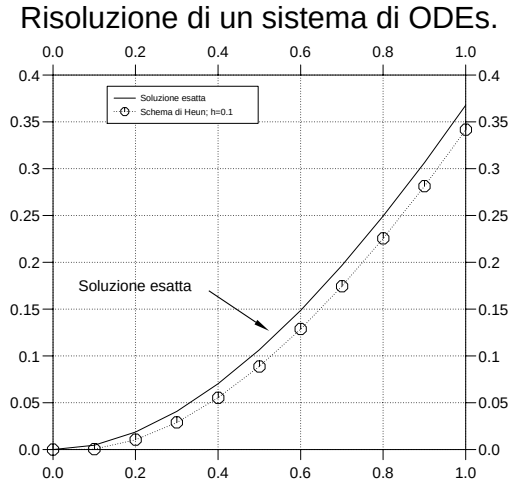


Figura 10.2: Sistema di ODE risolto con lo schema di Heun.

La figura (10.2) mostra l' andamento della componente $\bar{y}_n$ della soluzione di questo sistema di equazioni alle differenze, usando un passo $h = 0.1$ e imponendo le condizioni iniziali (10.20).

10.5 Risoluzione di un BVP con shooting.

E' dato il problema con valori al contorno (BVP)

$$y'' + 2y' = 4x + 3e^x, x \in [0, 1] \tag{10.37}$$

$$y(0) = 1, \quad y(1) = e. \tag{10.38}$$

Vogliamo calcolare un' approssimazione numerica della sua soluzione, usando un metodo di *shooting*. Esso consiste nel risolvere i problemi

$$y'' + 2y' = 4x + 3e^x, x \in [0, 1] \tag{10.39}$$

$$y(0) = 1, \quad y'(0) = s \tag{10.40}$$

per diversi valori del parametro s: ad ogni valore corrisponderà una soluzione $y^*(x;\ s)$ che in generale *non* soddisfa alla condizione al contorno in $x = 1$, ossia risulta $y^*(1;\ s) \neq y(1)$. Risolvere il problema con le condizioni al contorno (10.38) equivale a calcolare la soluzione s dell' equazione

$$\Phi(s) = y^*(1;\ s) - y(1) = 0. \tag{10.41}$$

Si può determinarla con qualsiasi schema per calcolare le soluzioni di un' equazione non lineare. Useremo lo schema della regula falsi

$$s_{k+1} = s_k - \Phi(s_k)/\frac{\Phi(s_k)-\Phi(s_{k-1})}{s_k - s_{k-1}} \tag{10.42}$$

$$k = 1, 2, \ldots, \quad s_0, s_1 \text{ valori dati.} \tag{10.43}$$

Per integrare numericamente l' equazione (10.37) usiamo lo schema alle differenze che si ottiene usando le approssimazioni

$$y'_n = \frac{\Delta y_n}{h} + O(h), \tag{10.44}$$

$$y''_n = \frac{\Delta^2 y_n}{h^2} + O(h), \tag{10.45}$$

e scrivendo l' equazione (10.37) nel punto $x_n = nh$, ossia lo schema

$$\bar{y}_{n+2} + 2(h-1)\bar{y}_{n+1} + (1-2h)\bar{y}_n = 4h^2 x_n + 3h^2 e^{x_n}. \tag{10.46}$$

Stimiamo ora i due valori iniziali di s. Poniamo ad esempio

$$s_0 = \frac{\bar{y}_1^{+\delta} - y(0)}{h}, \quad s_1 = \frac{\bar{y}_1^{-\delta} - y(0)}{h}, \tag{10.47}$$

$h = 1/2$, dove i valori $\bar{y}_1^{+\delta}$, $\bar{y}_1^{-\delta}$ sono stati ricavati dalla relazione (10.46) ponendo $h = 1/2$, $\bar{y}_0 = y(0) = 1$, $\bar{y}_2^{\pm\delta} = y(1) \pm \delta$, essendo δ un valore arbitrario, diciamo $\delta = 1$.

Utilizzando questi due valori iniziali e applicando lo schema (10.42) otteniamo i risultati:

x	y	y0	y1	y2	yb
0.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.1000	1.0152	0.4037	0.8037	1.0458	1.0000
0.2000	1.0614	-0.0434	0.6766	1.1125	1.0300
0.3000	1.1399	-0.3639	0.6121	1.2030	1.0912
0.4000	1.2518	-0.5757	0.6051	1.3201	1.1847
0.5000	1.3987	-0.6926	0.6520	1.4662	1.3121
0.6000	1.5821	-0.7254	0.7503	1.6438	1.4747
0.7000	1.8038	-0.6822	0.8984	1.8554	1.6743
0.8000	2.0655	-0.5689	1.0956	2.1034	1.9126
0.9000	2.3696	-0.3899	1.3417	2.3901	2.1917
1.0000	2.7183	-0.1479	1.6374	2.7183	2.5137

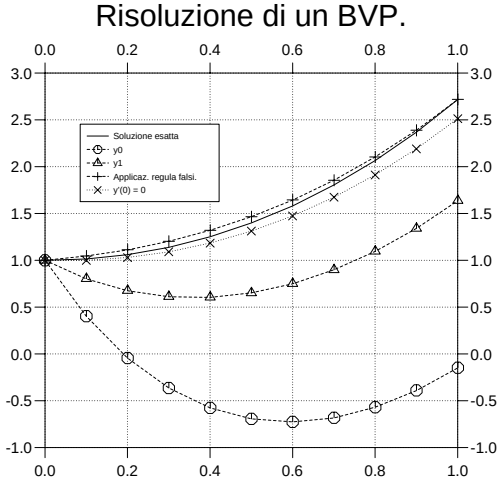


Figura 10.3: Applicazione del metodo di shooting con uso della regula falsi.

dove `y` è la soluzione esatta $y = x^2 - x + e^x$ del problema dato, `yi` $= \bar{y}^{(i)} = \{\bar{y}_j(s_i) = \bar{y}(x_j;\ s_i),\quad j = 1, \ldots n = 1/h\}$ sono le soluzioni dello schema (10.46), ottenute ponendo di volta in volta

$$\bar{y}_1(s_i) = y_0 + hs_i, \tag{10.48}$$

dove s_0 e s_1 sono dati dalla (10.47) e s_2 quello ottenuto applicando la regula falsi (10.42). Notare come basta una sola iterazione con lo schema della regula falsi per ottenere una buona approssimazione della soluzione cercata. Se si calcola anche $\bar{y}^{(3)}$ si vede che $\bar{y}^{(2)} = \bar{y}^{(3)}$: non si guadagna nulla procedendo oltre la prima iterazione.

`yb` $= \bar{y}(x_n;\ 0)$ è la soluzione numerica che si ottiene imponendo la condizione iniziale esatta $y'(0) = 0$. E' interessante notare come $\bar{y}^{(2)}$ sia piú accurata di $\bar{y}(x_n;\ 0)$: conoscere esattamente la condizione iniziale sulla derivata è meno vantaggioso che conoscere il valore della funzione nell' estremo finale dell' intervallo.

La figura (10.3) fornisce uno schizzo delle curve che si ottengono plottando i valori trovati.

10.6 BVP e sistemi alle DF

E' dato il problema con valori al contorno (BVP)

$$y'' + 2y' = 4x + 3e^x, x \in [0, 1] \tag{10.49}$$

$$y(0) = 1, \quad y(1) = e, \tag{10.50}$$

che ha la soluzione

$$y = x^2 - x + e^x. \tag{10.51}$$

Si può determinare una soluzione numerica di questo problema usando uno schema alle differenze finite. Usiamo lo schema alle differenze

$$\bar{y}_{n+2} + 2(h-1)\bar{y}_{n+1} + (1-2h)\bar{y}_k = 4h^2 x_n + 3h^2 e^{x_n} \tag{10.52}$$

$x_n = nh$, che si ottiene dalla equazione (10.49) impiegando note approssimazioni dell' operatore differenziale. Posto $h = 0.25$, abbiamo $\bar{y}_0 = y(0) = 1$, $\bar{y}_4 = y(1) = e$, e scrivendo le relazioni (10.52) per $n = 0, 1, 2$, otteniamo un sistema lineare di 3 equazioni in tre incognite

$$\mathbf{As} = \mathbf{b} \tag{10.53}$$

dove

```
    [ 2 * (h - 1)   1               0          ]
A = [ (1 - 2 * h)   2 * (h - 1)     1          ]
    [ 0             (1 - 2 * h)     2 * (h - 1)]

     [ -1.5000    1.0000         0 ]
A =  [  0.5000   -1.5000    1.0000 ]
     [       0    0.5000   -1.5000 ]

    [ (3 * h*h - (1 - 2 * h) )                    ]
b = [ (4 * h*h *   h + 3 * h*h * exp(h) )         ]
    [ (4 * h*h * 2*h + 3 * h*h * exp(2*h) - exp(1))]

    [ -0.3125 ]
  = [  0.3033 ]
    [ -2.2841 ]
```

Risolvendo questo sistema lineare otteniamo il risultato

```
         x          y         yn
         0     1.0000     1.0000
    0.2500     1.0965     1.2673
    0.5000     1.3987     1.5884
    0.7500     1.9295     2.0522
    1.0000     2.7183     2.7183
```

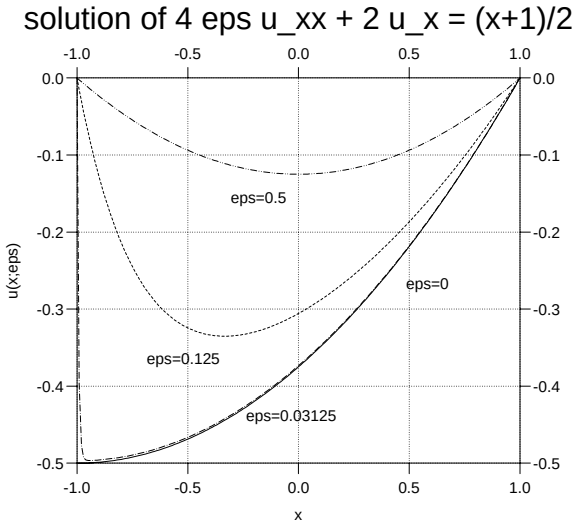


Figura 10.4: Soluzioni del problema per alcuni valori di ϵ.

10.7 Strato limite

10.7.1 Formulazione del problema

Si vuole risolvere numericamente il problema [HS93]

$$\begin{cases} 4\epsilon u_{xx} + 2u_x &= (x+1)/2, \quad x \in (-1,1), \quad \epsilon > 0, \\ u(-1) = u(1) &= 0 \end{cases} \tag{10.54}$$

La sua soluzione analitica è

$$\begin{aligned} u &= -\frac{(4\,\epsilon - 1)\; e^{\frac{1}{2\,\epsilon} - \frac{x}{2\,\epsilon}}}{2\,e^{\frac{1}{\epsilon}} - 2} + \\ &+ \frac{x^2 + (1 - 4\,\epsilon)\; x + 8\,\epsilon^2 - 2\,\epsilon}{4} - \\ &+ \frac{(4\,\epsilon^2 - 3\,\epsilon + 1)\; e^{\frac{1}{\epsilon}} - 4\,\epsilon^2 - \epsilon}{2\,e^{\frac{1}{\epsilon}} - 2} \end{aligned}$$

Quando $\epsilon \to 0$, la soluzione di questa equazione ha uno *strato limite* o *boundary layer* in $x = -1$. Il significato di questa affermazione si puó capire guardando le figure 10.4 e 10.5.

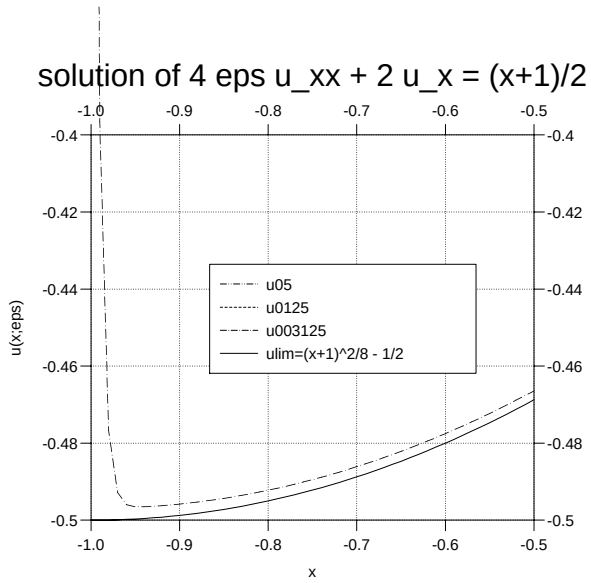


Figura 10.5: Dettaglio della figura precedente.

10.7.2 Risoluzione numerica

Definiamo una griglia in [0,1] ponendo

$$x_i = -1 + ih, i = 0, 1, \ldots, N$$

N intero positivo dato, $h = 2/N$.

Lo schema alle differenze finite (FD) del *secondo ordine centrato* (*centered FD*) é definito come

$$4\epsilon\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + 2\frac{u_{j+1} - u_{j-1}}{2h} = \frac{x_j + 1}{2}, \quad j = 1, 2, \ldots, N-1. \quad (10.55)$$

Lo schema del *primo ordine upwind* (*Upwind FD*) come

$$4\epsilon\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + 2\frac{u_{j+1} - u_j}{h} = \frac{x_j + 1}{2}, \quad j = 1, 2, \ldots, N-1. \quad (10.56)$$

Quando $\epsilon = O(1)$, l' errore di troncamento di (10.55) è $O(h^2)$, mentre quello di (10.56) è $O(h)$.

Sia $u(x)$ la soluzione esatta e $\tilde{u}(x)$ quella approssimata. Poniamo

$$e(x) = u(x) - \tilde{u}(x), \quad E = \max_{0 \le j \le N} |e(x_j)|. \quad (10.57)$$

La figura 10.6 mostra $\log_{10} E$ in funzione di $-\log_{10} \epsilon$.

La figura 10.7 mostra $\log_{10} |e|$ per alcuni valori di ϵ.

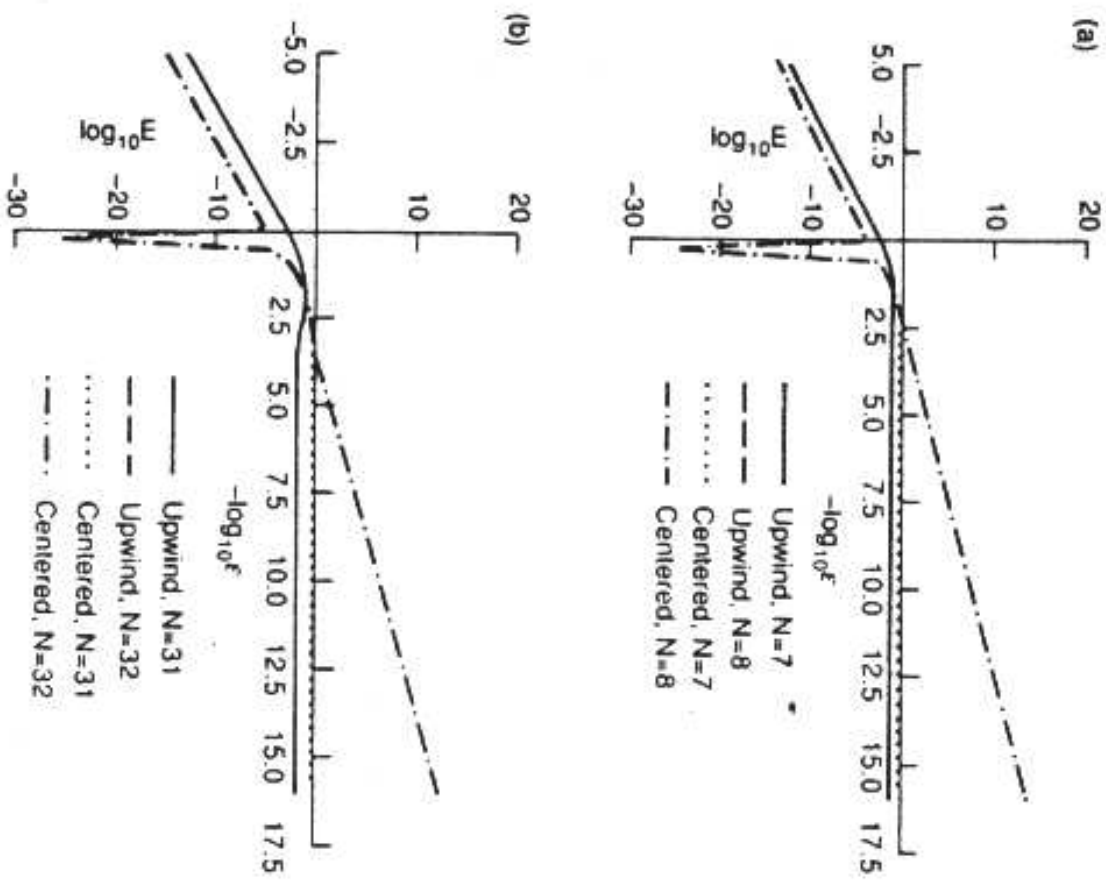


Figura 10.6: $\log_{10} E$ in funzione di $-\log_{10}\epsilon$.

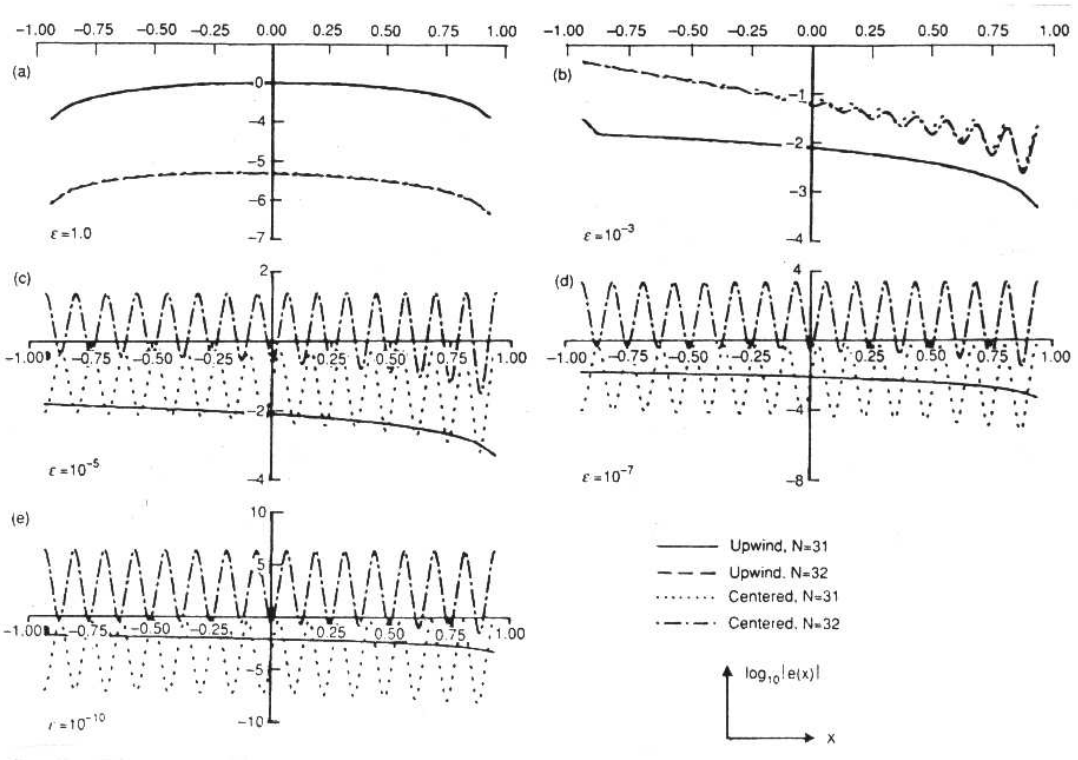


Figura 10.7: $\log_{10}|e|$ in funzione di ϵ.

10.7.3 Conclusioni

Notiamo le seguenti caratteristiche dei due metodi.

Differenze centrali:

- la soluzione calcolata oscilla quando $\epsilon \to 0$;
- errore e soluzione sono limitati per N dispari e illimitati per N pari;
- $E = O(h^2)$ quando $\epsilon = O(1)$;

Differenze Upwind:

- la soluzione calcolata *non* oscilla quando $\epsilon \to 0$;
- i comportamenti dell' errore e della soluzione non dipendono dal fatto che N sia pari o dispari;
- $E = O(h)$ quando $\epsilon = O(1)$;

Capitolo 11

Metodi Multigrid

11.1 Introduzione

Supponiamo di voler risolvere sul dominio Ω, costituito dal quadrato $[0,1] \times [0,1]$, il problema di Poisson in due dimensioni

$$-\Delta u = -\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y), \tag{11.1}$$

con condizioni al contorno di Dirichlet [QS06].

Discretizziamo il dominio Ω con una griglia quadrata uniforme come quella schizzata in figura 11.1, dividendo ogni lato in n parti, individuate dalle ascisse e ordinate $x_i = ih$ e $y_j = jh$, $i,j = 0,\ldots,n$, $h = 1/(n+1)$. Nella figura i nodi sono numerati in ordine lessicografico per colonne. Siano $u_{ij} = u(ih, jh)$ e $f_{ij} = f(ih, jh)$.

Approssimiamo l' operatore di Laplace Δ usando il classico stencil a 5 punti. Otteniamo un sistema lineare di n^2 equazioni in n^2 incognite

$$4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i-1,j-1} - u_{i,j+1} = h^2 f_{ij}, \quad i,j = 1,\ldots,n. \tag{11.2}$$

Risolviamo il sistema (11.2), che in forma matriciale scriviamo $A \cdot x = b$, con il metodo di rilssamento di Jacobi [QS06]

$$u^{(k+1)} = -D^{-1}(L+U)u^{(k)} + D^{-1}b = u^{(k)} + D^{-1}r^{(k)},$$

dove $A = L + D + U$ è lo splitting di A in componente triangolare bassa (L), diagonale (D) e triangolare alta (U), $r^{(k)} = b - Au^{(k)}$ è il residuo.

Se $g(x)$ è una funzione periodica in $I = [0,1]$ (il nostro dominio è il quadrato $[0,1]^2$, quindi la soluzione, come funzione della sola x o della sola y

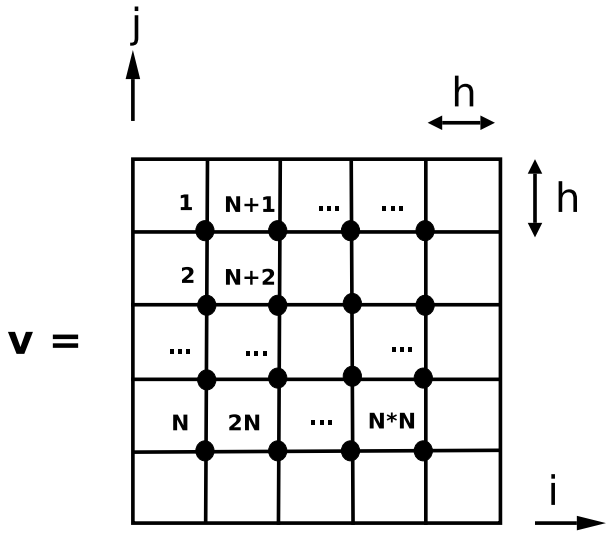


Figura 11.1: Griglia quadrata uniforme e numerazione delle incognite.

non ci interessa al di fuori di I, possiamo assumere che sia periodica...) sotto opportune ipotesi possiamo scrivere la serie di Fourier [Ber94]

$$g(x) = \frac{a_0}{2} + \sum_{m=1}^{+\infty} (a_m \cos(2\pi m x) + b_m \sin(2\pi m x)). \tag{11.3}$$

I valori $s_m = \sqrt{a_m^2 + b_m^2}$ rappresentano il cosiddetto *spettro di potenza* della $g(x)$: quando s_m è "grande", significa che la componente m-esima (detta *m-esima armonica*) della somma 11.3 è "grande" e viceversa. Notate che la componente m-esima ha frequenza m, quindi se s_m è grande quando $m \gg 1$, diciamo che $g(x)$ ha una "forte" componente in armoniche alte, una debole componente se s_m è piccolo quando m è grande.

Poniamo $n = 15$, il sistema (11.2) ha $N = 15^2 = 225$ incognite. Consideriamo l' errore dello schema di Jacobi alla k-esima iterazione, $e^{(k)} = u - u^{(k)}$. L' iterazione di Jacobi calcola ogni nuova approssimazione $u^{(k+1)}(x_i, y_j) = u_{i,j}^{(k+1)}$ usando la relazione

$$v_{i,j}^{(k+1)} = \frac{v_{i-1,j}^{(k)} + v_{i+1,j}^{(k)} + v_{i,j-1}^{(k)} + v_{i,j+1}^{(k)} + h^2 f_{i,j}}{4}. \tag{11.4}$$

Il valore in ogni nodo, P, viene aggiornato usando i valori calcolati al passo precedente, associati al nodo stesso e ai nodi *immediatamente vicini*, ossia a nord, sud, est e ovest di P. L' aggiornamento non tiene conto dei nodi pi˘

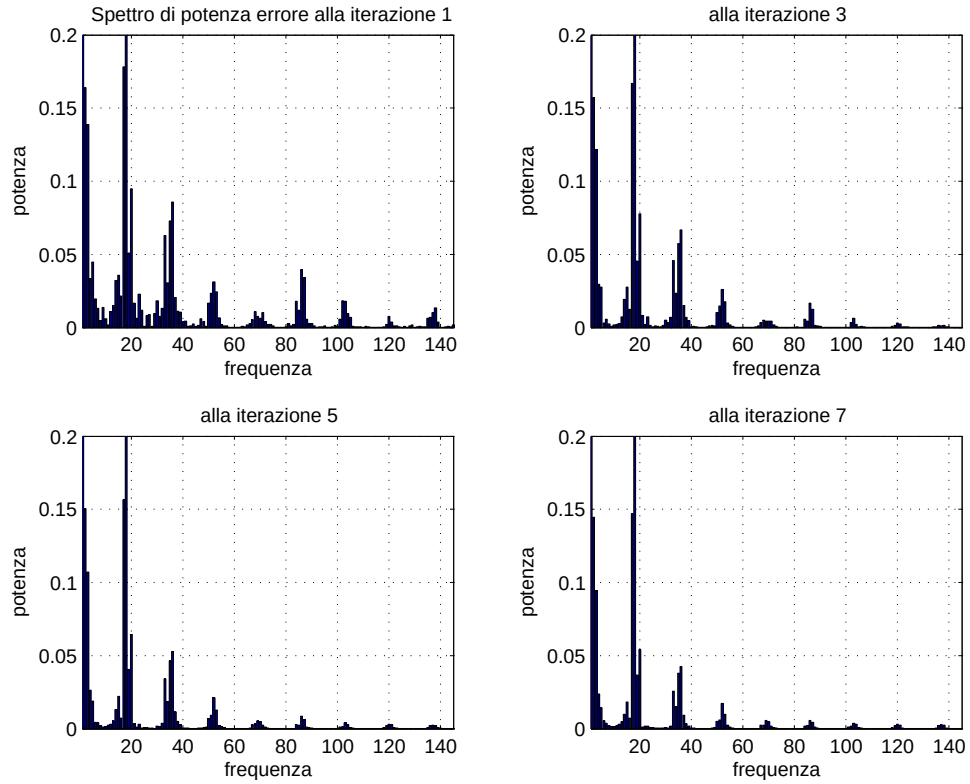


Figura 11.2: Spettro di potenza dell' errore in alcune iterazioni di Jacobi.

lontani da P, conseguentemente ci aspettiamo che, al crescere del numero di iterazioni, le componenti in frequenza dell' errore che coinvolgono punti *lontani*, ossia le frequenze pi$\breve{}$ *basse* (m piccolo) , vengano ridotte *meno* delle componenti ad *alta* frequenza (m grande), che riguardano punti *vicini* nella griglia. Questa supposizione è avvalorata dalla figura 11.2, che mostra lo *spettro di potenza*, per le frequenze $m = 1, \ldots, 150$ dell' errore, alle iterazioni $k = 1, 3, 5, 7$ (per $m > 256/2 = 128$ lo spettro dell' errore ha grandezza trascurabile). La figura mostra come per $m < 50$, ad esempio, le componenti dello spettro dell' errore, vengano ridotte molto meno di quelle per $m > 50$.

I metodi multigrid, che verrano descritti nei prossimi paragrafi, per smussare anche le componenti di frequenza bassa combinano diverse soluzioni approssimate ottenute discretizzando il problema su pi$\breve{}$ di una griglia, a diversi livelli di raffinamento.

Risolviamo il problema di Poisson, combinando il metodo di Jacobi con

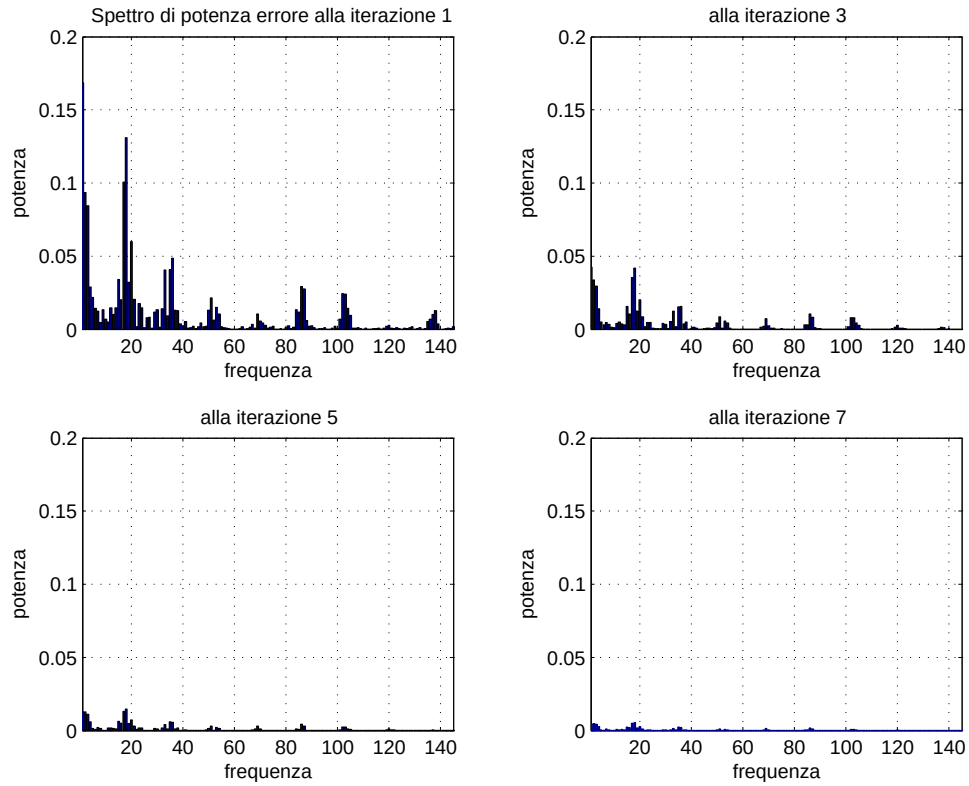


Figura 11.3: Spettro di potenza dell'errore in alcune iterazioni del metodo Multigrid.

uno schema Multigrid. Quest' ultimo (vedere pi˘ avanti per comprendere il significato dei termini) consiste in una serie di cicli di tipo V, articolati su $K = 4$ livelli, ogni livello grossolano ha celle (quadrati) di lato doppio di quelle del livello immediatamente pi˘ raffinato. Ogni iterazione di livello k prevede un' operazione di pre–smussamento (smoothing), una applicazione ricorsiva del metodo e un passo di post–smussamento. La figura 11.3 mostra che *tutte* le componenti di frequenza $m < 150$ vengono egualmente ridotte nelle prime 7 iterazioni.

11.2 Nested iterations

Sia dato un dominio *poligonale* Ω, ossia un dominio che ha una frontiera fatta da segmenti. In generale un dominio "reale" avrà una frontiera che è una curva arbitraria, ma noi supponiamo di poterla approssimare con una *spezzata* fatta di segmenti di retta sufficientemente piccoli, senza perdere molto in accuratezza. Ricordiamo che il *diametro* di un insieme di punti, D, è:

$$\mathrm{diam}(D) = \sup_{P,Q \in D} \mathrm{dist}(P, Q).$$

Supponiamo di voler risolvere il sistema lineare

$$A_K \, z = g, \tag{11.5}$$

dove $A_K(p_K \times p_K)$ è l' approssimazione discreta di un dato operatore differenziale, che caratterizza il problema da risolvere, tramite una griglia, τ_K, su Ω. Vogliamo utilizzare K sistemi ausiliari

$$A_i x_i = b_i, \quad i = 1, \ldots, K-1,$$

che sono discretizzazioni su una gerarchia di griglie dello stesso problema continuo, ognuna avente p_i incognite, $p_1 < \ldots < p_K$. Pensiamo ad esempio al dominio $[0,1]^2$ e alle griglie quadrate ottenute dividendo i lati in 2, 4, 8, 16, ... parti.

L' algoritmo *Nested Iterations* [KA03] (NI), è descritto nella tabella 11.1. L' operatore $\mathcal{S}(A, x, b)$ rappresenta un solutore iterativo di sistemi lineari (rilassamento o *smoothing*). Le relazioni che useremo sono:

$$x_k^{(i+1)} = \mathcal{S}(A_k, x_k^{(i)}, b_k), \quad e_k^{(i+1)} = \left| x - x_k^{(i+1)} \right| \simeq \mu_k \left| x - x_k^{(i)} \right| = \mu_k e_k^{(i)},$$

dove μ_k è la *costante di contrattività* del metodo iterativo, pensato come metodo del punto fisso. L' operatore $\mathcal{I}_{k-1}^k$ è un *operatore di prolungamento*

Tabella 11.1: Nested iterations.

```
1  begin
2    Choose   m_k ∈ ℕ, k = 1, ..., K;
3    x := A_1^{-1} b_1;
4    for k := 1, K do
5      x := I_{k-1}^k x; { x is prolonged }
6      for i := 1, m_k do
7        x := S(A_k, x, b);
8      endfor
9    endfor
10   return   x;
11 end
```

(concetto spiegato meglio pi˘ avanti) che permette di *interpolare* una funzione da una griglia grossolana ad una pi˘ raffinata. L' algoritmo NI può essere utilizzato per risolvere il sistema lineare (11.5), quando i K sistemi ausiliari, che sono discretizzazioni dello stesso problema continuo, hanno p_i incognite, e risulta $p_1 < \ldots < p_K$.

Se ogni metodo risolutivo $\mathcal{S}(A_k, \cdot, \cdot)$ ha costante di contrattività μ_k, ed esiste un valore μ

$$\mu_k \leq \mu < 1, \quad k = 1, \ldots, K,$$

t.c. ponendo $\mu_k := \mu$ il metodo NI converge, allora ha *complessità ottimale* [KA03] ed è chiamato *metodo multigrid*. In tal caso si può scegliere un valore m, tale che ponendo $m_k = m$, $k = 0, \ldots, K$, l' algoritmo NI converge.

Esempio 11.2.1 Un suggerimento esemplificativo. Supponiamo di approssimare su ogni mesh τ_i con un metodo agli Elementi Finiti la soluzione, $u(x, y)$, di un problema differenziale su Ω, ottenendo le approssimazioni $\tilde{u}_i(x, y)$, $i = 1, \ldots, m$. Supponiamo che fra i diametri, d_i, valgano le relazioni $\text{diam}_{i+1} = \text{diam}_i/2$ e

$$u - \tilde{u}_i = e_i \simeq C \, \text{diam}_i^2, \tag{11.6}$$

dove la costante C *non dipende da* diam_i. Allora abbiamo

$$e_i \simeq 4 \, e_{i+1}, \quad \tilde{u}_i - \tilde{u}_{i+1} \simeq 3 \, e_{i+1},$$

$$u \simeq \tilde{u}_{i+1} + \frac{\tilde{u}_i - \tilde{u}_{i+1}}{3} = \frac{2 \, \tilde{u}_{i+1} - \tilde{u}_i}{3}. \tag{11.7}$$

Usando la relazione (11.7), possiamo calcolare un' approssimazione della soluzione, pi˘ accurata di $\tilde{u}_{i+1}$.

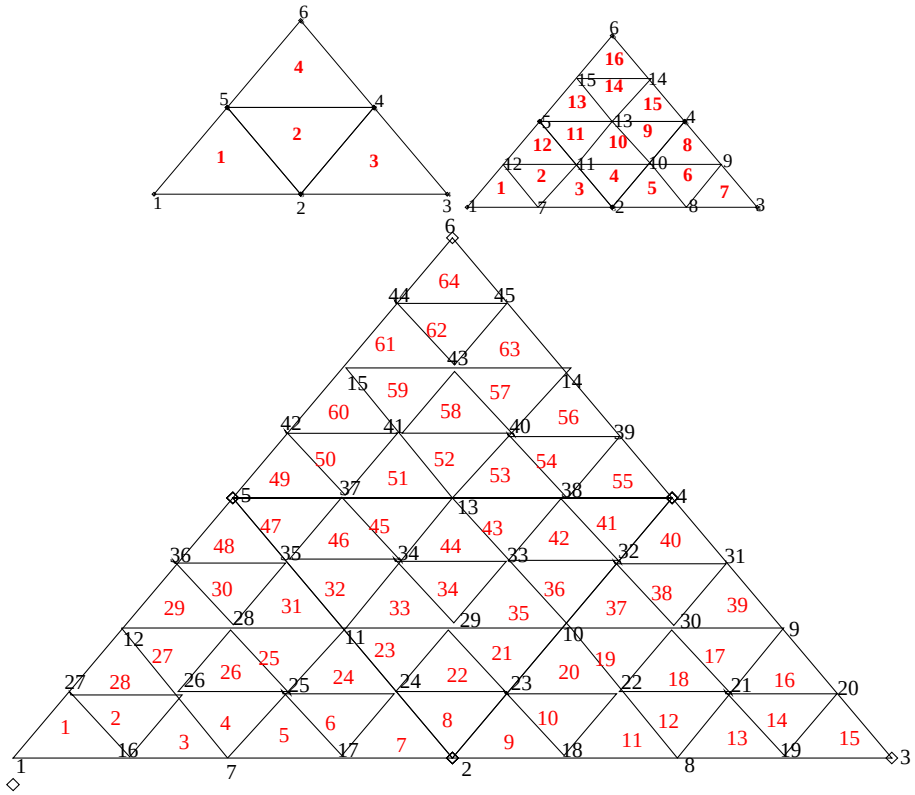


Figura 11.4: Una mesh (livello $k = 0$) e due suoi raffinamenti regolari (livelli $k = 1$ e $k = 2$). La mesh pi˘ raffinata è stata ingrandita.

11.3 Griglie triangolari

Una *triangolazione* di Ω è una suddivisione fatta di triangoli, in cui nessun *vertice* di triangolo cade su un *lato* di un' altro triangolo. I lati dei triangoli vengono detti anche *lati della triangolazione*.

Supponiamo di avere un insieme di triangolazioni, τ_i, $i = 1, \ldots, K$, su Ω. Diciamo che τ_i è la *mesh a livello* i. Assumiamo che queste mesh siano raffinamenti *regolari* della mesh iniziale, τ_1, ossia ogni triangolo di τ_{i+1} si ottiene unendo i punti medi dei lati di un triangolo di τ_i. Se quest' ultima ha diametro diam_i, p_i nodi e t_i elementi, si ottiene una mesh che ha $t_{i+1} = 4t_i$ elementi e diametro $\text{diam}_{i+1} = \text{diam}_i/2$.

La figura 11.4 mostra un esempio di mesh triangolare e due raffinamenti regolari.

I triangoli di ogni mesh raffinata sono simili a quelli della mesh da cui provengono, quindi se τ_1 è di Delaunay [KA03] tutte le altre lo sono.

Una *famiglia* di mesh τ_k, $k = 1, \ldots$, sul dominio Ω, ognuna avente diametro diam_k, è *quasi uniforme* [BRS02] se esiste $\rho > 0$ t.c.

$$\max\{\mathrm{diam}(B_T) : T \in \tau_k\} \geq \rho\, \mathrm{diam}_k\, \mathrm{diam}(\Omega), \tag{11.8}$$

per ogni $\mathrm{diam}_k \in (0, 1]$, dove B_T è la pi˘ grande palla contenuta in T.

Sia V_i lo spazio delle funzioni $\phi \in \mathcal{C}^0$ lineari a tratti su τ_i, che sono *nulle su* $\partial\Omega$,

$$\phi = 0, \mathrm{su}\ \ \partial\Omega. \tag{11.9}$$

Si noti che se $n(\tau_k)$ è l' insieme dei nodi della k-esima mesh,

$$n(\tau_i) \supset n(\tau_j) \Rightarrow V_i \supset V_j, \quad 1 \leq j \leq i \leq K. \tag{11.10}$$

Nota: il fatto che ogni $v \in V_k$ sia nulla su $\partial\Omega$ non è limitante, se il problema di risolvere è di Dirichlet. Infatti, se $\tilde{u}_h \to \tilde{u}$, soluzione del problema di Dirichlet omogeneo, quando u è la soluzione del problema di Dirichlet $u = g$ su $\partial\Omega$, risulta

$$u_h = \tilde{u}_h + \sum_{x_i \in \partial\Omega} g(x_i)\, \phi_i(x) \to u.$$

Quando il problema da risolvere non ha condizioni al contorno solamente di Dirichlet, può essere necessario cambiare lo spazio delle funzioni approssimanti.

Definizione 11.3.1 *Sia N_k il numero dei nodi, $P_{\psi_k(i)}$, $i = 1, \ldots, N_k$,* interni *a τ_k, $\psi_k : \{1, \ldots, N_k\} \to \{1, \ldots, p_k\}$. Definiamo il prodotto interno su V_k, dipendente dalla mesh (mesh–dependent inner product), come*

$$(v, w)_k := \mathrm{diam}_k^2 \sum_{i=1}^{N_k} v(P_{\psi_k(i)})\, w(P_{\psi_k(i)}).$$

La condizione (11.9) semplifica le dimostrazioni, ma i ragionamenti che faremo possono essere estesi a spazi di funzioni che non soddisfano questa condizione. Per sviluppare alcuni esempi, definiamo anche gli spazi W_i delle funzioni $\phi \in \mathcal{C}^0$ lineari a tratti su τ_i. Notare che $W_i \supset V_i$. Il prodotto interno su W_k, dipendente dalla mesh, è naturalmente

$$(v, w)_k := \mathrm{diam}_k^2 \sum_{i=1}^{p_k} v(P_i)\, w(P_i).$$

Definizione 11.3.2 *Sia*

$$\mathcal{I}_{k-1}^{k} : V_{k-1} \to V_k,$$

l' operatore di prolungamento *(coarse–to–fine operator), detto anche* interpolazione *(interpolation), che facciamo coincidere con l'* operatore di immersione*:*

$$\mathcal{I}_{k-1}^{k} v = v, \quad \forall v \in V_{k-1}.$$

Definizione 11.3.3 *Sia*

$$\mathcal{I}_{k}^{k-1} : V_k \to V_{k-1},$$

l' operatore di restrizione *o pesatura residuale (residual weighting, fine–to–coarse operator), che definiamo come* il trasposto di $\mathcal{I}_{k-1}^{k}$*, rispetto ai prodotti interni* $(\cdot,\cdot)_{k-1}$*,* $(\cdot,\cdot)_k$*, ossia*

$$(\mathcal{I}_{k}^{k-1} w, v)_{k-1} = (w, \mathcal{I}_{k-1}^{k} v)_k = (w, v)_k, \quad \forall v \in V_{k-1},\ w \in V_k.$$

Nota: quando la restrizione è il trasposto del prolungamento, è piú agevole provare che un metodo multigrid converge. Tuttavia sono stati proposti algoritmi in cui questo non si verifica: in questi casi non vengono date dimostrazioni di convergenza e ci si limita ad una "validazione" dell' algoritmo su problemi test.

11.4 Multigrid

Lo schema multigrid usa due strategie principali [Wes92]:

- smussamento su τ_i: vengono smussate le componenti ad alta frequenza dell' errore;
- correzione dell' errore su τ_{i-1}, usando uno schema iterativo sviluppato su quest' ultimo spazio: vengono smussate le componenti a bassa frequenza dell' errore.

Descriviamo in dettaglio un algoritmo multigrid "generale", su K griglie triangolari, $\tau_1 \subset \ldots \subset \tau_K$, implementato dalla funzione $MG(k, z, g)$, che viene definita ricorsivamente.

Le considerazioni che faremo valgono anche, con modifiche opportune, per griglie quadrate e rettangolari.

Innanzitutto $MG(1, z_0, g) := A_1^{-1}\, g$, ossia al livello pi$^{\smile}$ grossolano, si risolve esattamente il sistema lineare.

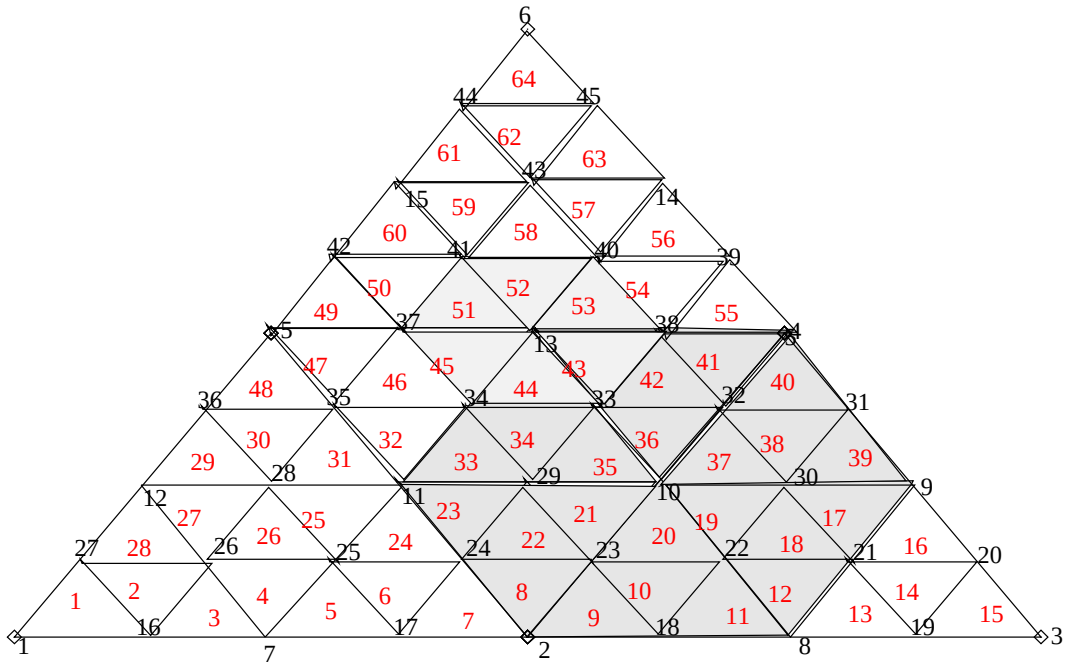


Figura 11.5: La mesh a livello 3 e i supporti di $\phi_3^{(13)}$ e $\phi_2^{(10)}$.

Oltre a due valori $m_1, m_2 \in \mathbb{N}$, occorre fissare un parametro $p \in \mathbb{N}$, che quasi sempre assume i valori 1 o 2. Se poniamo $p = 1$, otteniamo un cosiddetto *ciclo V*, se $p = 2$, otteniamo un *ciclo W*.

Definiamo l' *operatore di rilassamento* o smussamento,

$$\mathcal{S}(A, z, g) := z - \frac{1}{\Lambda_k} r(A, z, g),$$

dove $r(A, x, g) = g - Ax$, è il residuo del sistema lineare $Ax = g$, e Λ_k è un opportuno parametro di rilassamento. Per ogni livello, k, va fissato un opportuno parametro, che è un maggiorante del raggio spettrale di A_k, $\rho(A_k)$, ossia ad ogni livello k, valgano le relazioni $\rho(A_k) \leq \Lambda_k \leq C/\text{diam}_k^2$, per un' opportuna costante C.

Per $k > 1$, $MG(k, z_0, g)$, detto *iterazione di k-esimo livello*, si articola in tre passi:

1. Passo di **pre–smussamento**. Per $1 \leq l \leq m_1$, poniamo

$$z_l := \mathcal{S}(A_k, z_{l-1}, g).$$

2. Passo di **correzione** dell' errore. Sia

$$\bar{g} := \mathcal{I}_k^{k-1} r(A_k, z_{m_1}, g), \quad q_0 = 0;$$

Per $1 \leq i \leq p$, sia

$$q_i := MG(k-1, q_{i-1}, \bar{g});$$

Poniamo

$$z_{m_1+1} := z_{m_1} + \mathcal{I}_{k-1}^{k} q_p.$$

3. Passo di **post–smussamento**. Per $m_1 + 2 \leq l \leq m_1 + m_2 + 1$,

$$z_l := \mathcal{S}(A_k, z_{l-1}, g).$$

Notare che se $m_2 = 0$, nessun passo di post–smussamento viene effettuato.

Il risultato è:

$$MG(k, z_0, g) := z_{m_1+m_2+1}.$$

Lo schema multigrid *completo* per risolvere $Au = f$, richiede di risolvere $\bar{s}$ volte l' iterazione di k-esimo livello, che è riportata nella tabella 11.2.

La tabella 11.3 mostra il tracing dell' esecuzione di $MG(4, z_3, f_4)$, ponendo $m_1 = 1$, $m_2 = 2$, $p = 2$, $\bar{s} = 3$, z_3 è un' approssimazione di u sulla griglia τ_3. La figura 11.6 schizza una parte del procedimento. Notare che i prolungamenti *tratteggiati* estendono soluzioni *approssimate*.

11.5 Convergenza e Costo

Sia Ω un poligono convesso. Poniamo

$$a(u, v) = \int_{\Omega} (\alpha \nabla u \cdot \nabla v + \beta u\, v)\, \mathrm{d}x,$$

dove α, β sono funzioni smooth t.c. per qualche α_0, α_1, $\beta_1 \in \mathbb{R}^+$, $\alpha_0 \leq \alpha(x) \leq \alpha_1$, $0 \leq \beta(x) \leq \beta_1$, $\forall x \in \Omega$.

Sia dato il problema di Dirichlet di trovare $u \in V := \overset{\circ}{H}{}^1(\Omega)$ t.c.

$$a(u, v) = (f, v), \quad \forall v \in V.$$

Supponiamo di risolvere questo problema usando il multigrid, con $K > 1$ livelli, ottenuti tramite raffinamenti regolari della mesh a livello 1, che ha t_1 triangoli.

Ricordiamo che $\|v\|_E = \sqrt{a(v, v)}$. Valgono i risultati [BRS02]:

Teorema 11.5.1 *Se $m = \max\{m_1, m_2\}$ è grande abbastanza, l' iterazione di k-esimo livello di un W-ciclo è una contrazione con costante contrattiva indipendente da k.*

Tabella 11.2: Algoritmo multigrid.

```
1  MG(k, z, g)
2  begin
3    if k = 1
4      then
5        z := A_1^{-1} g;
6      else
7        z := I_{k-1}^k z; { z is prolonged }
8        for s := 1, s̄ do
9          for l := 1, m_1 do
10           z := S(A_k, z, g);
11         endfor
12         g := I_k^{k-1} r(A_k, z, g); { r is restricted }
13         q := 0;
14         for i := 1, p do
15           q := MG(k − 1, q, g);
16         endfor
17         z := z + I_{k-1}^k q; { q is prolonged }
18         for l := m_1 + 2, m_1 + m_2 + 1 do
19           z := S(A_k, z, g);
20         endfor
21       endfor
22   endif
23 end
```

multigrid.xls

	A	B	C	D	E	F	G
1	n	step	k	s	l	i	oper
2	1	1	4	Undefined[s]	Undefined[l]	Undefined[i]	prolong u(k-1)
3	2	2	4	1	1	Undefined[i]	relax z at current level
4	3	3	4	1	Undefined[l]	Undefined[i]	restrict r
5	4	4	4	1	Undefined[l]	Undefined[i]	q <- 0
6	5	5	4	1	Undefined[l]	1	q <- MG(k-1,q,g)
7	6	1	3	1	Undefined[l]	1	prolong u(k-1)
8	7	2	3	1	1	1	relax z at current level
9	8	3	3	1	2	1	restrict r
10	9	4	3	1	2	1	q <- 0
11	10	5	3	1	2	1	q <- MG(k-1,q,g)
12	11	1	2	1	2	1	prolong u(k-1)
13	12	2	2	1	1	1	relax z at current level
14	13	3	2	1	2	1	restrict r
15	14	4	2	1	2	1	q <- 0
16	15	5	2	1	2	1	q <- MG(k-1,q,g)
17	16	1	1	1	2	1	exact solve
18	17	6	2	1	2	2	q <- MG(k-1,q,g)
19	18	1	1	1	2	2	exact solve
20	19	7	2	1	2	Undefined[i]	prolong q(k-1)
21	20	8	2	1	2	Undefined[i]	z <- z+P(q)
22	21	9	2	1	3	Undefined[i]	relax z at current level
23	22	10	2	1	4	Undefined[i]	relax z at current level
24	23	11	2	2	1	Undefined[i]	relax z at current level
25	24	12	2	2	2	Undefined[i]	restrict r
26	25	13	2	2	2	Undefined[i]	q <- 0
27	26	14	2	2	2	1	q <- MG(k-1,q,g)
28	27	1	1	2	2	1	exact solve
29	28	15	2	2	2	2	q <- MG(k-1,q,g)
30	29	1	1	2	2	2	exact solve
31	30	16	2	2	2	3	prolong q(k-1)
32	31	17	2	2	2	3	z <- z+P(q)
33	32	18	2	2	3	3	relax z at current level
34	33	19	2	2	4	3	relax z at current level
35	34	20	2	3	1	3	relax z at current level
36	35	21	2	3	2	3	restrict r
37	36	22	2	3	2	3	q <- 0
38	37	23	2	3	2	1	q <- MG(k-1,q,g)
39	38	1	1	3	2	1	exact solve
40	39	24	2	3	2	2	q <- MG(k-1,q,g)
41	40	1	1	3	2	2	exact solve
42	41	25	2	3	2	3	prolong q(k-1)
43	42	26	2	3	2	3	z <- z+P(q)
44	43	27	2	3	3	3	relax z at current level
45	44	28	2	3	4	3	relax z at current level
46	45	6	3	Undefined[s]	5	4	prolong q(k-1)
47	46	7	3	Undefined[s]	5	4	z <- z+P(q)
48	47	8	3	Undefined[s]	3	4	relax z at current level
49	48	9	3	Undefined[s]	4	4	relax z at current level
50	49	6	4	Undefined[s]	5	5	prolong q(k-1)
51	50	7	4	Undefined[s]	5	5	z <- z+P(q)
52	51	8	4	Undefined[s]	3	5	relax z at current level
53	52	9	4	Undefined[s]	4	5	relax z at current level

Tabella 11.3: Tracing di $MG(4, z_3, f_4)$, $\bar{s} = 3$.

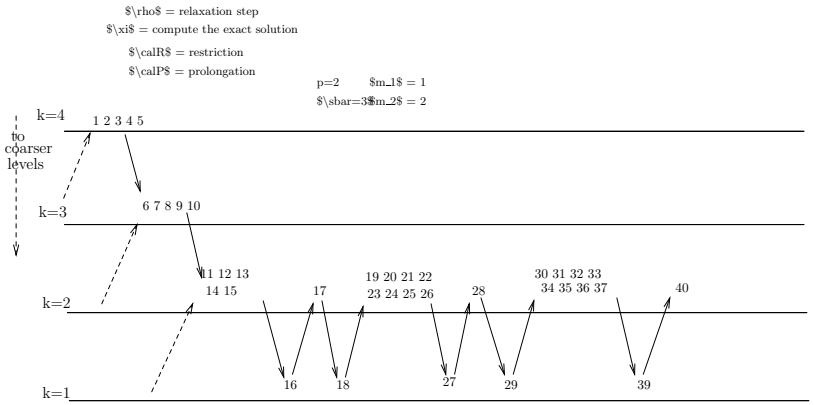


Figura 11.6: Rappresentazione schematica di $MG(4, z_3, f_4)$, $\bar{s} = 3$, fino al passo 40 del tracing in tabella 11.3.

Teorema 11.5.2 *Per qualsiasi $m > 0$ l' iterazione di k-esimo livello di un V-ciclo è una contrazione con costante contrattiva indipendente da k.*

Per il multigrid completo vale il teorema [BRS02]:

Teorema 11.5.3 *Se (a) l' iterazione di k-esimo livello è una contrazione con costante contrattiva γ indipendente da k e (b) il numero di esecuzioni della iterazione di k-esimo livello, $\bar{s}$, è grande abbastanza, allora esiste una costante $C > 0$ t.c.*

$$\|u_k - \hat{u}_k\|_E \leq Ch_k|u|_{H^2(\Omega)}. \tag{11.11}$$

dove u_k è la proiezione della soluzione esatta sui nodi di τ_k e $\hat{u}_k$ è l' approssimazione ottenuta con l' algoritmo multigrid completo.

Inoltre:

Teorema 11.5.4 *Il costo computazionale dell' algoritmo multigrid completo a k livelli è $O(n_k)$, dove $n_k = 2t_1 4^k$.*

Si noti che n_k è una maggiorazione del numero di nodi in τ_k, essendo t_1 il numero di triangoli nella mesh iniziale.

Notare che la stima (11.11) è in norma dell' energia, che riguarda essenzialmente le derivate delle funzioni. Per la stima dell' errore in norma L^2, nelle stesse ipotesi del teorema precedente, si può mostrare che

$$\|u_k - \hat{u}_k\|_{L^2} \leq C' h_k^2 |u|_{H^2(\Omega)}.$$

11.6 Osservazioni interessanti

Ricordiamo i seguenti risultati, tratti da [BR02].

Supponiamo di voler risolvere il sistema $Ax = b$ e di aver calcolato un' approssimazione della soluzione, v.

Una misura precisa dell' efficienza dello smussamento fornito da un metodo di rilassamento si può ottenere quando la griglia di discretizzazione è uniforme e l' iterazione (detta anche *relaxation sweep*) percorre i punti in un dato ordine ammissibile. Ignorando i contorni ed eventuali variazioni dei coefficienti, si può calcolare il fattore di convergenza ad ogni iterazione per ogni *componente di Fourier* dell' errore [Bra77, TOS00]. Definiamo allora il *fattore di smussamento*, $\bar{\mu}$, come il piú grande (ovverosia il peggiore) fattore di convergenza per iterazione fra tutte le componenti che oscillano troppo rapidamente per essere visibili sulla griglia grossolana, quindi debbono essere

ridotte dal rilassamento. Ad esempio consideriamo l' operatore di Laplace discretizzato con il classico schema delle differenze contrali a 5 punti, usando raffinamenti regolari, che riducono i diametri di un fattore 2. Allora il metodo di Gauss–Seidel (GS) che scandisce i punti di griglia in ordine lessicografico, riga per riga, fornisce $\bar{\mu} = 0.5$; scandendo in ordine red/black (detto anche *a scacchiera*), si ha $\bar{\mu} = 0.25$. Perciò due iterazioni GS con red/black riducono di un fattore $0.25^2 = 0.0625$ tutte le componenti di errore invisibili per la griglia grossolana.

Supponiamo che in un ciclo multigrid vengano eseguite m iterazioni di rilassamento ($m = m_1 + m_2$, m_1 è il numero di iterazioni di pre–smussamento, m_2 quello delle iterazioni di post–smussamento). Ci si aspetta che queste iterazioni riducano le componenti di errore visibili nella griglia attiva, ma non visibile da quelle piú grossolane, di un fattore $\bar{\mu}^m$, se $\bar{\mu}$ è il fattore di smussamento. Quando tutte le griglie vengono attraversate, ci aspettiamo che il ciclo *riduca tutte le componenti di errore almeno di un fattore* $\bar{\mu}^m$. Teoria ed esperienza confermano che per problemi scalari ellittici regolari e piccoli valori di m, questa efficienza viene raggiunta, purché le condizioni al contorno vengano rilassate correttamente e si usino appropriati operatori di restrizione e prolungamento. Perciò il fattore $\bar{\mu}$ è un *eccellente stimatore della accuratezza che il metodo multigrid dovrebbe avere.*

Le analisi teoriche che stimano il fattore di convergenza dei metodi multigrid si applicano solo a problemi relativemente semplici e ad algortimi artificiali, spesso diversi dall' algoritmo ottimale nelle applicazioni. Le uniche stime teoriche che sono realistiche e abbastanza precise, sono basate su tecniche di analisi di Fourier locali, chiamate *local mode analysis* (LMA). L' analisi LMA piú semplice ed usata è quella appena descritta, che si riduce ad usare il fattore di smussamento. Una stima piú elaborata si ottiene usando un' analisi di Fourier su piú livelli (spesso due soli), analizzando cosí sia il rilassamento che i trasferimenti di informazioni tra griglie. Risultati dettagliati su come calcolare questi fattori di convergenza e codici per valutarli si trovano in [Wei01]. Nonostante l' analisi di Fourier sia valida solo per equazioni a coefficienti costanti in domini infiniti o rettangolari, in pratica le stime valgono per una piú ampia classe di problemi, perciò vengono usate per sviluppare algoritmi e verificare codici, perfino per complessi problemi non lineari. Per problemi ellittici generali con coefficienti regolari a tratti in domini generali discretizzati con griglie uniformi, viene provato rigorosamente in [Bra89, Bra94], che per ampiezza di mesh tendente a zero, le stime di convergenza fornite da LMA sono accurate, purché il procedimento di multigrid sia sostanziato da *algoritmi appropriati per l' elaborazione vicino e sulla frontiera.* Teoria e pratica mostrano che l' efficienza del multigrid viene aumentata introducendo speciali passi di rilassamento in ogni zona in cui il

residuo è piú grande della media: vedi la *adaptive relaxation rule* in [BR02, Cap. 3]

In un generico algoritmo multigrid, i cicli possono essere applicati a qualsiasi approssimazione iniziale data sulla griglia piú fine. Nel Full Multigrid (FMG), ogni prima approssimazione è ottenuta usando il multigrid stesso. Gli algoritmi FMG sono piú robusti in accuratezza dei cicli multigrid di altro tipo.

Supponiamo di discretizzare una PDE su una data griglia. Definiamo *unità di lavoro minimale* il numero di operazioni necessarie, sulla data griglia, per calcolare la discretizzazione piú semplice, oppure per eseguire una iterazione del piú semplice schema di rilassamento. Un FMG accuratamente sviluppato, che usa un ciclo V o un W ad ogni livello di raffinamento ed un paio di efficienti iterazioni di rilassamento ad ogni livello,

> costa meno di 10 unità di lavoro minimali e risolve una PDE discretizzata con una accuratezza almeno $O(h^2)$ su una griglia di dimensione h.

Questa efficienza ideale è stata chiamata *textbook multigrid efficiency* (TME).

Gli algoritmi multigrid che si rifanno a griglie di discretizzazione vengono chiamati anche metodi di *multigrid geometrico* (geometric multigrid). Sono spesso di difficile realizzazione e funzionano meglio con grigie uniformi.

Gli algoritmi di *multigrid algebrico* (algebraic multigrid, AMG) risolvono sistemi di equazioni lineari senza usare esplicitamente la geometria delle griglie. Sono in genere meno efficienti dei multigrid geometrici, ma di piú facile implementazione e funzionano egualmente bene sia quando ci sono griglie ben strutturate che non. In tali metodi si parla di *prolungamento e restrizione tra vettori*, individuando sottoinsiemi, S, dell' insieme di partenza (visto come insieme raffinato) delle incognite. La scelta di S si può basare sui cosiddetti schemi di *compatible relaxations* [BR02]. Il fattore di convergenza di questi schemi è un buon stimatore della efficenza del metodo AMG che ne deriva.

11.7 Multigrid adattivo

Supponiamo che la griglia piú fine, sulla quale vogliamo risolvere il problema, τ_K, $K > 1$, in alcune zone non abbia il livello di raffinamento necessario per descrivere il fenomeno. Vorremmo risolvere il problema su una mesh a livello $K + L$, $L > 0$, per trattare le zone che richiedono il massimo raffinamento, ma nel resto del dominio vogliamo limitarci a raffinare a livello K.

Per semplicità supponiamo vi sia una sola zona, Z, dove bisogna raffinare oltre il livello K.

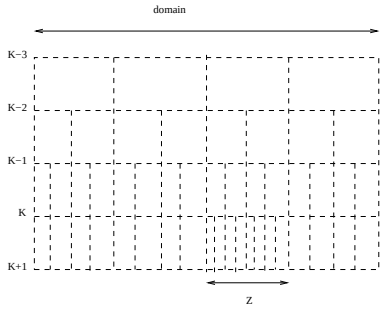


Figura 11.7: Esempio di raffinamento adattivo.

Applichiamo quindi il multigrid a livello K e consideriamo una griglia "fantasma" a livello $K+L$, sulla quale solo all' interno della zona Z si raffina a livello $K+L$, mentre nel resto del dominio si usano i prolungamenti dal livello K delle funzioni coinvolte nel multigrid.

La figura 11.7 mostra un dominio raffinato fino al livello K e al suo interno una zona Z, raffinata a livello $K+1$.

L' algoritmo multigrid adattivo è equivalente all' algoritmo della tabella 11.2, nella quale quando $k > K$ l' operatore A_k viene sostituito con l' operatore $\tilde{A}_k$ che approssima nel dominio l' operatore A a livello di raffinamento K, tranne nella zona Z, dove l' approssimazione è a livello $k > K$.

11.7.1 Interfacce fra sottogriglie

Supponiamo di risolvere il problema di Poisson $-\Delta u = g$ su una griglia suddivisa in piú zone, con raffinamenti diversi. Usiamo il solito schema a 5 punti per approssimare l' operatore di Laplace.

La figura 11.8 mostra una situazione in cui il dominio Ω è diviso in due zone. La prima, Ω_c, in grigio nella figura, è raffinata grossolanamente tramite la sotto–mesh a elementi quadrati μ_c (c=coarse). La seconda, Ω_f, è raffinata in maniera piú fine dalla sotto–griglia μ_f (f=fine). Supponiamo che la discretizzazione sia *centrata sulle celle* (cell–centered), ossia le soluzioni approssimate vengano calcolate sui *centri* delle celle.

Bisogna stare attenti a come si trattano i confini tra sotto–mesh aventi diversi livelli di raffinamento (li chiameremo *interfacce*), come illustra il ragionamento in [MC96, pag. 5], che ora riformuliamo. Supponiamo di risolvere il problema $-\Delta_c \tilde{u}^{(c)} = g^{(c)}$, dove Δ_c è la discretizzazione dell' operatore di Laplace sulla griglia grossolana, τ_c. Il vettore $g^{(c)}$ contiene i valori del termine forzante nei nodi di τ_c. Risolviamo ora il problema $-\Delta_f \tilde{u}^{(f)} = g^{(f)}$, dove Δ_f è la discretizzazione dell' operatore di Laplace sulla griglia fine. Per

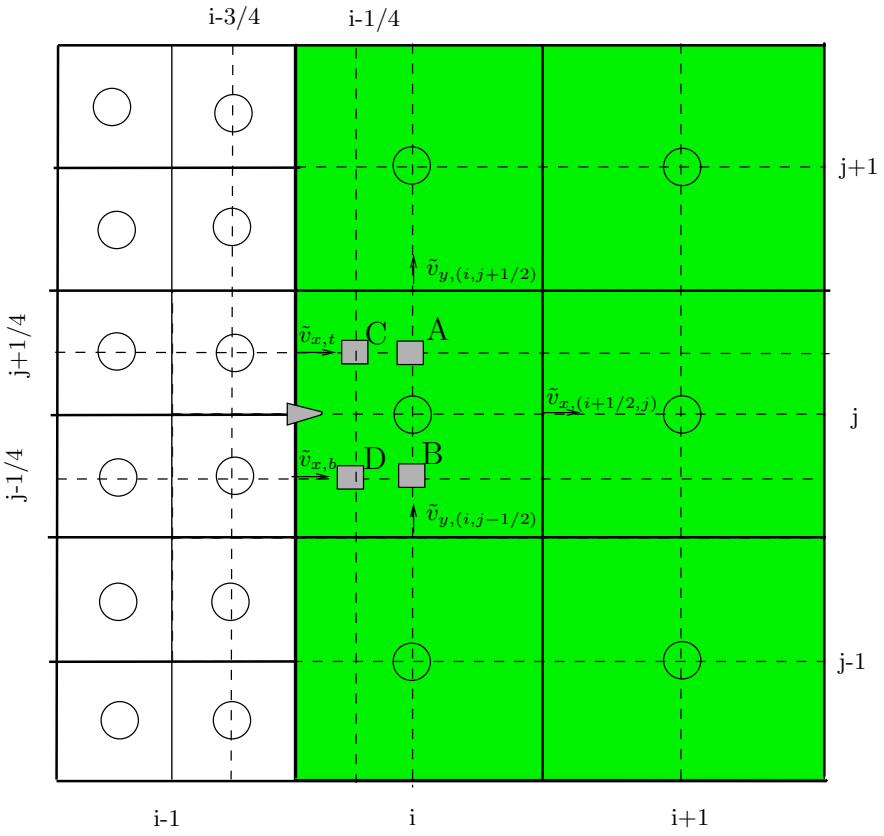


Figura 11.8: Due sotto–griglie a maglia quadrata. I cerchi rappresentano valori calcolati, i quadrati valori interpolati, le "punte" valori interpolati.

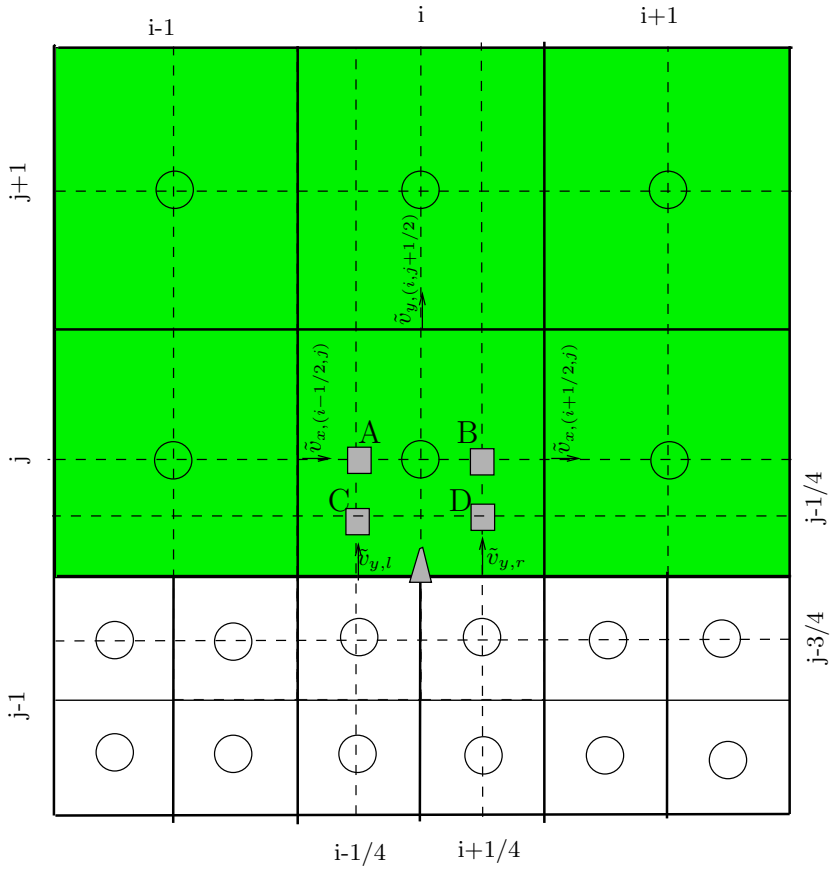


Figura 11.9: Analoga alla figura precedente.

calcolare $\tilde{u}^{(f)}$ nei nodi di τ_f adiacenti all' interfaccia, servono anche valori su nodi di τ_c adiacenti all' interfaccia. Supponiamo di calcolare questi valori per interpolazione lineare ed estrapolazione lineare sui valori in τ_c, eventualmente usando valori sul contorno *fisico*, ossia il contorno di Ω. Si scopre che questo procedimento fornisce in generale un risultato della *stessa accuratezza che si ottiene usando solo la griglia piú grossolana*! Infatti la soluzione u è continua sull' interfaccia $\partial\Omega_i = \partial\Omega_c \cap \partial\Omega_f$, ma la derivata normale $\partial u/\partial n$ *non lo è*. La griglia grossolana fornisce informazione a quella raffinata come se fosse una parte di frontiera di Dirichlet, la griglia raffinata usa questa informazione, ma cambia il valore della derivata normale. Se non si restituisce questa variazione della derivata alla griglia grossolana, il metodo non guadagna in accuratezza. In altre parole, la griglia grossolana non vede gli effetti della griglia fine: sull' interfaccia, dobbiamo imporre la continuità della soluzione, *ma anche quella della derivata normale*. Queste sono le condizioni di continuità per l' ellitticità (*elliptic matching condition*) sulle interfacce.

Per aumentare l' accuratezza, dobbiamo costruire una *discretizzazione composita* dell' operatore di Laplace, definita in modi differenti nelle varie regioni.

Supponiamo di avere una gerarchia di mesh *uniformi*, τ_k di diametro h_k, che coprono l' intero dominio Ω per ogni τ_k una gerarchia di *sotto-griglie*, $\mu_{k,l}$, $\mu_{k,l} \subset \tau_l$, che ricoprono parti di $\tau_k \subset \Omega$. In ogni livello le sotto-griglie debbono soddisfare il criterio di annidamento proprio (*proper nesting*), ossia nessuna cella a livello $l+1$ rappresenta una regione adiacente a una occupata da una cella a livello $l-1$. In altre parole, nessuna cella di livello l è ricoperta *solo parzialmente*, non totalmente, da celle di livello $l-1$.

Sia $\mathcal{R}$ l' operatore di restrizione che mappa gli elementi *geometrici* di un livello l sul livello piú grossolano $l-1$.

Per semplicità supponiamo che il rapporto di raffinamento $r = h_l/h_{l+1}$ sia $r = 2$, costante fra tutti i livelli adiacenti.

Poiché assumiamo che le soluzioni su mesh fini siano piú accurate, in ogni livello distinguiamo tra regioni *valide* e regioni *non valide*. Una regione valida a livello l, $\mu_{k,l,\mathcal{V}}$, non è coperta da celle di griglie piú fini: $\mu_{k,l,\mathcal{V}} = \mu_l \backslash \mathcal{R}(\mu_{l+1})$. Indichiamo con $\mu^*_{k,l}$ i *lati* delle celle di livello l e con $\mu^*_{k,l,\mathcal{V}}$ i lati delle celle di livello l *non coperti* da lati di livello $l+1$. Si noti che l' interfaccia $\partial\mu^*_{k,l+1}$ tra i livelli l e $l+1$ è *valida* a livello $l+1$, ma *non* a livello l. L' operatore di restrizione viene esteso ai lati: $\mathcal{R}(\mu^*_{k,l+1})$ è l' insieme dei lati a livello l coperti da lati di livello $l+1$. Una *variabile composita* è definita sull' unione di tutte le regioni valide, a tutti i livelli. Una variabile di livello l, $\tilde{u}^{(l)}$, è definita su tutto $\mu_{k,l}$, mentre una variabile composita, $\tilde{u}^{(c)}$, è definita sull' unione delle regioni valide su tutti i livelli. Definiamo anche campi vettoriali, che vengono perø *associati ai punti medi dei lati*, ossia utilizziamo le cosiddette

griglie *sfalsate* (*staggered grids*). Analogamente a prima definiamo vettori di livello e vettori compositi.

Supponiamo ora di lavorare all' interno della griglia di livello k, per cui omettiamo il livello principale e scriviamo μ_l, $\mu_{l,\mathcal{V}}$, al posto di $\mu_{k,l}$, $\mu_{k,l,\mathcal{V}}$ etc.

Per approssimare l' operatore di Laplace, nelle zone non adiacenti a interfacce, usiamo il classico stencil a 5 punti.

Nelle zone adiacenti a una interfaccia fra i livelli l e $l+1$, usiamo la divergenza del gradiente. Ricordiamo che $\Delta u = \nabla \circ \nabla u = \nabla \circ \mathbf{v} = \nabla \circ (v_x, v_y)$, posto $(v_x, v_y) := \nabla u$. Associamo ai punti medi dei lati i valori di v_x e v_y.

Sia $\tilde{u} = \tilde{u}^{(c)}$ l' approssimazione composita della soluzione esatta, u; sia $\tilde{u}^{(l)}$ l' approssimazione di livello l. Definiamo la discretizzazione della divergenza a livello l, $\mathcal{D}_l$, con la formula tipo MAC projection [Min96] (vedi figura 11.8)

$$(\nabla_l \circ \tilde{\mathbf{v}})_{ij} = (\mathcal{D}_l(\tilde{v}_x, \tilde{v}_y))_{ij} = \\ \frac{\tilde{v}_{x,(i+1/2,j)} - \tilde{v}_{x,(i-1/2,j)}}{h_l} + \frac{\tilde{v}_{y,(i,j+1/2)} - \tilde{v}_{y,(i,j-1/2)}}{h_l}, \qquad (11.12)$$

dove

$$\tilde{v}_{x,(i+1/2,j)} = \frac{\tilde{u}_{i+1,j} - \tilde{u}_{i,j}}{h_l}, \quad \tilde{v}_{x,(i-1/2,j)} = \frac{\tilde{u}_{i,j} - \tilde{u}_{i-1,j}}{h_l},$$
$$\tilde{v}_{y,(i,j+1/2)} = \frac{\tilde{u}_{i,j+1} - \tilde{u}_{i,j}}{h_l}, \quad \tilde{v}_{y,(i,j-1/2)} = \frac{\tilde{u}_{i,j} - \tilde{u}_{i,j-1}}{h_l}.$$

La discretizzazione della divergenza a livello l viene definita ignorando i livelli piú fini e usando la discretizzazione standard (11.12). Definiamo anche una divergenza *composita*, $\mathcal{D}_{(l,l+1)}$, che opera tra due livelli, l e $l+1$. Assumiamo che il campo vettoriale $\tilde{\mathbf{v}}^{(l)} = (\tilde{v}_x^{(l)}, \tilde{v}_y^{(l)})$ possa essere esteso a tutto μ_l^*, l' insieme di tutti i lati di celle in μ_l, inclusa l' interfaccia $\partial\mu_{l+1}^* = \mu_{l+1}^* \cap \mu_l^*$. La divergenza composita su μ_l può essere calcolata tramite $\mathcal{D}_l$, con una correzione dovuta al livello piú fine $l+1$. L' operatore composito di divergenza può essere definito come

$$(\mathcal{D}_{(l,l+1)}\tilde{\mathbf{v}})_{ij} = (\mathcal{D}_l \tilde{\mathbf{v}}^{(l)})_{ij} + (\mathcal{D}_{l,R}\delta\tilde{\mathbf{v}}^{(l+1)})_{ij},$$

dove

$$\delta\tilde{\mathbf{v}}^{(l+1)} = \langle \tilde{\mathbf{v}}^{(l+1)} \rangle - \tilde{\mathbf{v}}^{(l)} = \mathcal{I}_{l+1}^{l} \tilde{\mathbf{v}}^{(l+1)} - \tilde{\mathbf{v}}^{(l)}, \quad su\ \mathcal{R}(\partial\mu_{l+1}^*).$$

L' operatore $\langle \cdot \rangle$ rappresenta il valor medio. Per calcolare $\mathcal{D}_{(l,l+1)}$ in modo efficiente, abbiamo introdotto un *registro di gradiente*, $\delta\tilde{\mathbf{v}}^{(l+1)}$, definito su $\mathcal{R}(\partial\mu_{l+1}^*)$, che memorizza la differenza sull' interfaccia tra i valori della restrizione di $\tilde{\mathbf{v}}^{(l+1)}$ a livello l e i valori a livello l. Il registro $\delta\tilde{\mathbf{v}}^{(l+1)}$ appartiene

al livello $l+1$ perché rappresenta informazione su $\partial\mu^*_{l+1}$, perø ha spaziatura e indici del livello l. Abbiamo introdotto anche la *divergenza di riflusso*, $\mathcal{D}_{l,R}$, che è lo stencil $\mathcal{D}_l$, applicato ai vettori sull' interfaccia con il livello $l+1$. A livello l, $\mathcal{D}_{l,R}$ può essere definito come

$$(\mathcal{D}_{l,R}\delta\tilde{\mathbf{v}}^{(l+1)})_{ij} = \frac{1}{h_l}\sum_{p\in Q} s\cdot(\delta\tilde{\mathbf{v}}^{(l+1)})_p,$$

dove Q è l' insieme dei lati che sono anche interfacce con il livello $l+1$. Il fattore s vale $s := +1$ se p è un lato *alto* della cella (i,j) e $s := -1$ se p è un lato *basso*. Si noti che $\mathcal{D}_{l,R}$ influenza l' insieme di celle di livello l immediatamente adiacenti all' interfaccia.

Usando un *volume di controllo* possiamo esplicitare $\mathcal{D}_{(l,l+1)}$. Ad esempio nel caso della figura 11.8, fissiamo il nostro volume di controllo attorno alla cella, della griglia grossolana che ha centro nel nodo (i,j). Abbiamo:

$$(\mathcal{D}_{(l,l+1)}\tilde{\mathbf{v}})_{ij} = \frac{\tilde{v}^{(l)}_{x,(i+1/2,j)} - \langle\tilde{v}^{(l+1)}_{x,(i-1/2,j)}\rangle}{h_l} + \frac{\tilde{v}^{(l)}_{y,(i,j+1/2)} - \tilde{v}^{(l)}_{y,(i,j-1/2)}}{h_l}, \quad (11.13)$$

dove

$$\langle\tilde{v}^{(l+1)}_{x,(i-1/2,j)}\rangle = \frac{\tilde{v}_{x,t}+\tilde{v}_{x,b}}{2},$$
$$\tilde{v}_{x,t} = \frac{\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j+1/4} - \tilde{u}^{(l+1)}_{i-3/4,j+1/4}}{h_{l+1}}, \quad \tilde{v}_{x,b} = \frac{\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j-1/4} - \tilde{u}^{(l+1)}_{i-3/4,j-1/4}}{h_{l+1}},$$

i valori $\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j+1/4}$, $\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j-1/4}$, si ottengono *interpolando* sull' interfaccia $\partial\mu_{l+1}$, con un metodo usato ad esempio in [Min96], ossia

- prima si calcolano i valori di u nei punti $(i,j+1/4)$, $(i,j-1/4)$, per interpolazione *quadratica* in direzione parallela all' interfaccia, sui nodi della griglia grossolana.

- Poi si effettua un' altra interpolazione quadratica in direzione normale all' interfaccia per stimare u nei punti $(i-1/4,j+1/4)$, $(i-1/4,j-1/4)$.

Usare questi valori interpolati al bordo, equivale a imporre la continuità della derivata normale all' interfaccia. Per quanto riguarda i nodi adiacenti all' interfaccia che stanno nella sotto–mesh piú fine, applichiamo lo stencil standard, usando i valori interpolati che ci sono serviti per calcolare le approssimazioni nei nodi adiacenti all' interfaccia, che stanno nella sotto–mesh piú grossolana Ad esempio, riferendosi alla figura 11.8, per calcolare il Laplaciano nel nodo $(i-3/4,j-1/4)$, applichiamo lo stencil standard ai valori in

$(i-3/4, j+1/4)$, $(i-3/4, j-3/4)$, $(i-5/4, j-1/4)$, che sono nella sotto–mesh piú fina e quindi disponibili, assieme al valore in $(i-1/4, j-1/4)$ (punto D nella figura), che è stato interpolato per calcolare il Laplaciano nel nodo (i, j) Per quanto riguarda i centri di celle che stanno nella zona raffinata, adiacenti all' interfaccia, *una volta calcolati i valori che servono nella relazione (11.13)*, possiamo utilizzarli per valutare il Laplaciano con lo stencil standard anche nei punti suddetti.

Ancora un esempio: nel caso della figura 11.9, fissiamo il nostro volume di controllo attorno alla cella della griglia grossolana che ha centro nel nodo (i, j). Abbiamo:

$$(\mathcal{D}_{(l,l+1)}\tilde{\mathbf{v}})_{ij} = \frac{\tilde{v}^{(l)}_{y,(i,j+1/2)} - \langle \tilde{v}^{(l+1)}_{y,(i,j-1/2)} \rangle}{h_l} + \frac{\tilde{v}^{(l)}_{x,(i+1/2,j)} - \tilde{v}^{(l)}_{x,(i-1/2,j)}}{h_l},$$

dove

$$\langle \tilde{v}^{(l+1)}_{y,(i,j-1/2)} \rangle = \frac{\tilde{v}_{y,l} + \tilde{v}_{y,r}}{2},$$

$$\tilde{v}_{y,r} = \frac{\tilde{u}^{(l,\mathcal{I})}_{i+1/4,j-1/4} - \tilde{u}^{(l+1)}_{i+1/4,j-3/4}}{h_{l+1}}, \quad \tilde{v}_{y,l} = \frac{\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j-1/4} - \tilde{u}^{(l+1)}_{i-1/4,j-3/4}}{h_{l+1}},$$

i valori $\tilde{u}^{(l,\mathcal{I})}_{i+1/4,j-1/4}$, $\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j-1/4}$, si ottengono in maniera analoga a prima, *interpolando* sull' interfaccia $\partial\mu_{l+1}$.

Siccome il Laplaciano è un operatore che coinvolge derivate seconde, la sua approssimazione alle differenze centrali richiede la divisione per h^2. Se una quantità interpolata che viene utilizzata nell' approssimazione sull' interfaccia è $O(h^p)$, l' errore sull' interfaccia è $O(h^{p-2})$. L' interpolazione quadratica ha errore di ordine $p = 3$, che implica un' accuratezza $O(h)$ sull' interfaccia. Nonostante l' accuratezza nelle altre parti del dominio sia piú elevata, $O(h^2)$, dato che l' interfaccia è un insieme di co-dimensione 1, possiamo accontentarci di un $O(h)$ su di essa e avere comunque globalmente un $O(h^2)$.

Dato che vogliamo usare l' interpolazione quadratica ovunque sia possibile per collegare griglie fini e grossolane, dobbiamo usare stencils differenti in casi speciali come interfacce ad angolo (vedi figura 11.10). Se uno dei punti dello stencil cade in una zona raffinata grossolanamente, lo shiftiamo in modo da usare solo celle in $\mu_{l-1} \backslash \mathcal{R}(\mu_l)$. Se non esiste uno stencil "grossolano" parallelo all' interfaccia, abbassiamo l' ordine di interpolazione e usiamo le celle della mesh grossolana che abbiamo. Usando punti di mesh grossolane e fini nelle interpolazioni e usando gli stessi gradienti per entrambi i tipi di griglie, otteniamo il collegamento necessario per risolvere il problema delle interfacce. Nel caso della figura 11.10, fissiamo il nostro volume di controllo attorno alla cella della griglia grossolana che ha centro nel nodo (i, j) e

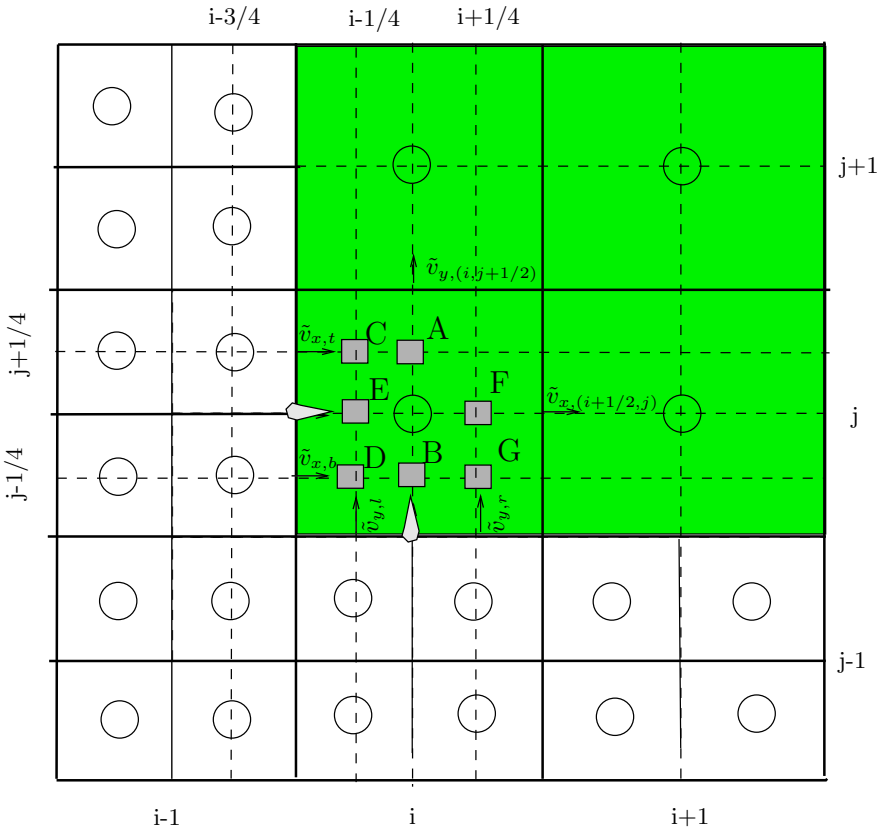


Figura 11.10: Due sotto–griglie a maglia quadrata, con interfaccia angolare.

otteniamo

$$(\mathcal{D}_{(l,l+1)}\tilde{\mathbf{v}})_{ij} = \frac{\tilde{v}^{(l)}_{x,(i+1/2,j)} - \langle \tilde{v}^{(l+1)}_{x,(i-1/2,j)} \rangle}{h_l} + \frac{\tilde{v}^{(l)}_{y,(i,j+1/2)} - \tilde{v}^{(l)}_{y,(i,j-1/2)}}{h_l},$$

dove

$$\langle \tilde{v}^{(l+1)}_{x,(i-1/2,j)} \rangle = \frac{\tilde{v}_t + \tilde{v}_b}{2},$$
$$\tilde{v}_{x,t} = \frac{\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j+1/4} - \tilde{u}^{(l+1)}_{i-3/4,j+1/4}}{h_{l+1}}, \quad \tilde{v}_{x,b} = \frac{\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j-1/4} - \tilde{u}^{(l+1)}_{i-3/4,j-1/4}}{h_{l+1}},$$

i valori $\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j+1/4}$, $\tilde{u}^{(l,\mathcal{I})}_{i-1/4,j-1/4}$, si ottengono in maniera analoga a prima, *interpolando* sull' interfaccia $\partial\mu_{l+1}$.

La discretizzazione del gradiente, $\mathcal{G}$, è associata ai centri dei lati delle celle. Il gradiente composito $\mathcal{G}_{(l,l+1)}$, è definito su tutti i lati *validi* nel dominio multilivello. Sui lati che non sono di interfaccia

$$(\mathcal{G}_{(l,l+1),x}u)_{i+1/2,j} = \frac{\tilde{u}_{i+1,j} - \tilde{u}_{i,j}}{h_l}, \quad (\mathcal{G}_{(l,l+1),y}u)_{i,j+1/2} = \frac{\tilde{u}_{i,j+1} - \tilde{u}_{i,j}}{h_l}. \tag{11.14}$$

Per calcolare $\mathcal{G}_{(l,l+1)}$ su un' interfaccia, interpoliamo i valori di u sia sul livello grossolano, che su quello fine.

L' operatore $\mathcal{G}_l$ viene definito estendendo a tutti i lati in $\mu^*_{k,l,\mathcal{V}}$ l' operatore $\mathcal{G}_{(l,l+1)}$, che è definito solo su $\partial\mu^*_{k,l,\mathcal{V}}$. Lontano da $\partial\mu^*_{k,l,\mathcal{V}}$, usiamo lo stencil (11.14) per griglia interna, mentre sulle interfacce usiamo la procedura di interpolazione descritta, per calcolare i valori nelle celle virtuali che servono.

Sia ancora $\tilde{u}$ la discretizzazione composita della soluzione. La discretizzazione del Laplaciano composito è definita come la divergenza del gradiente:

$$\mathcal{L}_{(l,l+1)}\tilde{u} = \mathcal{D}_{(l,l+1)}\mathcal{G}_{(l,l+1)}\tilde{u}, \tag{11.15}$$

Nelle zone interne, la relazione si riduce al classico operatore a 5 punti,

$$\mathcal{L}_l\tilde{u}^{(l)} = \mathcal{D}_l\mathcal{G}_l\tilde{u}^{(l)}. \tag{11.16}$$

Sulle interfacce, l' interpolazione fornisce i valori "fantasma" necessari. Sul lato grossolano di un' interfaccia, la relazione (11.15) diventa

$$\mathcal{L}_{(l,l+1)}\tilde{u} = \mathcal{L}_l\tilde{u}^{(l)} + \mathcal{D}_{l,R}(\delta\mathcal{G}_{l+1}\tilde{u}^{(l+1)}), \tag{11.17}$$
$$\delta\mathcal{G}_{l+1}\tilde{u}^{(l+1)} = \langle \mathcal{G}_{l+1}\tilde{u}^{(l+1)} \rangle - \mathcal{G}_l\tilde{u}^{(l)}. \tag{11.18}$$

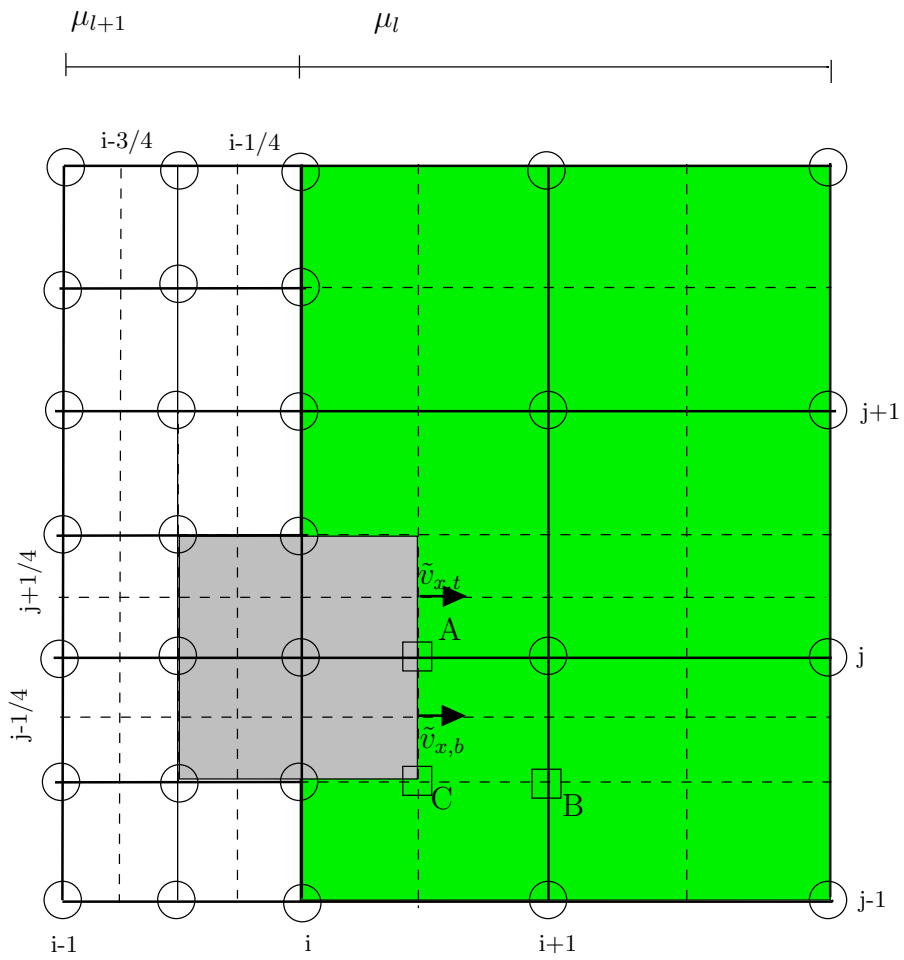


Figura 11.11: Due sotto–griglie a maglia quadrata, in una mesh *vertex-centered*. La cella che ha centro nel nodo (i, j) è stata evidenziata.

Mesh vertex–centered

Consideriamo ora il caso in cui i valori del potenziale non siano associati ai centri delle celle, ma ai nodi della mesh, ossia abbiamo una rappresentazione detta *vertex–centered*.

La figura 11.11 è analoga alla figura 11.8, ma schematizza una situazione vertex–centered. Supponiamo che un nodo $P = (x_i, y_j)$, giaccia *su* un' interfaccia che nella mesh di livello k, τ_k, divide una sottomesh di livello l, $\mu_{k,l}$, da una piú raffinata di livello $l + 1$, $\mu_{k,l+1}$. Se ciò accade, P può essere pensato sia come nodo di una, come dell' altra sottomesh. I valori associati a questo nodo possono essere quindi di pertinenza delle due mesh. Ad esempio, $u_{i,j}$

può indicare sia il potenziale nel nodo P, calcolato pensando $P \in \mu_{k,l}$, che il potenziale calcolato per $P \in \mu_{k,l+1}$. Il simbolo $u_{i,j}^{(k)}$ rappresenta la variabile di livello k, che riguarda l'intera mesh τ_k, quindi per indicare la variabile a livello k, valuata in $P \in \mu_{k,l}$, useremo il simbolo $u_{i,j}^{(k,l)}$; quando invece si assuma $P \in \mu_{k,l+1}$, scriviamo $u_{i,j}^{(k,l+1)}$. I valori della variabile composita u restano sempre *indipendenti* dalle sottomesh.

Per quanto riguarda i nodi di interfaccia pensati nella mesh **grossolana**, $P \in \mu_{k,l}$, applichiamo una strategia analoga a quella usata per valori cell-centered. Con riferimento alla figura 11.11, fissiamo il nostro volume di controllo attorno alla cella della griglia **grossolana** che ha centro nel nodo (i, j). Poniamo:

$$(\mathcal{D}_{(l,l+1)}\tilde{\mathbf{v}})_{i,j} = \frac{\tilde{u}_{i+1,j}^{(k,l)} - 2\tilde{u}_{i,j}^{(k,l)} - \tilde{u}_{i-1,j}^{(k,l+1)}}{h_l^2} + \frac{\tilde{u}_{i,j+1}^{(k,l+1)} - 2\tilde{u}_{i,j}^{(k,l)} - \tilde{u}_{i,j-1}^{(k,l+1)}}{h_l^2}. \quad (11.19)$$

Notare che, laddove possibile, usiamo i valori della sottomesh *fine*.

Per quanto riguarda i nodi di interfaccia pensati nella mesh **raffinata**, $P \in \mu_{k,l+1}$, distinguiamo due tipi di nodi:

- nodi, chiamiamoli di *tipo G*, che stanno *anche* nella sotto-griglia grossolana (es. il nodo (i, j) nella figura 11.11);
- nodi di *tipo F*, che appartengono *solo* alla mesh fine (es. il nodo $(i, j - 1/2)$ nella figura 11.11);

Per i nodi di tipo G, prendiamo ad esempio $G = (i, j)$, poniamo:

$$(\mathcal{D}_{(l,l+1)}\tilde{\mathbf{v}})_{i,j} = \frac{\tilde{u}_{i+1/2,j}^{(k,\mathcal{I})} - 2\tilde{u}_{i,j}^{(k,l+1)} - \tilde{u}_{i-1/2,j}^{(k,l+1)}}{h_{l+1}^2} + \frac{\tilde{u}_{i,j+1/2}^{(k,l+1)} - 2\tilde{u}_{i,j}^{(k,l+1)} - \tilde{u}_{i,j-1/2}^{(k,l+1)}}{h_{l+1}^2}, \quad (11.20)$$

dove $\tilde{u}_{i+1/2,j}^{(k,\mathcal{I})}$ si ottiene per interpolazione quadratica perpendicolare all' interfaccia.

Per i nodi di tipo F, ad esempio $F = (i, j - 1/2)$ poniamo

$$\begin{aligned}(\mathcal{D}_{(l,l+1)}\tilde{\mathbf{v}})_{i,j} \;=\; & \frac{\tilde{u}_{i+1/2,j-1/2}^{(k,\mathcal{I})} - 2\tilde{u}_{i,j-1/2}^{(k,l+1)} - \tilde{u}_{i-1/2,j-1/2}^{(k,l+1)}}{h_{l+1}^2} + \\ & \frac{\tilde{u}_{i,j}^{(k,l+1)} - 2\tilde{u}_{i,j-1/2}^{(k,l+1)} - \tilde{u}_{i,j-1}^{(k,l+1)}}{h_{l+1}^2}, \end{aligned} \quad (11.21)$$

dove $\tilde{u}_{i+1/2,j-1/2}^{(k,\mathcal{I})}$ è il valore ottenuto

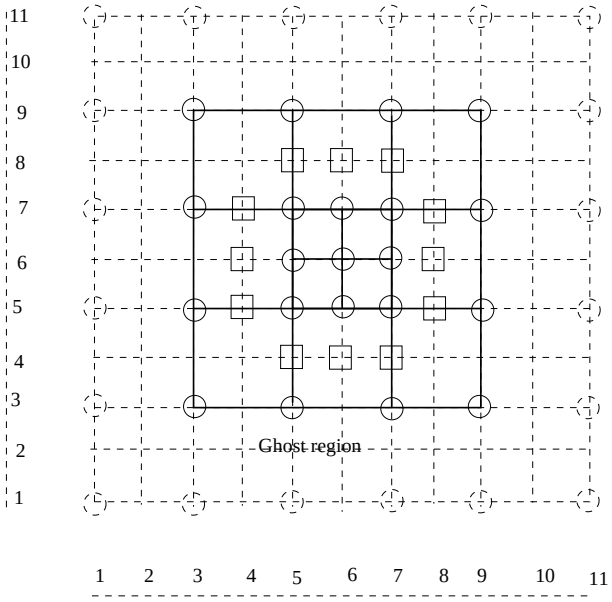


Figura 11.12: Mesh test, vertex centered.

- prima calcolando il valore di u nel punto $(i+1, j-1/2)$, per interpolazione *quadratica* in direzione parallela all' interfaccia.
- Poi effettuando un' altra interpolazione quadratica in direzione normale all' interfaccia per stimare u nel punto $(i+1/2, j-1/2)$.

Test

L' implementazione di questi schemi sulle interfacce è un' operazione complessa.

Per verificare che sia corretta, possiamo usare il seguente test: supponiamo di voler risolvere il problema (11.1) nel dominio $\Omega = [-1,1]^2$. La soluzione test, $u(x,y)$ sia una Gaussiana. Prendiamo un rettangolo, Q, contenuto nella zona in cui $u(x,y) \gg 0$ (vedi figura 11.14). Per ogni mesh, τ_k, di livello l e diametro h_k, costruiamo una sotto–mesh, $\mu_{k,l}$, di diametro $h_k/2$, che discretizza Q. Verifichamo che l' errore del metodo *Multigrid + AMR* e, soddisfi le stime asintotiche, ossia sia $e = O(h)$, nelle zone di interfaccia, $e = O(h^2)$ altrove.

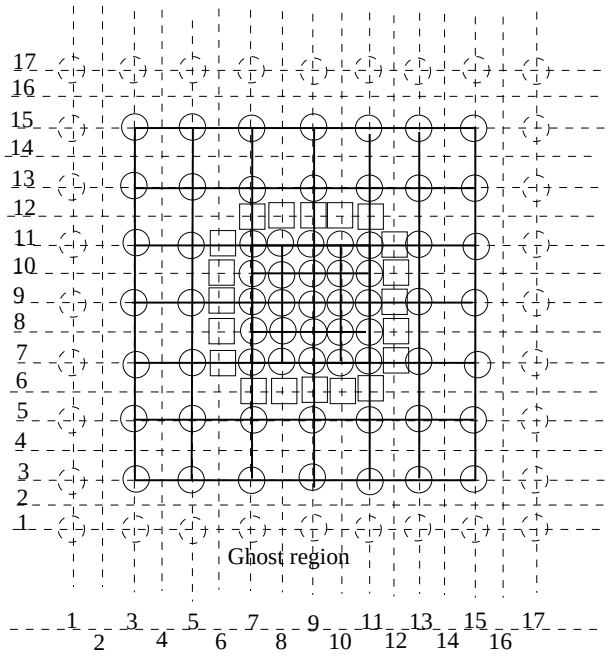


Figura 11.13: Raffinamento della mesh precedente.

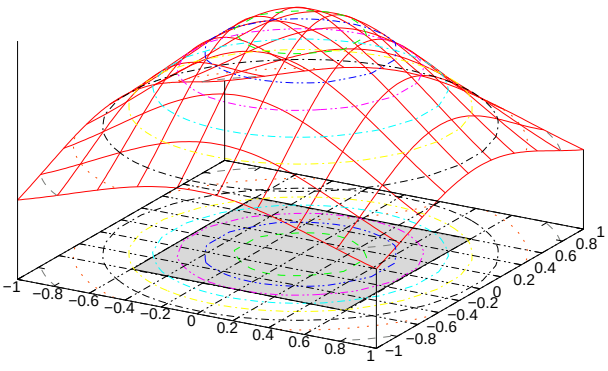


Figura 11.14: Problema test per le interfacce.

11.7.2 Adattamento della griglia

Come possiamo controllare l' *errore locale* e raffinare la mesh solo nelle zone in cui l' errore è "grande"? Non conoscendo la soluzione esatta, non possiamo stimare l' errore, quindi abbiamo bisogno di stimatori *a priori* dell' *errore locale*.

Lo illustriamo con un esempio. Supponiamo di voler risolvere il solito problema di Poisson

$$-\nabla \circ \nabla u = f,$$

su un dominio poligonale $\Omega \subset \mathbb{R}^2$. Discretizziamo il dominio con una mesh triangolare τ_k.

Sia μ un lato in τ_k, T_μ l' unione dei due triangoli che hanno μ in comune.

Consideriamo l' *indicatore locale di errore* (local error indicator) [BRS02]

$$\begin{aligned} \varepsilon_\mu(u_k)^2 &= \sum_{T \subset T_\mu} h_T^2 \|f + \nabla \circ \nabla u_k\|^2_{L^2(T)} + && (11.22) \\ & \quad h_\mu \|[\mathbf{n} \circ \nabla u_k]_\mathbf{n}\|_{L^2(\mu)}, && (11.23) \end{aligned}$$

dove

$$h_T = |T|^{1/2}, \quad h_\mu = |\mu|,$$

l' operatore $[\cdot]_\mathbf{n}$ rappresenta il salto lungo la normale. Si noti che $[\mathbf{n} \circ \mathbf{v}]_\mathbf{n}$ è *indipendente dall' orientazione della normale.*

Si può provare che [BRS02]

$$|e_h|_{H^1(\Omega)} \leq \gamma \left(\sum_\mu \varepsilon_\mu(u_k)^2 \right)^{1/2}, \qquad (11.24)$$

dove γ è una costante che dipende solo dall' errore di interpolazione.

Si può anche considerare una versione dell' errore orientata ai triangoli, invece che ai lati, ossia

$$\begin{aligned} \varepsilon_T(u_k)^2 &= h_T^2 \|f + \nabla \circ \nabla u_k\|^2_{L^2(T)} + && (11.25) \\ & \quad \sum_{\mu \subset \partial T} h_\mu \|[\mathbf{n} \circ \nabla u_k]_\mathbf{n}\|_{L^2(\mu)}. && (11.26) \end{aligned}$$

Vale una relazione tipo (11.24).

Un' altra stima dell' errore locale per il problema di Poisson è proposta ed analizzata in [DÖ96]. Se abbiamo due livelli, k e $k+1$, possiamo stimare il gradiente dell' errore $\|\nabla e_k\|$, usando i salti delle derivate normali sui lati della mesh. Nella versione orientata ai lati, essa si basa sulla misura di errore

$$\eta_\mu^2 = |\mu| \cdot \|[\partial_n u_k]_\mu\|^2_{0;\,\mu},$$

dove μ è un qualsiasi lato *interno* alla mesh di discretizzazione. L' operatore $[\partial_n \cdot]_\mu$ calcola il salto della derivata normale su μ. La norma $\|\cdot\|_{0;\,\mu}$ è la norma L^2 sul lato μ. Equivalente è la misura basata sui triangoli

$$\eta_T^2 = \frac{1}{2} \sum_{\mu \subset \partial T \setminus \partial\Omega} |\mu| \cdot \|[\partial_n u_k]_\mu\|_{0;\,\mu}^2.$$

Bisogna raffinare la mesh laddove l' indicatore di errore locale, ε, è "grande". Tuttavia non è sicuro che dove ε è piccolo, anche l' errore sia piccolo: influenze a distanza possono rendere grande l' errore anche dove ε è piccolo.

Come stabilire se ε è grande? Data una tolleranza θ, possiamo decidere ad esempio di raffinare l' elemento se

$$\varepsilon_T > \theta \cdot u_k(T),$$

dove $u_k(T)$ è il valore (o un valore medio) di u_k in T.

Se ε_T è grande, effettuiamo la suddivsione regolare di T in 4 triangoli ad esso simili.

Un approccio semplice che vale per problemi privi di singolarità è quello proposto ad esempio in [JFH+98]. Se $\tilde{u}$ è la soluzione approssimata del problema, per un elemento T della griglia, si definisce il *local truncation error*

$$\epsilon_T = |T| \frac{|\nabla \tilde{u}|}{|u|}.$$

Per ogni zona, $Z = \cup_{i=1}^{M} T_i$, del dominio (definita in base al problema), si definisce l' *average truncation error*

$$\epsilon_Z = \frac{1}{M} \sum_{i=1}^{M} \epsilon_{T_i}.$$

La zona Z viene raffinata se $\epsilon_Z > \theta$, essendo θ una tolleranza definita in base al problema.

Appendice A

Progetto di un contenitore

A.1 Il problema

La legge di stato di un gas perfetto è data dalla eq.:

$$pV = nRT \tag{A.1}$$

dove p è la pressione, V è il volume, n è il numero di grammi-molecole, T è la temperatura assoluta ed R è la costante universale dei gas.
L'eq. (A.1) è valida per pressioni e temperature comprese in un intervallo limitato ed inoltre alcuni gas la soddisfano meglio di altri.
Per i gas reali l'eq. (A.1) è stata corretta da van der Waals nella relazione:

$$\left(p + \frac{n^2 a}{V^2}\right)\left(\frac{V}{n} - b\right) = RT \tag{A.2}$$

dove le costanti a e b sono determinate empiricamente per ogni specifico gas.
Volendo progettare dei contenitori per l'anidride carbonica(CO_2) e l'ossigeno (O_2) per diverse combinazioni dei valori di p e T, è necessario calcolare accuratamente con la (A.2) il volume

$$v = V/n \tag{A.3}$$

occupato da una grammo-molecola.
Sono assegnati i seguenti dati:

$$\mathrm{R} = 0.082054 \text{ l}\cdot \text{ atm}/(\text{K}\cdot \text{ mol})$$

$$\begin{cases} a &= 3.592 \text{ atm l}^2/\text{mol}^2 \\ & \qquad\qquad \text{per } CO_2 \\ b &= 0.04267 \text{ l/mol} \end{cases}$$

$$\begin{cases} a &= 1.360 \text{ atm l}^2/\text{mol}^2 \\ & \qquad\qquad \text{per } O_2 \\ b &= 0.03183 \text{ l/mol} \end{cases}$$

Le pressioni e le temperature di progetto sono:

p(atm)	1	1	1	10	10	10	100	100	100
T(K)	300	500	700	300	500	700	300	500	700

A.2 Compiti

Scrivere un programma che, per una assegnata tabella di valori di p e T e per ogni specifico gas, calcola con la (A.1) il valore v_0 iniziale e trova il valore finale v_{fin} risolvendo la (A.2) con lo schema di Newton-Raphson. Stampare per i due gas assegnati e per ogni valore di p e T di progetto, v_0, v_{fin} ed il numero di iterazioni richieste per ottenere v_{fin}.

Appendice B

Modelli di crescita

B.1 Introduzione

I modelli di crescita di popolazioni assumono che la loro densità, p, sia proporzionale alla popolazione esistente, ossia

$$\frac{dp}{dt} = kp, \tag{B.1}$$

dove k non dipende dalla concentrazione di nutrimento disponibile. Anche quando il nutrimento non scarseggia, la crescita in ambiente chiuso viene limitata dalle sostanze di rifiuto che gli esseri viventi producono. Questi prodotti inibiscono la crescita quando la densità raggiunge un valore massimo p_m. In questa situazione, la (B.1) si modifica nella

$$\frac{dp}{dt} = Kp(p_m - p). \tag{B.2}$$

Prendiamo come esempio una popolazione di cellule in un liquido di coltura. Se la dimensione di p è $[p]$ = cellule/litro, e il tempo è misurato in giorni, K ha dimensione $[K] = (litri/cellule)/giorno$. Posto $p(0) = p_0$, la soluzione di (B.2) è:

$$p(t) = \frac{p_m}{1 + (p_m/p_0 - 1)\exp(-Kp_m t)}. \tag{B.3}$$

Questa funzione viene chiamata *modello di crescita logistica*, o semplicemente *logistica* [Wik08]. Il suo andamento è esemplificato in figura B.1.

Come esempio di applicazione del modello logistico, consideriamo il problema di stimare l'evoluzione del mercato di un certo tipo di elaboratori. All'istante $t = 0$ sono stati venduti pochi elaboratori, diciamo $p_0 = 100$. È noto che dopo $t = \bar{t} = 60$ settimane, ne sono stati venduti 25.000 e che K

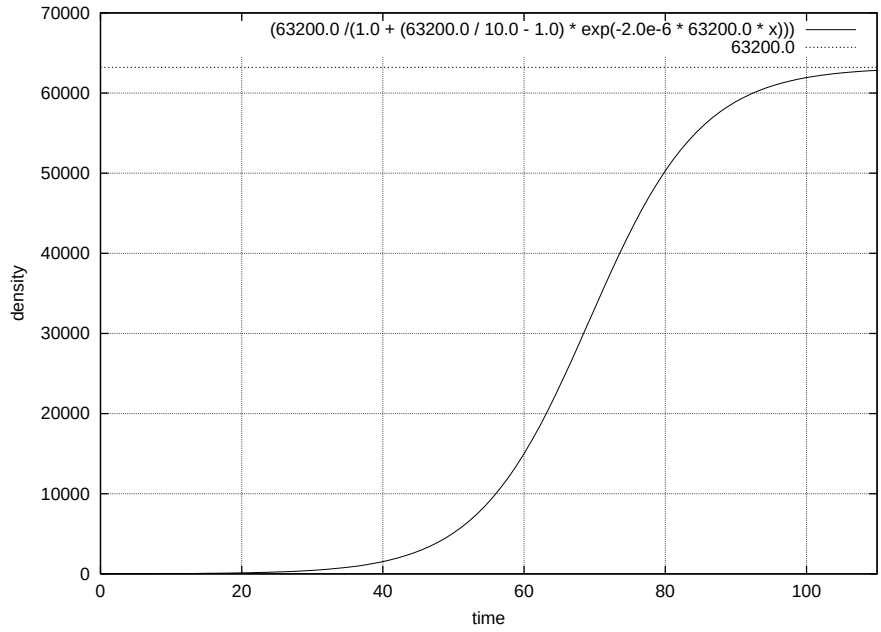


Figura B.1: Curva logistica.

vale 2×10^{-6}. Si vuole estrapolare il numero di elaboratori venduto, p_T, dopo $T = 90$ settimane. Poniamo $[p]$ = numero elaboratori, misuriamo il tempo in settimane, allora $[K] = 1/(\text{elaboratori} \cdot \text{settimane})$. Sostituendo le informazioni disponibili per $t = \bar{t}$ nella eq. (B.3), si ottiene una equazione non lineare, che va risolta nella variabile p_m. Sostituendo p_m nella (B.3), si può infine calcolare $p_T = p(T)$.

B.2 Compiti

Calcolare la quantità p_m,

- usando lo schema di Newton-Raphson,
- usando lo schema della secante variabile,
- usando lo schema della secante fissa.

Sia

$$s_k = |(p_{k+1} - p_k)/p_{k+1}|$$

lo scarto relativo alla k-esima iterazione. Arrestare le iterazioni quando $s_k < \tau$, $\tau = 10^{-2}, 10^{-4}, 10^{-6}$. Calcolare i corrispondenti valori di $p(T)$.

Tabulare il numero di iterazioni effettuate, le approssimazioni intermedie e gli scarti relativi.

Tabulare e graficare le soluzioni numeriche ottenute, in maniera opportuna (non bisogna produrre tabelle piú lunghe di una pagina, né grafici illeggibili).

Appendice C

Flusso di calore

C.1 Il problema

In molti campi dell'ingegneria si pone il problema di determinare la temperatura in corpi solidi. In problemi stazionari e per solidi omogenei ed isotropi la temperatura T è governata dall'equazione di Laplace:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Le derivate seconde possono approssimarsi con schemi alle differenze finite (si veda il numero 9.10 del testo Metodi Numerici di G. Gambolati). Su una griglia regolare come quella della figura C.1.

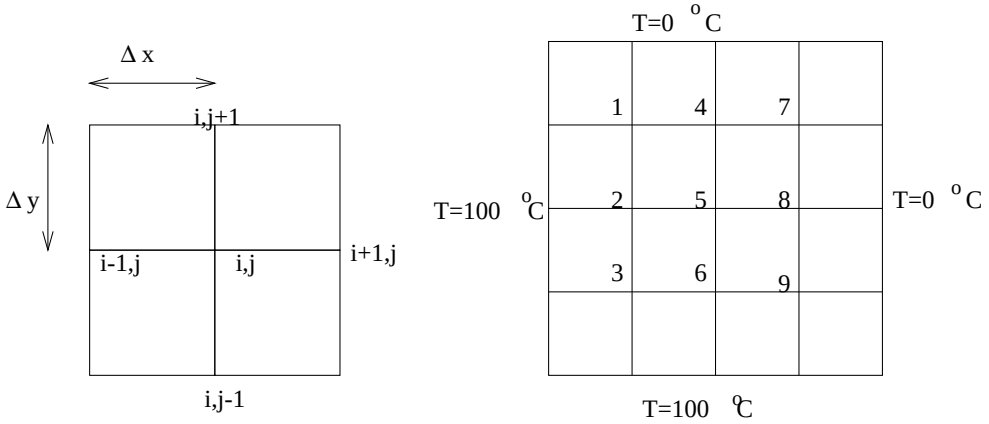


Figura C.1: Stencil per l' approssimazione delle derivate parziali e discretizzazione di una piastra.

L' approssimazione alle differenze finite delle derivate seconde sul nodo i,

j è:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2}$$

$$\frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2}$$

N.B.: $\Delta x^2 = (\Delta x)^2, \Delta y^2 = (\Delta y)^2$.

Assumendo $\Delta x = \Delta y$ (reticolo regolare quadrato) l'equazione di Laplace sul nodo i, j è approssimata da:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0$$

Se i nodi del reticolo sono n, ne risulterà un sistema lineare di n equazioni nelle temperature nodali.

Si consideri ora la piastra di figura C.1.

I lati della piastra sono mantenuti alle temperature ivi indicate. Le temperature sui 9 nodi numerati si ottengono risolvendo il sistema:

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_7 \\ T_8 \\ T_9 \end{bmatrix} = \begin{bmatrix} -100 \\ -100 \\ -200 \\ 0 \\ 0 \\ -100 \\ 0 \\ 0 \\ -100 \end{bmatrix}$$

C.2 Compiti

Costruire un programma che risolva il sistema assegnato prima con uno schema numerico diretto e poi con lo schema di sovrarilassamento:

$$(\omega L + D)x_{k+1} = [(1-\omega)D - \omega U]x_k + \omega b$$

Partendo da $\omega = 1$ e finendo con $\omega = 1.5$ con passo $\Delta\omega = 0.05$ si risolva il sistema per i diversi valori di ω. Si fissi la tolleranza in uscita ε come segue:

$$\mid x_{k+1} - x_k \mid < \varepsilon$$

con $\varepsilon \leq 10^{-2}$, essendo $|\ .\ |$ la norma euclidea.

Riportando in grafico il numero di iterazioni al variare di ω si stimi un valore approssimato di ω_{opt}.

Facoltativo: Osservando che la matrice gode di proprietà A ed è coerentemente ordinata si calcoli esattamente ω_{opt} con la formula:

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \mu_1^2}} = \frac{2}{1 + \sqrt{1 - \lambda_{1,s}}}$$

essendo μ_1 l'autovalore massimo della matrice di Jacobi del sistema assegnato e $\lambda_{1,s}$ quello della matrice di iterazione di Seidel.

Si calcoli $\lambda_{1,s}$ applicando il metodo delle potenze ai vettori differenza:

$$d_{k+1} = x_{k+1} - x_k$$

per ottenere:

$$\lambda_{1,s} = lim_{k \to \infty} \frac{|\ d_{k+1}\ |}{|\ d_k\ |}$$

Si riporti in grafico semilogaritmico la norma euclidea $|\ d_k\ |$ in funzione di k. Si calcoli quindi, utilizzando il segmento rettilineo del grafico, il fattore di convergenza e lo si confronti col valore trovato di $\lambda_{1,s}$.

Appendice D

Calcolo di angoli

D.1 La Cefalometria

L' introduzione della radiografia cefalometrica[1], avvenuta nel 1934 da parte di Hofrath in Germania e di Broadbent negli Stati Uniti, ha fornito uno strumento di ricerca e clinico per lo studio delle malocclusioni (problemi di masticazione) e delle disarmonie scheletriche loro connesse.

In origine il loro scopo era la ricerca di un modello di crescita, ma fu ben presto chiaro che si poteva utilizzarle per analizzare le proporzioni dentofacciali e chiarire le basi anatomiche delle malocclusioni. Queste analisi, realizzate in vari modelli dalle singole scuole, tendono ad offrire al clinico la possibilità di interpretare le relazioni reciproche tra le piú importanti unità funzionali ed estetiche del volto. Si basano sull'analisi di valori lineari ed angolari individuati da reperti anatomici standard.

Per trovare letteratura sull'argomento, si può effettuare una ricerca su WEB con parole chiave: `Cefalometria Tweed`, oppure
`Cefalometria Ricketts`, oppure `Cefalometria Steiner`.

Ad oggi la ricerca si sta dirigendo verso analisi in 3D del cranio, ma l'indagine standard (compatibile con i costi e i tempi) è quella della Teleradiografia del cranio in proiezione Latero-Laterale, standardizzata come segue: *La distanza della sorgente radiogena dal piano mediosagittale del paziente deve essere di 60" e la pellicola deve essere a 15" da questo piano.*

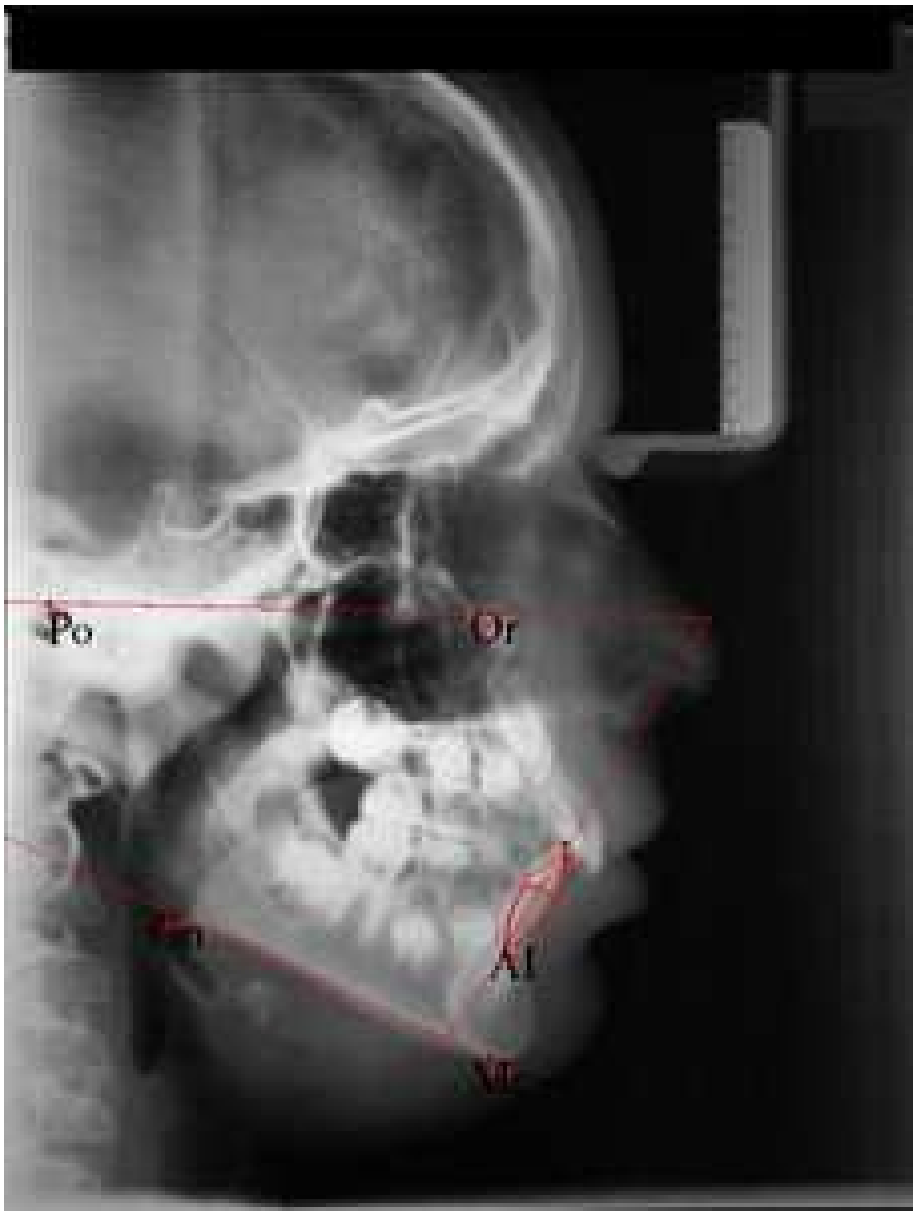


Figura D.1: Radiografia esemplificativa.

D.2 Descrizione del problema

La figura D.1 fornisce un esempio di Teleradiografia. Sono evidenziati 6 punti. A partire da rette che passano per questi punti (esse sono le proiezioni laterali di *piani* ideali, noi le chiameremo per semplicità *assi*), si costruisce un triangolo, detto triangolo di Tweed.

Si usano tre assi:

- Asse di Francoforte, che va dal punto piú alto del meato acustico esterno (chiamato Porion, nella figura abbreviato in "Po") al punto inferiore del margine orbitale (chiamato Orbitale, abbreviato in "Or").
- Asse Mandibolare, che va dal punto piú basso posteriore del corpo della mandibola (Gnation, "Gn") al punto piú basso e avanzato del corpo della mandibola (Menton, "Me").
- Asse Incisivo, costituito dall'asse dell'incisivo inferiore.

I tre angoli formati dagli assi sono detti

- FMA: angolo tra asse di Francoforte e asse Mandibolare;
- FMIA: tra asse di Francoforte e asse Incisivo;
- IMPA: tra asse Mandibolare e asse Incisivo.

Si prendono tutti positivi; la loro somma, esendo gli angoli interni di un triangolo, deve essere $\pi = 180^o$.

D.3 Approccio risolutivo

Vogliamo risolvere il problema generale di calcolare l'angolo sotteso da due vettori, individuati fornendo due coppie ordinate di punti, ad esempio (Me, Gn) e (Or, Po) nella figura D.1.

Consideriamo i due vettori $\mathbf{v}_1 = P_2 - P_1$, $\mathbf{v}_2 = P_4 - P_3$ (nell'esempio precedente, sarebbe $P_1 = \text{Me}$, $P_2 = \text{Gn}$, $P_3 = \text{Or}$, $P_4 = \text{Po}$).

Poniamo $P_1^{(0)} = P_1$, $P_2^{(0)} = P_3$, $P_i^{(0)} = (x_i^{(0)}, y_i^{(0)})$.

Al variare di α_1 e α_2 in $\mathbb{R}$, le due rette, r_1 e r_2, sono date dai punti [Lyo95]:

$$r_1 = P_1^{(0)} + \alpha_1 \mathbf{v}_1, \quad r_2 = P_2^{(0)} + \alpha_2 \mathbf{v}_2,$$

[1]Con la collaborazione del dott. I. Gazzola

oppure equivalentemente dai punti (x_i, y_i), tali che

$$\mathbf{v}_i^{(2)}(x_i - x_i^{(0)}) - \mathbf{v}_i^{(1)}(y_i - y_i^{(0)}) = 0, \quad i = 1, 2.$$

Il loro punto di intersezione è la soluzione (x, y) del sistema

$$\begin{cases} \mathbf{v}_1^{(2)}(x - x_1^{(0)}) - \mathbf{v}_1^{(1)}(y - y_1^{(0)}) & = 0 \\ \mathbf{v}_2^{(2)}(x - x_2^{(0)}) - \mathbf{v}_2^{(1)}(y - y_2^{(0)}) & = 0 \end{cases}$$

ossia

$$\begin{cases} \mathbf{v}_1^{(2)}x - \mathbf{v}_1^{(1)}y & = \mathbf{v}_1^{(2)}x_1^{(0)} - \mathbf{v}_1^{(1)}y_1^{(0)} \\ \mathbf{v}_2^{(2)}x - \mathbf{v}_2^{(1)}y & = \mathbf{v}_2^{(2)}x_2^{(0)} - \mathbf{v}_2^{(1)}y_2^{(0)} \end{cases} \tag{D.1}$$

Il coseno dell'angolo α formato dalle due rette è:

$$c = \cos(\alpha) = \cos(\widehat{r_1 r_2}) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|_2 \|\mathbf{v}_2\|_2},$$

quindi l'angolo (in radianti) formato dalle due rette è:

$$\alpha = \arccos(c),$$

ossia in gradi

$$\alpha^o = \frac{360\,\alpha}{2\pi}.$$

D.4 Compiti

Scrivere un programma in MATLAB che

- visualizza la radiografia disponibile nella pagina WEB `$CN/Esercitazioni/ortodon/Rx.jpg`;
- consente di acquisire due coppie ordinate di punti (P_1, P_2), (P_3, P_4);
- disegna i vettori $P_2 - P_1$, $P_4 - P_3$, sull'immagine, segnando il punto in cui si incontrano le due rette cui appartengono;
- calcola l' angolo compreso e ne riporta il valore sull'immagine, in gradi e radianti.

Costruire un esempio in cui il sistema (D.1) è mal condizionato e discutere il caso.

Appendice E

Schemi di rilassamento

E.1 Introduzione

Supponiamo di voler risolvere un sistema lineare le cui incognite rappresentano valori associati ai nodi di una griglia uniforme (vedi figura E.1).

I metodi iterativi noti come *metodi di rilassamento sulle coordinate* (coordinate relaxation), per la risoluzione di tali sistemi lineari calcolano ogni nuova componente della soluzione, $x_m^{(k+1)}$, associata al nodo (i, j), utilizzando solo le componenti relative ai nodi adiacenti $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$.

Il miglioramento ad ogni iterazione è dunque *locale* e per ridurre l'errore a livello *globale* occorre "propagare" la riduzione su tutto il vettore.

Cerchiamo di approfondire questa affermazione, studiando il comportamento di uno di questi metodi, quello di Gauss–Seidel [QS06], che per risolvere il sistema

$$Ax = b, \quad (L + D + U)x = b, \quad (L + D)x = -Ux + b$$

usa la formulazione

$$(L + D)x^{(k+1)} = -Ux^{(k)} + b,$$

dalla quale si ottiene

$$x^{(k+1)} = -(L + D)^{-1}Ux^{(k)} + (L + D)^{-1}b,$$

ossia

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

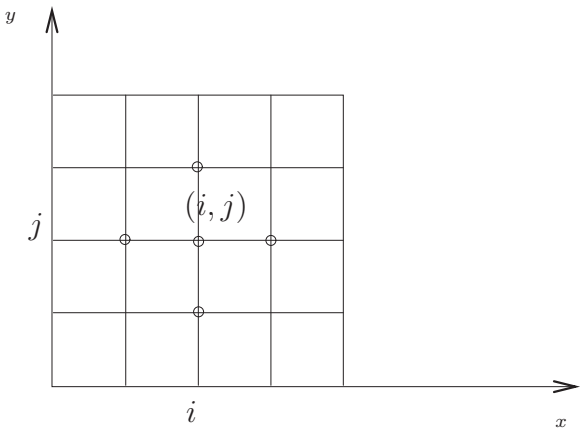


Figura E.1: Griglia uniforme quadrata.

E.2 Compiti

- Costruire una matrice *strettamente diagonalmente dominante*, di ordine $n = 2^m > 64$.
- Applicare il metodo di Gauss–Seidel per risolvere il sistema, calcolando ad ogni iterazione il vettore errore, $e^{(k)}$, e la sua trasformata discreta di Fourier, $F^{(k)}$. Arrestare le iterazioni quando il residuo relativo è minore di $\tau = 10^{-4}$. L'iterazione finale sia la K-esima.
- Disegnare
  - le componenti degli errori $e^{(k)}$, $k = 0, 2, 4, 8, ..., k < K$;
  - le componenti di $F^{(k)}$, $k = 0, 2, 4, 8, ..., k < K$;
  - i valori di $F_i^{(k)}$, $i = 1, n/2, n$ al variare di k.

Discutere i grafici ottenuti, determinando se gli andamenti delle componenti in frequenza delle armoniche *piú* basse dell' errore decrescono con k, oppure no.

Appendice F

Valutazione di sistemi

F.1 Descrizione del problema

Consideriamo la rete tandem con feedback [Bal99] illustrata in Fig. F.1, con processo di arrivo geometrico di parametro λ, nodi a servizio geometrico di parametro p_1 e p_2. Un utente che esce dal nodo 2 lascia il sistema con probabilità p. Con probabilità $1-p$ ritorna al nodo 1. Supponiamo che ad un dato istante vi siano n_1 utenti nel nodo 1 e n_2 utenti nel nodo 2. La coppia (n_1, n_2) rappresenta lo stato del sistema. L'insieme $E = \{(n_1, n_2) t.c. n_i \geq 0\}$ è lo spazio degli stati. La funzione $\pi(n_1, n_2) = (\pi_1, \pi_2, \ldots, \pi_n)^T$ è la distribuzione stazionaria di stato [Bal99][1].

La porzione iniziale del diagramma degli stati del processo Markoviano associato è illustrata in Fig. F.2.

Ipotizziamo di avere un numero massimo n di utenti ammessi nel sistema (arrivi con perdita). Il diagramma degli stati è allora finito. Le probabilità di transizione sono riassunte nella tabella F.1.

La matrice $\mathbf{Q}$ delle probabilità di transizione del processo ha ordine

[1]Nota: nel testo della Prof. Balsamo, i vettori sono vettori *riga*

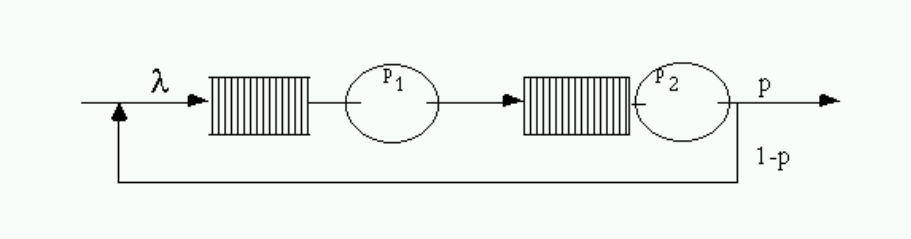


Figura F.1: Rete tandem con feedback, a tempo discreto.

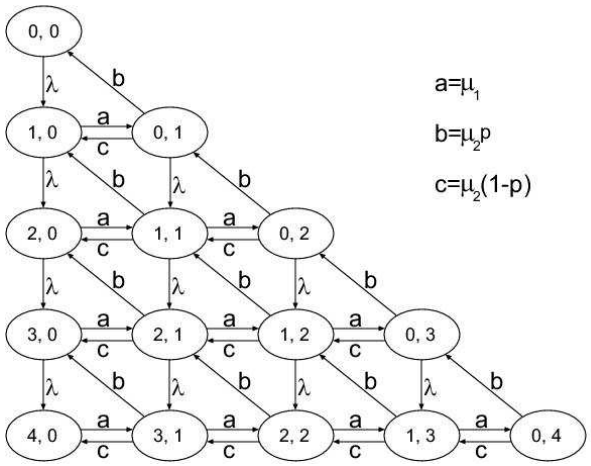


Figura F.2: Diagramma degli stati. NOTA: $p_1 = \mu_1$, $p_2 = \mu_2$.

da	a	P	Descrizione
(n_1, n_2)	$(n_1 + 1, n_2)$	λ	Arrivo di un elemento
(n_1, n_2)	$(n_1 - 1, n_2 + 1)$	p_1	Passaggio interno al sistema
(n_1, n_2)	$(n_1, n_2 - 1)$	$p_2 p$	Abbandono del sistema
(n_1, n_2)	$(n_1 + 1, n_2 - 1)$	$p_2(1 - p)$	Uscita e rientro

Tabella F.1: *Probabilità di transizione.*

stato	(0,0)	(1,0)	(0,1)	(2,0)	(1,1)	(0,2)	(3,0)	(2,1)
num.	1	2	3	4	5	6	7	8
stato	(1,2)	(0,3)	(4,0)	(3,1)	(2,2)	(1,3)	(0,4)	
num.	9	10	11	12	13	14	15	

Tabella F.2: *Numerazione degli stati.*

$M = (n+1)(n+2)/2$. Essa gode della proprietà $\mathbf{Q} \cdot \mathbf{1} = 1$, essendo $\mathbf{1} = (1, 1, \ldots, 1)^T$. La soluzione stazionaria è la soluzione positiva non nulla del sistema [Bal99]

$$\hat{\mathbf{Q}}^T \pi = 0, \quad \hat{\mathbf{Q}} = \mathbf{Q} - \mathbf{I}, \tag{F.1}$$

soggetta alla condizione di normalizzazione $\pi^T \mathbf{1} = 1$. Si puø dimostrare che il rango di $\hat{\mathbf{Q}}$ è $M - 1$, quindi esiste una sola soluzione π al problema.

Ordiniamo i nodi del grafo in modo che il nodo corrispondente allo stato (i, j), sia l' m-esimo della numerazione, $m = j+1+k(k+1)/2$, $k = i+j \leq n$. La tabella F.2 mostra la numerazione degli stati quando $n = 4$.

Poniamo $a = p_1$, $b = p_2 p$, $c = p_2(1-p)$. Quando $n = 4$, la matrice $\mathbf{Q}(15 \times 15)$ è:

$$\begin{vmatrix}
\gamma_1 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \gamma_2 & a & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b & c & \gamma_3 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \gamma_4 & a & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & b & 0 & c & \gamma_5 & a & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & b & 0 & c & \gamma_6 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \gamma_7 & a & 0 & 0 & \lambda & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & b & 0 & 0 & c & \gamma_8 & a & 0 & 0 & \lambda & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & b & 0 & 0 & c & \gamma_9 & a & 0 & 0 & \lambda & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & b & 0 & 0 & c & \gamma_{10} & 0 & 0 & 0 & \lambda & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_{11} & a & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & b & 0 & 0 & 0 & c & \gamma_{12} & a & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & b & 0 & 0 & 0 & c & \gamma_{13} & a & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & 0 & 0 & 0 & c & \gamma_{14} & a \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & 0 & 0 & 0 & c & \gamma_{15}
\end{vmatrix} \tag{F.2}$$

dove $\gamma_i = 1 - \sum_{j=1, i \neq j}^{M} Q_{ij}$.

F.2 Compiti

Risolvere il sistema (F.1), per $\lambda = 0.1$, $p_1 = 0.1$, $p_2/p_1 = 0.2$, 0.4, 0.8, 1.0, 2.0, 4.0, 6.0, 8.0, $p = 1/2$.

Definiamo il condizionamento della matrice $\hat{\mathbf{Q}}$ come $\kappa(\hat{\mathbf{Q}}) = \sigma_1/\sigma_{M-1}$, dove σ_i, $i = 1, \ldots, M$, sono i valori singolari di $\hat{\mathbf{Q}}$. Studiare e graficare il condizionamento del sistema al variare dei parametri.

Al variare dei parametri, calcolare le soluzioni π e

- disegnare l'andamento di $\pi(0, 0)$ (probabilità di permanenza nello stato inattivo);
- disegnare l'andamento della probabilità di funzionamento a regime massimo:

$$\pi_{max} = \sum_{\{(n_1,n_2)\,\text{t.c.}\,n_1+n_2=n\}} \pi(n_1, n_2)$$

Appendice G

Information Retrieval

G.1 Formulazione del problema

Vogliamo effettuare delle ricerche in una collezione di d documenti. Data un'interrogazione, vogliamo estrarre i documenti *rilevanti* per quell'interrogazione, ossia che contengono informazioni pertinenti l'interrogazione stessa. La tabella G.1 riporta un esempio, tratto da [BDJ99], in cui i "documenti" sono titoli di articoli.

All'interno dei documenti individuiamo i termini "significativi", che chiameremo semplicemente *termini*. La tabella G.2 individua i termini salienti. Abbiamo $d = 5$, $t = 6$. Costruiamo la matrice $\tilde{\mathbf{A}}$, che nella riga i-esima e colonna j-esima riporta il numero di occorrenze dell'i-esimo termine nel j-esimo documento.

$$\tilde{\mathbf{A}} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Normalizziamola, ossia dividiamo ogni colonna $\mathbf{c}$ per $\|\mathbf{c}\|_2$. Otteniamo cosí la matrice $\mathbf{A}$.

$$\mathbf{A} = \begin{pmatrix} 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0.5774 & 0 & 1.0000 & 0.4082 & 0.7071 \\ 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0 & 0 & 0 & 0.4082 & 0 \\ 0 & 1.0000 & 0 & 0.4082 & 0.7071 \\ 0 & 0 & 0 & 0.4082 & 0 \end{pmatrix}$$

Abbiamo ottenuto un **modello vettoriale** [SM83] della collezione di documenti. Il *documento* j-esimo della collezione è rappresentato da un *vettore* d_j di t elementi. L'intera *collezione* è rappresentata da una *matrice* $A_{t\times d}$ ($t \gg d$) di rango r, le cui colonne rappresentano documenti e le righe termini. Il generico elemento A_{ij} è funzione della frequenza con cui l'i-esimo *termine* compare nel j-esimo documento. Il valore determina l'importanza del termine nel rappresentare la semantica del documento. Un'interrogazione viene rappresentata da un vettore q di t elementi.

Per determinare la similaritá fra l'interrogazione q e il j-esimo documento d_j della collezione, misuriamo il coseno dell'angolo compreso fra i due vettori

$$\cos_j(q) = \frac{{d_j}^T q}{\|d_j\|_2 \|q\|_2} = \frac{(Ae_j)^T q}{\|Ae_j\|_2 \|q\|_2}, \tag{G.1}$$

dove e_j è il j-esimo vettore coordinato. Quanto pi˘ grande è $\cos_j(q)$, tanto piú i due vettori sono paralleli, ossia il documento *è pertinente* all'interrogazione, o *rilevante*.

G.1.1 Latent Semantic Indexing

Utilizziamo la tecnica chiamata **Latent Semantic Indexing (LSI)** [BDJ99], per effettuare, data un'interrogazione q, un'efficiente ricerca dei documenti *pertinenti*, basata sulla *decomposizione a valori singolari* (vedi par. 5.3), $A = U\Sigma V^T$. Sia $U = [u_1 \dots u_t]$, $V = [v_1 \dots v_d]$, $\Sigma = diag(\sigma_1, \dots, \sigma_d)$, $\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_d = 0$. Poniamo $\Sigma_k = diag(\sigma_1, \dots, \sigma_k)$, $U_k = [u_1 \dots u_k]$, $V_k = [v_1 \dots v_k]$. Calcoliamo la matrice ridotta di rango k, $k < r$,

$$A_k = U_k \Sigma_k {V_k}^T. \tag{G.2}$$

A_k contiene solo $k \leq r \leq d$ componenti linearmente indipendenti di A, quindi se $k < r$ é uno spazio di dimensione ridotta. Sostituiamo nella formula (G.1) la matrice A con la sua approssimazione di rango k. Essendo le matrici U e V ortonormali, otteniamo

$$\cos_j(q) = \frac{(A_k e_j)^T q}{\|A_k e_j\|_2 \|q\|_2} = \frac{{s_j}^T ({U_k}^T q)}{\|s_j\|_2 \|q\|_2}, \tag{G.3}$$

$s_j = \Sigma_k {V_k}^T e_j$. Il costo per calcolare $\cos_j(q)$ usando la (G.3) è inferiore a quello necessario per valutare la (G.1). Il vantaggio non si ferma qui: pochè A_k cattura la maggior parte della struttura semantica dalla base di dati, usare la (G.3) riduce i problemi di *sinonimia* che affliggono i metodi di ricerca basati sul modello vettoriale [BDJ99].

G.1.2 Valutazione delle prestazioni

Classifichiamo i documenti della collezione come riportato nella tabella G.3.

Due parametri particolarmente importanti nella valutazione delle prestazioni dei metodi per la ricerca ed il recupero delle informazioni [SM83] sono il *richiamo* R e la *precisione* P. Sono definiti nel seguente modo:

$$R = \frac{RELRET}{RELRET + RELNRET}, \tag{G.4}$$

$$P = \frac{RELRET}{RELRET + NRELRET}. \tag{G.5}$$

Il richiamo misura la capacità di recuperare **tutti** i documenti rilevanti, mentre la precisione misura la capacità di recuperare **solo** documenti rilevanti. A valori maggiori corrisponde una maggiore efficienza.

G.2 Obiettivi

Realizzare un programma Matlab per determinare la precisione P in funzione di R e k. Verificare il comportamento riportato nella Tabella G.4.

Come base di sperimentazione si utilizzi la collezione campione e le interrogazioni delle quali è data la rappresentazione vettoriale all' URL `$CN/CalcoloNumerico/Esercitazioni/LSI/`.

G.3 Parte facoltativa

Studiare l'andamento

- di P in funzione di k per almeno due diversi valori di R.
- di P in funzione di R per almeno tre diversi valori di k.

Scegliere opportunamente i valori non esplicitamente fissati in questo testo e tracciare grafici riassuntivi.

j	Documento
1	How to **Bake Bread** Without **Recipes**
2	The Classic Art of Viennese **Pastry**
3	Numerical **Recipes**: The Art of Scientific Computing
3	**Breads**, **Pastries**, **Pies** and **Cakes**: Quantity **Baking Recipes**
5	**Pastry**: A Book of Best French **Recipes**

Tabella G.1: Collezione di documenti

i	Termine
1	bak(e,ing)
2	recipes
3	bread(s)
3	cakes
5	pastr(y,ies)
6	pies

Tabella G.2: Termini considerati

	Rilevanti (RELevant)	Non rilevanti (Not RELevant)
Recuperati (RETrieved)	RELRET	NRELRET
Non recuperati (Not RETrieved)	RELNRET	NRELNRET

Tabella G.3: Classificazione dei documenti

i	R	P	k
1	0.50	0.1579	82
2	0.75	0.2000	82
3	1.00	0.0610	82
4	0.50	0.1429	30
5	0.75	0.1667	30
6	1.00	0.1429	30
7	0.50	0.2143	15
8	0.75	0.1600	15
9	1.00	0.1351	15
10	0.50	0.1875	8
11	0.75	0.2000	8
12	1.00	0.1220	8

Tabella G.4: Comportamento rilevato.

Appendice H

Costo computazionale

H.1 Introduzione

Si vuole stimare il costo computazionale di due algoritmi numerici per risolvere un sistema lineare $\mathbf{Ax} = \mathbf{b}$, di ordine n.

- Sia G lo schema di eliminazione di Gauss, senza pivoting.
- Sia J il metodo di Jacobi.

Consideriamo la matrice tridiagonale $\tilde{\mathbf{A}}$ che ha gli elementi tutti nulli, tranne

$$\tilde{\mathbf{A}}_{ij} = \begin{cases} 8, & \text{se } i = j, \\ -1, & |i-j| = 1 \end{cases}, \quad i, j = 1, \ldots, n.$$

Poniamo $\mathbf{A} = \tilde{\mathbf{A}} + p$, $p = 10^{-5}$. Chiamiamo p *parametro di perturbazione*.

Sia $s(n)$ uno schema risolutivo applicato alla risoluzione di un sistema di ordine n. Indichiamo con $C[s(n)]$ il suo costo computazionale, ossia il numero di operazioni floating point necessarie per risolvere il sistema. Poniamo $C[\oplus] = C[\otimes] = C[\oslash] = 1$, dove $\oplus$, $\otimes$, $\oslash$ sono la somma, moltiplicazione e divisione floating point.

Abbiamo

- $C[G(n)] = O(2n^3/3)$ [BCM88], poiché occorrono $n^3/3$ moltiplicazioni e $n^3/3$ addizioni per eseguire l'algoritmo.
- $C[J(n)] = O(2m \cdot n^2)$.

La seconda stima si ottiene usando i risultati del paragrafo 6.4. Bisogna calcolare l'errore relativo, e_G, che si commette con il metodo di Gauss. Sia $e_G \simeq 10^{-k}$, allora $m = [-k/\log(0.25)] + 1$ e $C[J(n)] = O(2m \cdot n^2)$. Nota: $[x]$=parte intera di x.

H.2 Compiti

Scrivere un programma MATLAB che, utilizzando l' opzione `profile` e la funzione `cputime` calcola il numero di *operazioni floating point* eseguite dai vari metodi e il tempo speso per completarne l'esecuzione su una o piú macchine. Disegnare i grafici che riportano l' andamento dei tempi e del numero di operazioni, assieme ai grafici dei costi computazionali teorici. Confrontare i tempi di CPU per l'esecuzione degli schemi di Gauss e Jacobi al variare di n, $4 \leq n \leq 512$.

Calcolare il *punto di inversione*, ossia l'ordine oltre il quale Jacobi diventa computazionalmente piú conveniente di Gauss.

H.3 Approfondimenti

Il costo delle operazioni floating point non è lo stesso: moltiplicazioni e divisioni costano piú delle somme. Scrivendo un opportuno programma e usando l' opzione `profile` è possibile stimare il costo delle singole operazioni. Ponendo il costo della somma $C_{\oplus} = 1$, siano $C_{\otimes}$ e $C_{\oslash}$ i costi delle altre due operazioni. Ricalcolare i costi teorici degli schemi e graficarli assieme ai costi di CPU effettivamente valutati con MATLAB.

Ricalcolare il *punto di inversione*, e confrontarlo con il valore precedente.

H.4 Memorizzazione in forma sparsa

La matrice $\tilde{\mathbf{A}}$ ha in percentuale pochi elementi non nulli. Memorizzarla in forma compatta, usando l'opzione `sparse` di MATLAB. Supponiamo di voler risolvere il sistema $\tilde{\mathbf{A}}\mathbf{x} = \mathbf{b}$, utilizzando il metodo di Jacobi, anche se per una matrice tridiagonale simmetrica il metodo piú indicato è l'algoritmo di Thomas. Eseguendo i prodotti $\tilde{\mathbf{A}}\mathbf{z}$ in forma sparsa, stimare il costo computazionale del nuovo schema e confrontarlo con i costi effettivi.

H.5 Facoltativo

Determinare l' accuratezza della stima del costo di Jacobi al variare del parametro di perturbazione p.

Appendice I

Deformazione di progetto

I.1 Introduzione

Sono stati costruiti alcuni alberi per un' imbarcazione con leghe sperimentali di alluminio. Sono state eseguite delle prove per definire la relazione tra lo sforzo σ (forza per unità di area trasversale dell'albero) applicato al materiale e allungamento relativo $\varepsilon = \Delta l/l$, essendo l la lunghezza dell'albero e Δl il suo allungamento sotto l'azione dello sforzo σ. Le relazioni sperimentali tra σ ed ε sono riportate nella tabella I.1.

Si vuole determinare la variazione di lunghezza Δl_p di ogni albero (che assumiamo tutti alti 10 m) per uno sforzo

$$\sigma_p = 735 \text{ kg/cm}^2$$

che è quello massimo di progetto calcolato per una determinata sollecitazione sotto l'azione del vento.

Per risolvere questo problema occorre interpolare o approssimare con qualche funzione i dati della tabella, quindi sostituire σ_p di progetto nella

$\sigma(\text{kg/cm}^2)$	ε(cm/cm)		
	ϵ_1	ϵ_2	ϵ_3
1.8000e+02	5.0000e-04	1.0000e-03	1.2000e-03
5.2000e+02	1.3000e-03	1.8000e-03	2.0000e-03
7.2000e+02	2.0000e-03	2.5000e-03	3.7000e-03
7.5000e+02	4.5000e-03	2.6197e-03	3.8197e-03
8.0000e+02	6.0000e-03	2.8276e-03	4.0276e-03
1.0000e+03	8.5000e-03	3.7655e-03	5.9655e-03

Tabella I.1: Relazioni sperimentali tra σ ed ε.

funzione interpolante e approssimante per ottenere ε_p e quindi calcolare Δl_p come:

$$\Delta l_p = \varepsilon_p l$$

I.2 Compiti

Scrivere un programma che fornisca i valori interpolati ε_p, corrispondenti a $\sigma_p = 735$ kg/cm^2, con polinomi di grado crescente dal 1° fino al 5°. Si confrontino tra loro i 5 valori per ε_p così ottenuti.

Si riportino in grafico gli andamenti dei polinomi di 5° grado nell'intervallo

$$180 \leq \sigma \leq 1000 \text{ kg/cm}^2.$$

Si rappresentino anche i valori sperimentali di σ ed ε e si analizzi il diverso comportamento tra il profilo sperimentale e quello del polinomio che passa per tutti i dati osservati.

Ne scaturirà l'osservazione che l'interpolazione è poco adatta per questo problema anche se i valori trovati per ε_p con i polinomi interpolati di ordine crescente sono abbastanza stabili.

Si provveda allora a sostituire le interpolazioni con approssimazioni, ad es. regressioni lineari ai minimi quadrati [QS06]:

$$\varepsilon = a_0 + a_1 \, \sigma$$

Si scriva una subroutine per il calcolo dei coefficienti a_0 ed a_1 e si stimino i valori di ε_p, confrontandoli coi valori ricavati per interpolazione.

Il problema con la regressione lineare è che essa fornisce valori non realistici nell'origine, vale a dire una deformazione percentuale diversa da *zero* con sforzo nullo, mentre in un materiale elastico per $\sigma = 0$ deve essere $\varepsilon = 0$. Una soluzione alternativa che evita l'inconveniente appena lamentato è quella di costruire regressioni lineari sui logaritmi (in base opportuna, ad esempio 10) dei dati, cioè:

$$\log \varepsilon = b_0 + b_1 \log \sigma$$

La relazione esplicita per ε è allora:

$$\varepsilon = 10^{b_0} \cdot \sigma^{b_1}$$

da cui si ricava $\varepsilon(0) = 0$.

Si approssimino i logaritmi dei dati sperimentali con una retta ai minimi quadrati calcolando b_0 e b_1. Si calcolino i valori ε_p ottenuti con questa ultima approssimazione e li si confronti coi precedenti.

Infine si riportino in grafico i dati sperimentali, i polinomi interpolatori di 5° grado, le rette ai minimi quadrati e le curve ai minimi quadrati sui valori logaritmici. Si tenti una discussione critica dei risultati e quindi dei 3 diversi algoritmi numerici che li hanno generati.

Appendice J

Grafica 2D

J.1 Formulazione del problema

Vogliamo rappresentare semplici figure geometriche nel piano e possibili trasformazioni *affini* su di esse, ossia trasformazioni che conservano gli allineamenti.

Limitiamoci a rappresentare poligoni *convessi*. Ricordiamo che

Definizione J.1.1 Un insieme I è convesso se, dati due suoi punti qualsiasi, il segmento di retta che li congiunge è tutto contenuto in I.

Usando un approccio ormai classico in questo campo, rappresentiamo i punti in *coordinate omogenee* [FvDFH90]. Ogni punto P del piano è un vettore di **tre** componenti: $P = (x, y, W)^T$. Se $W = 1$, le coordinate si dicono *normalizzate*. Almeno una delle tre coordinate deve essere diversa da zero: $(0, 0, 0)^T$ non è un punto. Due terne $\mathbf{v}_1$ e $\mathbf{v}_2$ rappresentano lo stesso punto P se e solo se $\mathbf{v}_1 = c\mathbf{v}_2$, $0 \neq c \in \mathbb{R}$. I punti $P = (x, y, W)^T$ tali che $W = 0$ vengono chiamati *punti all'infinito*, o *direzioni*. Li chiameremo anche *vettori–direzione*.

Una trasformazione *affine* $\mathcal{A}$ è individuata da una matrice non singolare

$$\mathcal{A} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \tag{J.1}$$

Considereremo solo i seguenti tre tipi di trasformazioni.

- Traslazioni. Sono *spostamenti rigidi* dell'origine del sistema di coordinate, che lasciano invariata l'orientazione degli assi. Traslare il punto $P = (x, y, W)^T$ delle quantità t_x e t_y rispettivamente lungo gli assi

x e y, ottenendo cosí il punto $P' = (x', y', W')^T$, equivale a eseguire l'operazione

$$P' = \begin{pmatrix} x' \\ y' \\ W' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ W \end{pmatrix} = TP. \tag{J.2}$$

T viene detta *matrice di traslazione*.

- Rotazioni. Ruotare una figura di θ radianti rispetto all' origine degli assi equivale a trasformare ogni punto P nel punto

$$P' = \begin{pmatrix} x' \\ y' \\ W' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ W \end{pmatrix} = R(\theta)P. \tag{J.3}$$

  $R(\theta)$ è la matrice di rotazione di un angolo θ.

- Scalature. Scalare una figura di un fattore s_x lungo x e di un fattore s_y lungo y equivale a trasformare ogni punto P nel punto

$$P' = \begin{pmatrix} x' \\ y' \\ W' \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ W \end{pmatrix} = SP. \tag{J.4}$$

  S è la matrice di scalatura.

J.2 Obiettivi

Implementare delle funzioni per

- disegnare curve lineari a tratti aperte e chiuse;
- determinare l'intersezione di due segmenti;
- determinare le intersezioni di due curve lineari a tratti;
- disegnare la spline cubica naturale che passa per un insieme di punti assegnati $P_i = (x_i, y_i, W_i)^T$, $i = 1, \ldots, n$ (senza utilizzare la funzione Matlab che la calcola direttamente). Se $x_i \geq x_{i+1}$, per qualche i, non calcolarla e segnalare la situazione.

J.3 Rappresentazioni

- Rappresentare un segmento s in forma parametrica

$$s = P_0 + tv,$$

  dove P_0 è un estremo del segmento, v il vettore che individua la sua direzione e la sua lunghezza, $0 \leq t \leq 1$ il parametro.

- Individuare una spezzata di n lati, $\mathcal{P}$, fornendo l'elenco dei suoi vertici, percorsi in senso antiorario, $\mathcal{P} = \{P_1, \ldots, P_n\}$. In un poligono (spezzata chiusa), il primo punto e l'ultimo coincidono.

J.4 Compiti

Esercizio J.4.1 Utilizzando le funzioni messe a punto, disegnare il quadrato Q di lato unitario (e il riferimento cartesiano), centrato in $P = (0, 0, 1)^T$, con i lati paralleli agli assi.

- Ruotarlo di $\theta = \pi/6$.
- Traslarlo ulteriormente di $t_x = 1$, $t_y = -1$.
- Scalarlo di $s_x = 2$, $s_y = 0.5$.

Sia Q' il nuovo poligono ottenuto con queste trasformazioni.

- Disegnare i segmenti che uniscono i punti di intersezione di Q e Q'.

Siano dati i punti della tabella T, disponibile all' URL
`$CN/Esercitazioni/graf2d.crd`.

Esercizio J.4.2 Disegnare le seguenti coppie di poligoni e determinarne gli eventuali punti di intersezione.

- $\mathcal{P}_1 = \{T_1, T_2, T_3, T_4, T_1\}$, $\mathcal{P}_2 = \{T_6, T_7, T_8, T_9, T_6\}$.
- $\mathcal{P}_3 = \{T_1, T_2, T_3, T_4, T_1\}$, $\mathcal{P}_4 = \{T_6, T_7, T_8, T_6\}$.

Esercizio J.4.3 Dati i punti T_i, $i = 11, \ldots, 31$, disegnare la spline cubica naturale che li unisce. Disegnare anche il polinomio interpolatore a tratti e la retta ai minimi quadrati sugli scarti verticali.

Nota

Particolare attenzione va data al condizionamento dei vari sistemi lineari che vengono risolti.

J.5 Parte facoltativa

Implementare la seguente strategia. Ogni vertice che entra in una nuova figura viene inserito, se non è già presente, nella tabella T. Le spezzate sono individuate dagli indici dei loro vertici, numerati secondo la tabella T.

Modificare le routines grafiche usando questo diverso approccio.

Appendice K

Analisi di carico

K.1 Il problema

La sezione trasversale di una imbarcazione è riprodotta in figura K.1.

La forza che il vento esercita sulla vela, espressa in kg per m di altezza dell'albero, varia con l'elevazione z come riportata nella figura. L'espressione analitica di tale pressione lineare è:

$$p(z) = 300\frac{3z}{5+3z}\exp(-2z/10) \qquad kg/m$$

Assumendo che la cima dell'albero si sposti di un valore trascurabile sotto l'azione del vento, si calcoli la forza F risultante della pressione del vento e la distanza verticale da O della corrispondente retta di applicazione.

La forza F è data dall'integrale:

$$F = \int_0^{10} 300\frac{3z}{5+3z}\exp(-2z/10)dz$$

mentre d si ottiene dal rapporto:

$$d = \frac{\int_0^{10} z\, p(z)dz}{\int_0^{10} p(z)dz} = \frac{\int_0^{10} z300\frac{3z}{5+3z}\exp(-2z/10)dz}{F}$$

K.2 Compiti

Scrivere un programma che calcoli F e d utilizzando sia la formula dei trapezi che quella di Cavalieri-Simpson. Dividere l'intervallo di integrazione un numero crescente n di sottointervalli a partire da $n = 2$ per finire con $n = 128$.

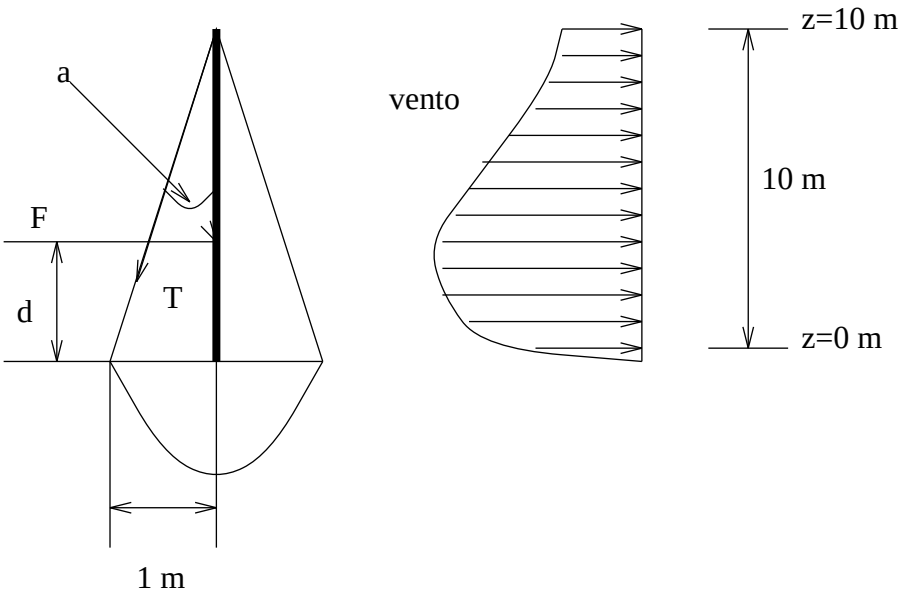


Figura K.1: Sezione trasversale di una imbarcazione.

Tabulare, al crescere di n, i valori ottenuti per F e per d coi due algoritmi. Commentare i risultati.

Appendice L

Moto di un giocattolo

L.1 Formulazione del problema

Un' auto giocattolo si muove su un piano illimitato a velocitá costante. Ad un assegnato istante l' asse anteriore inizia a girare a destra ad un tasso costante. L' asse posteriore dell' auto é fisso, quello anteriore puó girare indefinitamente.

Determinare il cammino percorso dall' automobilina.

L.2 Procedimento risolutivo.

Formalizzando la soluzione come in [AM88], sia (x, y) il centro dell' asse anteriore dell' auto e s la lunghezza del cammino percorso dal centro stesso (vedi figura L.1). La distanza l tra gli assi sia 1. Sia ξ l' angolo formato dall' auto con il centro degli assi di riferimento e sia θ l' angolo che l' asse anteriore forma con l' asse principale dell' auto.

Supponiamo che il guidatore inizi a girare il volante quando $x_0 = 0$, $y_0 = 0$ e che sia $\xi_0 = \theta_0 = 0$.

Allora il cammino dell' automobilina, ricordando che sia ds/dt che $d\theta/dt$ sono costanti, é governato dalle equazioni (vedi [AM88]):

$$\begin{cases} dx/ds &= \cos(\xi + \theta) \\ dy/ds &= \sin(\xi + \theta) \\ d\xi/ds &= \sin(\theta) \\ d\theta/ds &= k \end{cases} \tag{L.1}$$

k rappresenta la velocitá con cui il guidatore gira il volante. Il sistema di equazioni (L.1) assieme alla condizioni iniziali permette di calcolare la traiettoria dell' automobilina al variare di k.

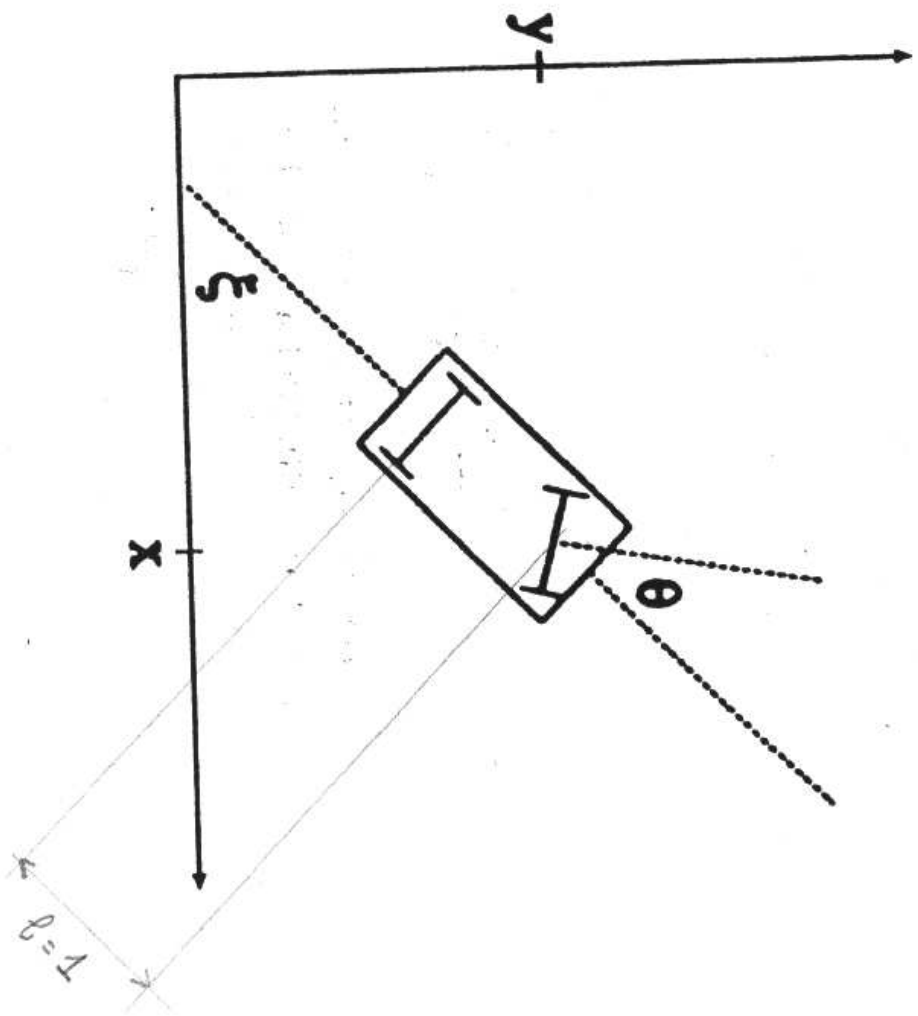


Figura L.1: Rappresentazione del problema.

L.3 Obiettivi

In figura L.2 sono riprodotte traiettorie per alcuni valori di k. L' obiettivo é quello di ricalcolare queste traiettorie integrando le equazioni (L.1).

L.4 Suggerimenti

Ecco alcuni modi per provare ad integrare numericamente le equazioni.

- Approssimare le 4 equazioni con lo schema alle differenze

$$du/ds = \Delta u/h + O(h). \tag{L.2}$$

- Generalizzare a sistemi di 4 equazioni gli schemi di Runge-Kutta proposti in [Gam94].

- Si puó provare ad utilizzare anche un pacchetto di manipolazione simbolica (es: Mathematica, Macsyma) o numerica (MATLAB), oppure una libreria numerica (IMSL) per integrare le equazioni.

- Risolvere analiticamente le ultime due equazioni. Sostituire i risultati nelle prime due. Il sistema si riduce a:

$$\begin{cases} dx/ds &= \cos(\alpha(s)) \\ dy/ds &= \sin(\alpha(s)) \end{cases}, \quad \alpha(s) = \frac{1}{k}(1 - \cos(ks)) + ks. \tag{L.3}$$

  Esso puø essere integrato ad esempio con lo schema dei trapezi. Confrontare le soluzioni con quelle trovate con gli altri schemi.

L.5 Esempio di risoluzione

Applicando lo schema alle differenze di Eulero (L.2) al sistema otteniamo le equazioni:

$$\begin{cases} x_{i+1} &= x_i + h\cos(\xi_i + \theta_i) \\ y_{i+1} &= y_i + h\sin(\xi_i + \theta_i) \\ \xi_{i+1} &= \xi_i + h\sin(\theta_i) \\ \theta_{i+1} &= \theta_i + hk \end{cases} \tag{L.4}$$

Usando il pacchetto di elaborazione numerica MATLAB (vedi [The06]), si puó facilmente calcolare una soluzione, usando un programma come il seguente:

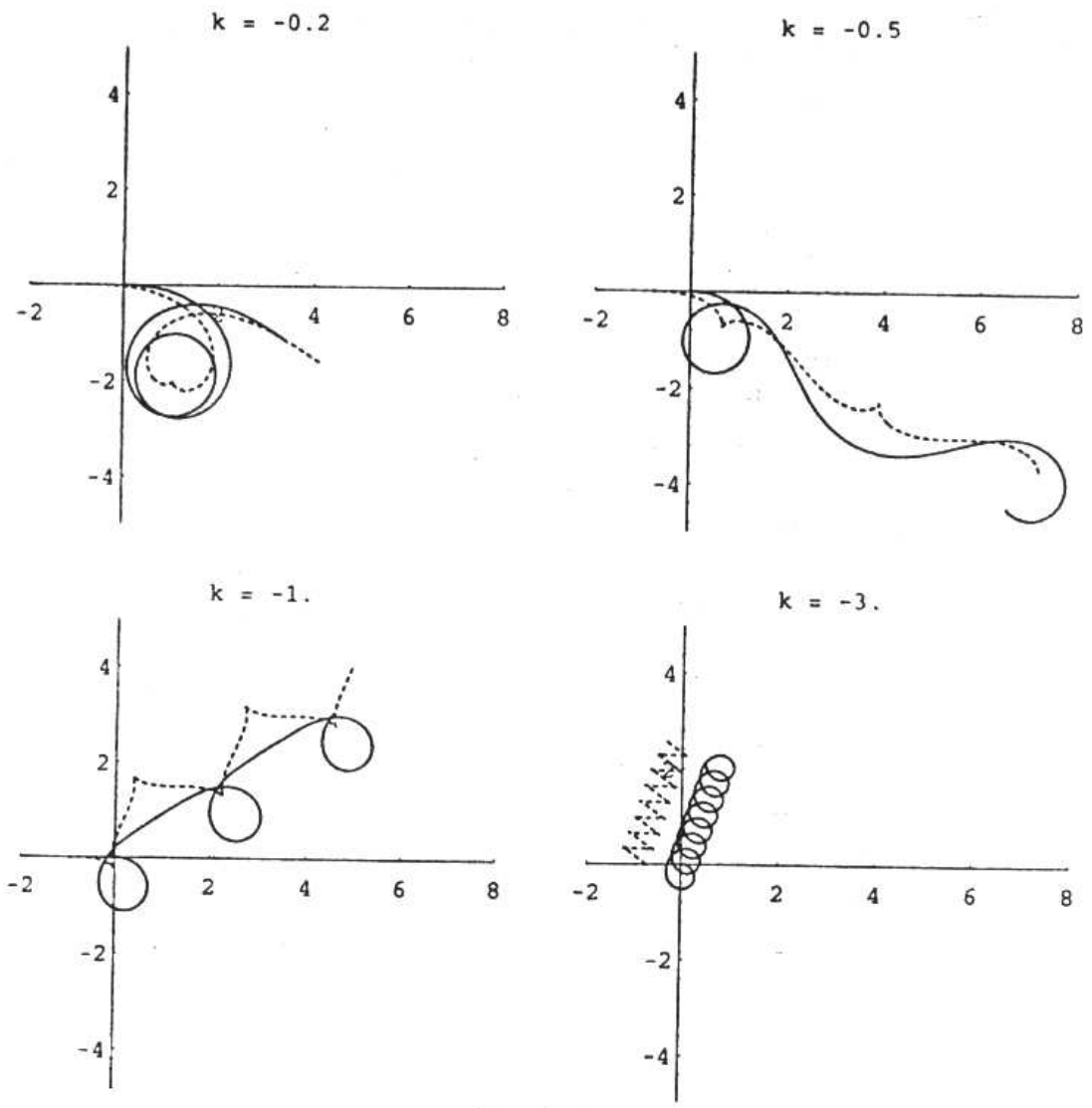


Figura L.2: Traiettorie ottenute scegliendo alcuni valori di k. Le linee continue rappresentano le traiettorie dell' asse anteriore, quelle tratteggiate le traiettorie dell' asse posteriore.

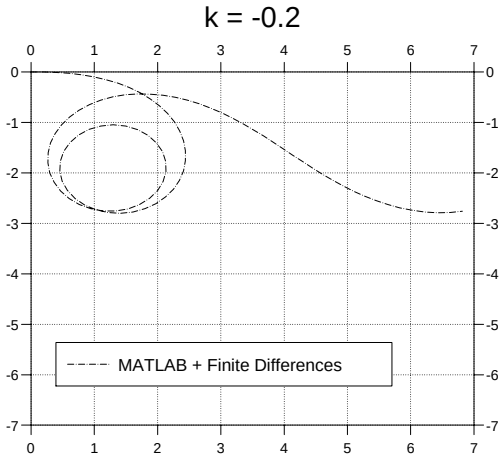


Figura L.3: Traiettoria ottenuta con lo schema di Eulero.

```
% risoluzione numerica del problema moto di un'auto, da
% SIAM Review 37(2) 1995

k = -0.2
sfin=20
nmax=100
h = sfin/nmax
x = 0*(1:(nmax+1));
y = x;
xi = x;
th = x;
i=1
s0 = 0
s(i) = 0;
x(i) = 0;
xi(i) = 0;
th(i) = 0;
for i=1:nmax
 s(i+1) = s0 + i*h;
 x(i+1) = x(i) + h*cos(xi(i) + th(i));
 y(i+1) = y(i) + h*sin(xi(i) + th(i));
```

```
 xi(i+1) = xi(i) + h*sin(th(i));
 th(i+1) = th(i) + h*k;
end
plot(x,y)
[x' y']
%
```

In figura L.3 si vede la traiettoria calcolata per $k = -0.2$, ponendo $h = 0.2$. Confrontandola con la corrispondente figura L.2 si vede che l' accordo è soddisfacente.

Appendice M

Onde di traffico

M.1 Introduzione

Vogliamo modellizzare lo scorrimento delle auto in una corsia di un' autostrada[1]. Per semplificare il problema. supponiamo che le auto non possano superarsi e che non vi siano caselli nel tratto che si studia[2]

M.2 Descrizione e compiti

Seguiamo la linea di ragionamento proposta in [Hab98].

Supponiamo di *concentrare* ogni auto nel suo baricentro, riducendola cosí ad un punto. Sia $u(x,t)$ il *campo di velocità*, ossia il valore $u(\bar{x},\bar{t})$ è la velocità associata al punto $\bar{x}$, all' istante $\bar{t}$. Se nel punto, in quell' istante non si trova alcuna auto, $u(\bar{x},\bar{t}) = 0$.

Sia $\rho(x,t)$ la *densità* del traffico, una funzione *liscia*[3] in x e t.

L' ipotesi che si possano usare funzioni continue per studiare il problema, va sotto il nome di *ipotesi del continuo*.

Indichiamo con q il *flusso di traffico*. Il valore $q(\bar{x},\bar{t})$ è la quantità di auto che passa *nell' unità di tempo* per il punto $\bar{x}$, all' istante $\bar{t}$. Si puø vedere che vale la relazione

$$q(x,t) = \rho(x,t)u(x,t). \tag{M.1}$$

Il principio di *conservazione*, afferma che la densità ρ e il campo di velocità, $u(\rho)$, pensato come funzione della densità, sono legati dalla relazio-

[1] Con il contributo di A. Ceccato.

[2] Queste assunzioni riducono la complessità del problema, ma ci allontanano anche molto dalla situazione reale!

[3] Per capire come si possa introdurre una funzione continua $\rho(x,t)$, a partire da una distribuzione discreta di auto puntiformi, si veda ancora [Hab98], par. 58.

ne [Hab98]

$$\frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial x} = \tag{M.2}$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho \cdot u(\rho))}{\partial x} = 0. \tag{M.3}$$

Usando la regola della catena, dalla (M.2), otteniamo la relazione

$$\frac{\partial \rho}{\partial t} + \frac{dq(\rho)}{d\rho}\frac{\partial \rho}{\partial x} = 0. \tag{M.4}$$

Ancora usando la regola della catena, si puø scrivere

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial t} + \frac{dx}{dt}\frac{\partial \rho}{\partial x}. \tag{M.5}$$

Confrontando le equazioni (M.4) e (M.5), possiamo dire che la densità è costante nel tempo, ossia

$$\frac{d\rho}{dt} = 0, \tag{M.6}$$

se

$$\frac{dx}{dt} = \frac{dq(\rho)}{d\rho} \equiv q'(\rho). \tag{M.7}$$

Questa relazione si puø interpretare dicendo che *se un osservatore, che ha la posizione x, si muove con velocità dx/dt data dalla relazione (M.7), allora egli osserva una densità di traffico costante.*

La densità tende a propagarsi in forma di onde, si parla di *onde di densità*. La relazione (M.7) ci dice che le onde di densità si propagano alla velocità $q'(\rho)$, che viene chiamata *velocità di densità locale*. Dato un punto x_0, nota la densità all' istante 0, $\rho(x_0, 0) = \rho_0$, partendo dall' istante $t = 0$ e integrando la funzione $q'(\rho)$, si ottiene una traiettoria

$$x_0^*(t) = x_0 + \int_0^t q'(\rho)dt, \tag{M.8}$$

lungo la quale la densità è costante. Lungo questa traiettoria, inoltre, la nostra equazione alle derivate parziali (M.4) si riduce all' equazione alle derivate *ordinarie* (M.6). Le curve $x_0^*(t)$ vengono chiamate *curve caratteristiche* della (M.4).

Fissato x_0 e noto ρ_0, dato che ρ è costante lungo le linee caratteristiche, la densità all' istante $t > 0$, nel punto $x_0^*(t)$, è $\rho(x_0^*, t) = \rho_0$.

Supponiamo di conoscere la densità iniziale, $\rho_0(x) = \rho(x, 0)$, all'istante $t = 0$, in ogni punto x. Possiamo calcolare una soluzione approssimata del problema differenziale

$$\begin{cases} \frac{\partial \rho}{\partial t} + \frac{dq(\rho)}{d\rho}\frac{\partial \rho}{\partial x} = 0, \quad x \in [a, b],\, t \in [0, T] \\ \rho(x, 0) = \rho_0(x), \end{cases} \tag{M.9}$$

usando il seguente metodo.

Sia u_{max} il limite di velocità sulla strada, che supponiamo nessuno superi! Sia ρ_{max} la densità massima raggiungibile, quella alla quale le auto si trovano *parafango contro parafango* (in realtà la densità massima corrisponde ad auto separate tra loro da un piccolo spazio).

Per semplificare la trattazione, supponiamo che il campo di velocità dipenda dalla densità in modo lineare, esattamente

$$u(\rho) = u_{max}\left(1 - \frac{\rho}{\rho_{max}}\right). \tag{M.10}$$

Essendo $q = \rho u$,

$$\frac{dq}{d\rho} = \frac{d\left(\rho\, u_{max}(1 - \rho/\rho_{max})\right)}{d\rho} = \tag{M.11}$$

$$u_{max}\left(1 - \frac{2\rho}{\rho_{max}}\right). \tag{M.12}$$

Poichè $(x_i^*(t), t)$ sono i punti di una curva caratteristica, $\rho(x, t)$ è costante su questa curva, quindi la relazione (M.8) diventa

$$x_i^*(t) = x_i(0) + u_{max}\left(1 - \frac{2\rho(x_i(0))}{\rho_{max}}\right)t. \tag{M.13}$$

Le curve caratteristiche sono dunque delle rette.

Ripetendo questo ragionamento per ogni x_i, otteniamo una soluzione del problema (M.9), sui punti di N caratteristiche. Approssimando questi valori con una funzione $\tilde{\rho}(x, t)$ abbiamo una soluzione ρ del problema (M.9). Ad esempio possiamo interpolare linearmente in t.

Questo schema per risolvere equazioni differenziali iperboliche [TS64] è detto *metodo delle caratteristiche*.

Consideriamo la situazione in cui nell' intervallo $[-1, 1]$ vi è una barriera, che rallenta il traffico e lo blocca, perciø non vi sono auto nella regione $x > 1$, mentre nella regione $x < -1$ le auto sono a densità massima, parafango–contro–parafango. Cosa accade se all' istante $t = 0$ la barriera viene rimossa?

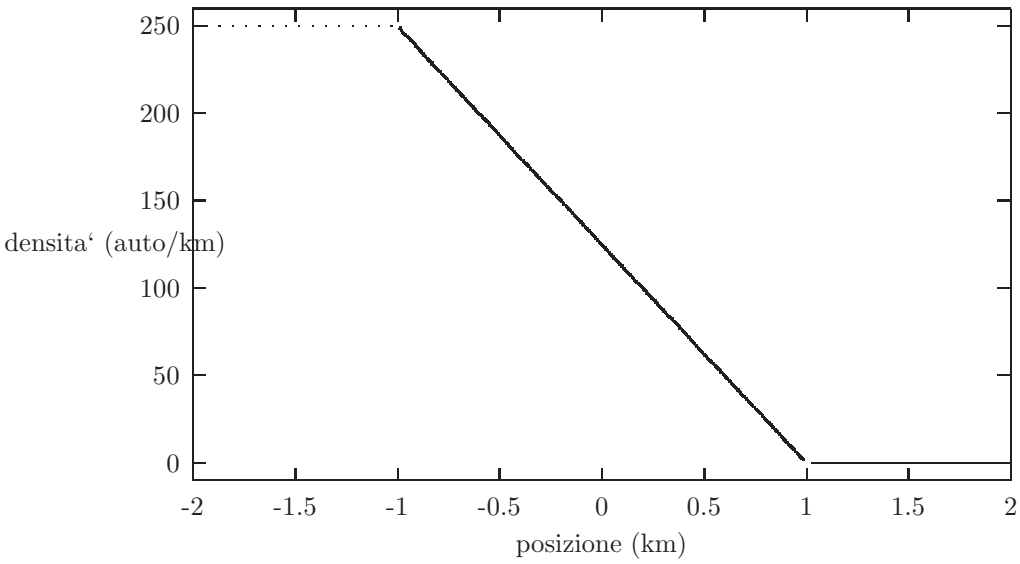


Figura M.1: Caso in cui viene rimossa una barriera.

Modellizziamo questa situazione con il problema (M.9), in cui poniamo

$$\rho(x,0) = \begin{cases} \rho_{max}, & \text{se } x < -1, \\ (1-x)\rho_{max}/2, & \text{se } -1 \le x \le 1, \\ 0, & \text{altrimenti.} \end{cases} . \tag{M.14}$$

Qui e nel seguito, poniamo $\rho_{max} = 250$ auto/Km, $u_{max} = 100$ Km/h; misuriamo le posizioni in Kilometri.

Esercizio M.2.1 Risolvere e disegnare i grafici delle caratteristiche a partire dai punti $x_i = -2 + i\Delta x$, $\Delta x = 4/M$, $i = 0, \ldots, M = 40$, nell'intervallo $0 \le t \le T = 0.027$ (0.027 ore $\simeq$ 100 sec.). Ricavare e disegnare la densità ρ del problema, nel rettangolo $R = [-2, 2] \times [0, T]$.

Esaminiamo il caso in cui una regione di bassa densità è seguita da una di alta densità. In questo caso, si forma un' *onda di compressione*, che si propaga all'indietro sull' asse delle x. Modellizziamo questa situazione con il problema (M.9), in cui poniamo

$$\rho(x,0) = |(1 + \pi/2 + \arctan(20 * x))\rho_{max}/(1+\pi)|. \tag{M.15}$$

In questa situazione, vi sono delle zone in cui le caratteristiche si intersecano. In queste zone il metodo delle caratteristiche non funziona. Esso predice valori diversi di $\rho(x,t)$, a seconda della caratteristica per (x,t) che si considera. In questi punti bisogna sviluppare un'analisi pi$\breve{}$ sofisticata.

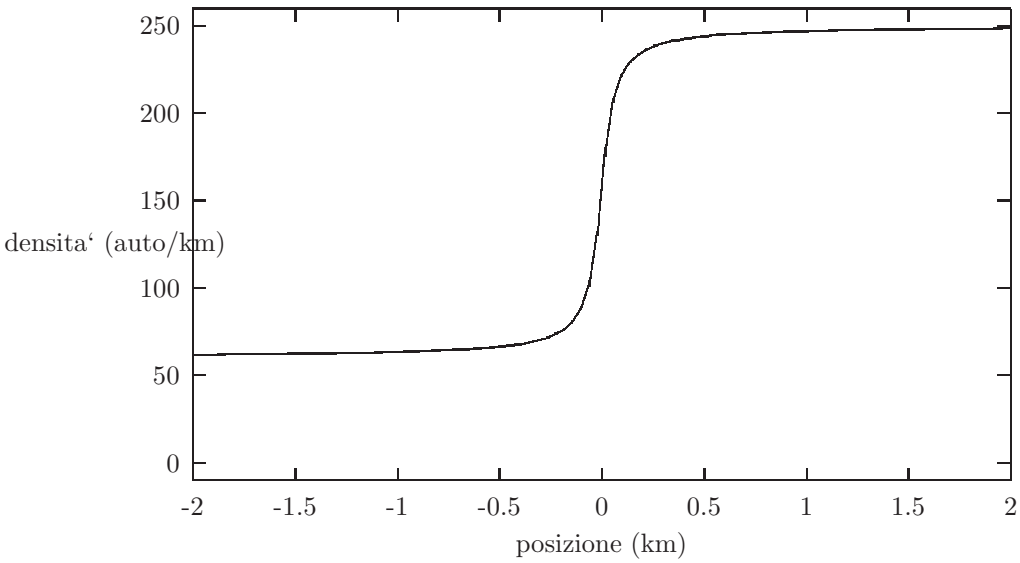


Figura M.2: Caso di congestione davanti alla colonna.

Esercizio M.2.2 Risolvere e disegnare i grafici delle caratteristiche negli stessi punti dell'esercizio precedente. Ricavare e disegnare la densità ρ del problema negli stessi punti dell'esercizio precedente. Non tracciare le caratteristiche oltre i punti in cui si intersecano. Non calcolare i valori di ρ nelle zone in cui le caratteristiche si intersecano.

Quando lungo una strada il traffico diventa più denso, le caratteristiche si intersecano e la teoria predice contemporaneamente due diversi valori per la densità. Il metodo non è applicabile.

Abbiamo assunto che il campo di velocità e la densità del traffico fossero funzioni continue (ipotesi del continuo). Rimovendo questa assunzione, possiamo trattare il caso in cui le caratteristiche si intersecano.

Ipotizziamo quindi che la densità di traffico e il campo di velocità siano discontinui in un certo punto, x_s, dello spazio e che tale discontinuità si possa propagare nel tempo secondo una data legge $x_s(t)$. La discontinuità viene chiamata *shock* e il punto x_s viene detto *punto di shock*.

Consideriamo una regione $x_1 < x < x_2$, tale che $x_1 < x_s(t) < x_2$. Il numero di auto contenute è dato da:

$$N(t) = \int_{x_1}^{x_2} \rho(x,t)\, dx \tag{M.16}$$

Questo integrale continua ad essere ben definito nonostante la discontinuità. La variazione della quantità di auto, $N(t)$, presente nella regione,

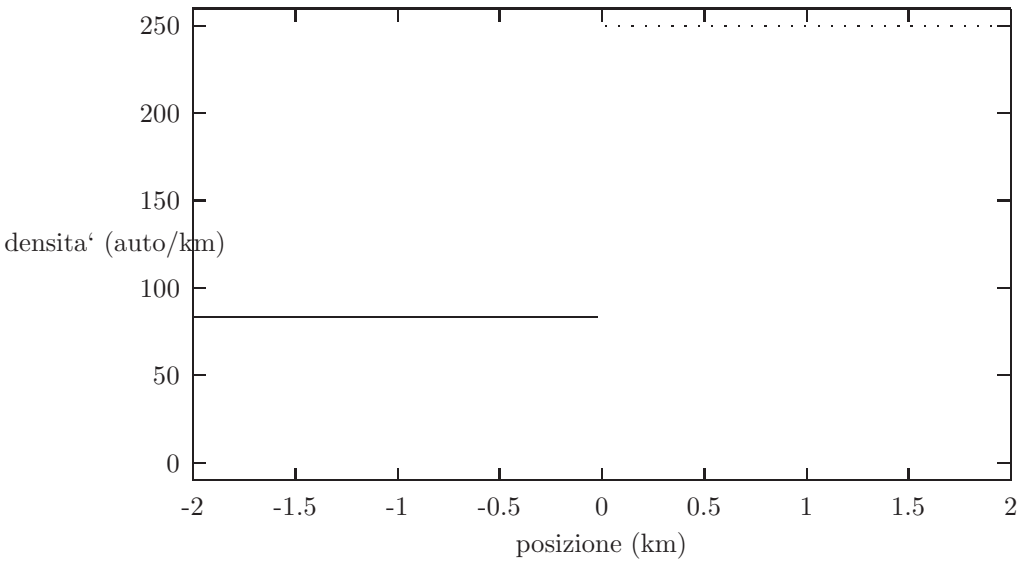


Figura M.3: Densità discontinua.

dalla legge di conservazione delle auto è data da:

$$\frac{dN(t)}{dt} = \frac{d}{dt}\int_{x_1(t)}^{x_2(t)} \rho(x,t)\,dx = q(x_1,t) - q(x_2,t). \tag{M.17}$$

Dividiamo l'intervallo in due parti

$$\int_{x_1}^{x_2} \rho\,dx = \int_{x_1}^{x_s} \rho\,dx + \int_{x_s}^{x_2} \rho\,dx. \tag{M.18}$$

Dato che x_s dipende dal tempo, anche x_1 e x_2 non sono costanti. Le legge di derivazione di integrali con estremi dipendenti dal tempo è:

$$\frac{d}{dt}\int_{\alpha(t)}^{\beta(t)} f(x,t)\,dx = \frac{d\beta}{dt}f(\beta,t) - \frac{d\alpha}{dt}f(\alpha,t) + \int_{\alpha(t)}^{\beta(t)} \frac{\partial f(x,t)}{\partial t}\,dx. \tag{M.19}$$

Questa formula non vale se vi è una discontinuità nell'intervallo di integrazione, ecco perchè lo dividiamo in due parti. Applicando la formula ai due integrali otteniamo:

$$\frac{d}{dt}\int_{x_1}^{x_s} \rho\,dx = \frac{dx_s^-}{dt}\rho(x_s^-,t) - \frac{dx_1}{dt}\rho(x_1,t) + \int_{x_1}^{x_s} \frac{\partial\rho(x,t)}{\partial t}\,dx; \tag{M.20}$$

$$\frac{d}{dt}\int_{x_s}^{x_2} \rho\,dx = \frac{dx_2}{dt}\rho(x_2,t) - \frac{dx_s^+}{dt}\rho(x_s^+,t) + \int_{x_s}^{x_2} \frac{\partial\rho(x,t)}{\partial t}\,dx; \tag{M.21}$$

Introduciamo l'operatore di salto (*jump operator*)

$$[z(x_s)] = [z] = z(x_s^-) - z(x_s^+). \tag{M.22}$$

La (M.19) diventa allora

$$\begin{aligned} \frac{d}{dt}\int_{x_1}^{x_2} \rho dt &= [\rho(x_s)]\frac{dx_s}{dt} + \\ \frac{dx_2}{dt}\rho(x_2,t) &- \frac{dx_1}{dt}\rho(x_1,t) + \\ \int_{x_1}^{x_s} \frac{\partial \rho}{\partial t}\, dx &+ \int_{x_s}^{x_2} \frac{\partial \rho}{\partial t}\, dx = q(x_1,t) - q(x_2,t). \end{aligned}$$

Se x_1 e x_2 sono vicini allo shock, il contributo dei due integrali è trascurabile, inoltre $(dx_2/dt)\rho(x_2,t) \simeq (dx_1/dt)\rho(x_1,t)$, quindi $(dx_2/dt)\rho(x_2,t) - (dx_1/dt)\rho(x_1,t) \simeq 0$. In definitiva si ha:

$$[\rho]\frac{dx_s}{dt} = [q]. \tag{M.23}$$

Nei punti di discontinuità questa condizione di shock sostituisce l'equazione differenziale che è valida negli altri casi.

Per esempio, consideriamo un flusso di traffico che viaggia con densità ρ_0 e che viene improvvisamente bloccato da un semaforo rosso situato nel punto $x = 0$. Ipotizziamo che le auto siano in grado di fermarsi istantaneamente. In $x = 0$ il traffico è bloccato ($u = 0$) e la densità è massima in ogni istante di tempo successivo a quello iniziale (vedi figura M.3). Il modello cosí proposto presenta caratteristiche che si intersecano. Introduciamo uno shock: le densità prima e dopo lo shock sono rispettivamente la densità iniziale ρ_0 e la densità massima ρ_{max}. Nel punto di shock deve essere soddisfatta la condizione:

$$\frac{dx_s}{dt} = \frac{[q]}{[\rho]} \tag{M.24}$$

La posizione iniziale dello shock è nota e costituisce una condizione del problema: $x_s(0) = 0$.

Si ha quindi:

$$\frac{[q]}{[\rho]} = \frac{u(\rho_{max})\rho_{max} - u(\rho_0)\rho_0}{\rho_{max} - \rho_0} \tag{M.25}$$

Ricordando che $u(\rho_{max}) = 0$, l'eq. (M.24) diventa

$$\frac{dx_s}{dt} = \frac{-u(\rho_0)\rho_0}{\rho_{max} - \rho_0} < 0 \tag{M.26}$$

La velocità dello shock è dunque costante e negativa e risolvendo l'equazione differenziale otteniamo

$$x_s = \frac{-u(\rho_0)\rho_0}{\rho_{max} - \rho_0} \cdot t.$$

Esercizio M.2.3 Considerare la densità:

$$\rho(x,0) = \begin{cases} \frac{\rho_{max}}{3}, & x < 0 \\ \rho_{max}, & x > 0. \end{cases} \tag{M.27}$$

Risolvere e disegnare i grafici delle caratteristiche, ricavare e disegnare la densità ρ negli stessi punti dell'esercizio precedente.

Appendice N

Analisi della dinamica

N.1 Introduzione

E' dato il sistema di equazioni differenziali:

$$\begin{cases} y_1' &= -[(-\pi\, y_1\, y_4 + \pi\, y_2\, y_3 - y_1)\; K + 2\, y_1\, D]\,/2 \\ y_2' &= -[(\pi\, y_2\, y_4 + \pi\, y_1\, y_3 - y_2)\; K + 2\, y_2\, D]\,/2 \\ y_3' &= 2\,\pi\, y_1\, y_2\, K - 4\, y_3\, D \\ y_4' &= (\pi\, y_2^2 - \pi\, y_1^2)\; K - 4\, y_4\, D \end{cases} \tag{N.1}$$

N.2 Compiti

(a) Analizzarne la dinamica rispetto al parametro K, posto $D = 1$.

(b) Date le condizioni iniziali

$$y_1(0) = \epsilon, \quad y_2(0) = \epsilon, \quad y_3(0) = 0, \quad y_4(0) = 0 \tag{N.2}$$

e posto

$$T = 20, \quad D = 1, \quad K = 4, \quad \epsilon = 10^{-6},$$

calcolare le soluzioni del sistema che si ottengono usando almeno due diversi metodi alle differenze finite, fra i quali:

- Runge-Kutta del secondo ordine;
- rappresentazione con differenze in avanti delle derivate e "congelamento" dei secondi membri agli istanti in cui le quantità sono note.

Tabulare e graficare le soluzioni numeriche ottenute.

Appendice O

Sistemi Hamiltoniani

O.1 Introduzione

La risoluzione di problemi riguardanti la dinamica [1] di sistemi meccanici iniziò con Galilei(1638) e con i *Principia di Newton*(1687). Newton ridusse la descrizione del movimento di un punto-massa ad un sistema di equazioni differenziali.

Lo studio del movimento di sistemi complessi (corpi rigidi o collegati tramite corde o aste) comportava serie difficoltà, fino a quando Lagrange trovò un modo elegante di trattarlo [Gan75]. Supponiamo di avere un sistema meccanico a d gradi di libertà, detto *sistema Lagrangiano*, la cui posizione è descritta dalle quantità $q = (q_1, ..., q_d)$, chiamate *coordinate generalizzate*.

Lagrange usa le quantità

$$T = T(q, \dot{q}), \quad U = U(q), \tag{O.1}$$

che sono rispettivamente l'*energia cinetica* e l' *energia potenziale* del sistema.

La funzione

$$L = T - U \tag{O.2}$$

viene chiamata il *Lagragiano* del sistema. Le coordinate $q_1(t), ..., q_d(t)$ obbediscono alle equazioni differenziali:

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) = \frac{\partial L}{\partial q}, \tag{O.3}$$

che vengono dette le *equazioni Lagragiane* del sistema. La risoluzione di queste equazioni permette di descrivere il movimento di un tale sistema, a partire da condizioni iniziali date.

[1] Sezione sviluppata con la collaborazione di E. Mion.

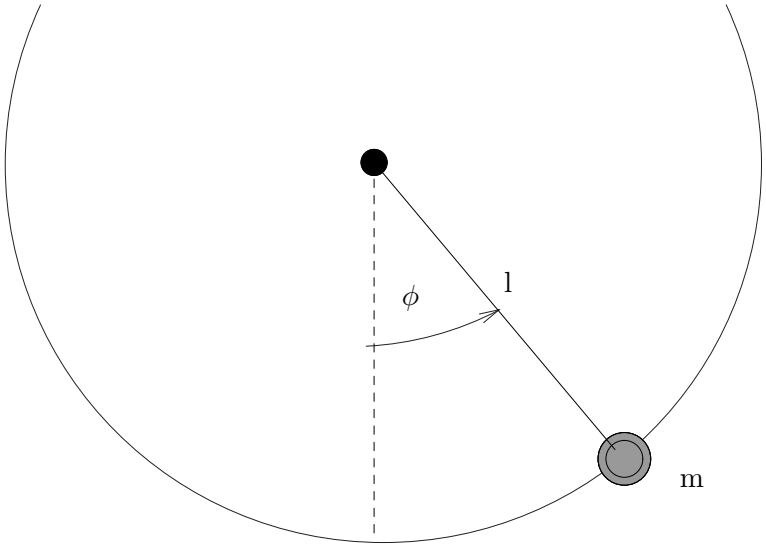


Figura O.1: Pendolo dell'esempio 1.

Esempio O.1.1 Supponiamo di avere un pendolo (Fig.O.1) di massa m, collegato ad un'asta rigida senza massa di lunghezza l. E' un sistema ad un solo grado di libertà. La sua energia cinetica è

$$T = \frac{1}{2} m\, l^2\, \dot{\phi}^2 \tag{O.4}$$

mentre l'energia potenziale è:

$$U = mgz = m\, g\, l \cdot \cos\phi. \tag{O.5}$$

Il Langragiano del problema è:

$$L = T - U = \frac{1}{2} m\, l^2\, \dot{\phi}^2 - m\, g\, l \cdot \cos\phi. \tag{O.6}$$

Hamilton nel 1934 semplificò la struttura delle equazioni di Lagrange e trasformò il problema in una forma che possiede una interessante *simmetria*. Egli

- introdusse le variabili di Poisson, ricavando i *momenti coniugati o generalizzati*

$$p_k = \frac{\partial L}{\partial \dot{q}_k}(q, \dot{q}), \quad k = 1, \ldots, d. \tag{O.7}$$

- considerò una nuova funzione, chiamata *Hamiltoniana*

$$H := p^T \dot{q} - L(q, \dot{q}). \tag{O.8}$$

E' possibile dimostrare che le equazioni di Lagrange sono equivalenti alle *equazioni di Hamilton*

$$\dot{p}_k = -\frac{\partial H}{\partial q_k}(p, q), \quad \dot{q}_k = \frac{\partial H}{\partial p_k}(p, q), \quad k = 1, \ldots, d. \tag{O.9}$$

Esempio O.1.2 Riprendendo l'esempio 1 possiamo scrivere:

$$p_\phi = \frac{\partial L}{\partial \dot{\phi}} = ml^2\dot{\phi} \tag{O.10}$$

e risolvendo rispetto a $\dot{\phi}$ otteniamo

$$\dot{\phi} = \frac{p_\phi}{ml^2}. \tag{O.11}$$

L' hamiltoniana del problema è:

$$H(p_\phi, \phi) = \frac{p_\phi^2}{2ml^2} - mgl \cdot cos\phi \tag{O.12}$$

Ponendo $p = p_\phi$ e $q = \phi$ otteniamo

$$H(p, q) = \frac{p^2}{2ml^2} - mgl \cdot \cos q. \tag{O.13}$$

Possiamo dunque scrivere le equazioni del moto

$$\dot{p} = -\frac{\partial H}{\partial q}(p, q) = -m\, g\, l\, \sin q \tag{O.14}$$

$$\dot{q} = \frac{\partial H}{\partial p}(p, q) = \frac{p}{m\, l^2}. \tag{O.15}$$

O.1.1 Compiti

Supponendo che la la massa m del pendolo sia 1 Kg, che la lunghezza dell'asta sia $l = 1$ m e che l' accelerazione di gravità agente sul punto sia $g = 1$ m/s^2, determinare il moto del pendolo utilizzando i metodi di Eulero Esplicito e

Eulero Implicito [2]. Calcolare il moto per i valori $h = 0.2$ s e $(p_0, q_0) = (0, 0.5)$. Disegnare due grafici che mostrano l'evoluzione temporale del sistema nelle coordinate p e q, $0 \leq t \leq 100$ s. Il piano (p, q) viene chiamato *spazio delle fasi* (phase space). Disegnare anche i grafici che mostrano l'andamento di ϕ in funzione di t.

O.2 Sistemi Simplettici

I sistemi Hamiltoniani hanno una importantissima proprietà, chiamata *simpletticità del flusso*.

Per illustrarne il significato, consideriamo un parallelogramma che giace su un piano. Il parallelogramma è individuato da due vettori

$$\xi = (\xi^p, \xi^q)^T, \quad \eta = (\eta^p, \eta^q)^T, \tag{O.16}$$

ed è formato dai punti

$$P = \{t\xi + s\eta \quad t.c. \quad 0 \leq s, t \leq 1\}. \tag{O.17}$$

Nel caso di dimensione $d = 1$, definiamo l'*area orientata*, A_o del parallelogramma

$$A_o(P) = \det \begin{pmatrix} \xi^p & \eta^p \\ \xi^q & \eta^q \end{pmatrix} = \xi^p \eta^q - \xi^q \eta^p. \tag{O.18}$$

Quando $d > 1$, consideriamo *la somma delle aree orientate della proiezione di P nelle coordinate (p_i, q_i)*

$$\omega(\xi, \eta) := \sum_{i=1}^{d} \det \begin{pmatrix} \xi_i^p & \eta_i^p \\ \xi_i^q & \eta_i^q \end{pmatrix} = \sum_{i=1}^{d} (\xi_i^p \eta_i^q - \xi_i^q \eta_i^p). \tag{O.19}$$

Questa è una mappa bilineare che agisce sui vettori di $\mathbb{R}^{2d}$. In notazione matriciale la mappa ha la forma

$$\omega(\xi, \eta) = \xi^T J \eta, \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}, \tag{O.20}$$

dove I è la matrice identica.

[2] Ricordiamo che i due schemi sono

$$y_{n+1} = y_n + hf(y_n) \quad \text{(Eulero Esplicito)}$$

$$y_{n+1} = y_n + hf(y_{n+1}) \quad \text{(Eulero Implicito)}$$

Per applicare quest'ultimo metodo al problema del pendolo, bisogna risolvere ad ogni passo un' equazione non lineare. Usate lo schema di Newton–Raphson per farlo.

Definizione O.2.1 Una funzione lineare $A : R^{2d} \rightarrow R^{2d}$ è chiamata *simplettica* se per ogni $\xi, \eta \epsilon R^{2d}$:

$$A^T J A = J, \tag{O.21}$$

ossia $\omega(A\xi, A\eta) = \omega(\xi, \eta)$.

Nel caso $d = 1$, possiamo dire che l'area del parallelogramma *si preserva*. In generale, la simplettictà garantisce che la somma delle aree orientate delle proiezioni di P su (p_i, q_i) è *uguale* a quella del parallelogramma trasformato $A(P)$.

Dato un sistema del tipo:

$$\dot{u} = a(u, v) \tag{O.22}$$
$$\dot{v} = b(u, v), \tag{O.23}$$

possiamo calcolarne una soluzione numerica tramite lo schema

$$u_{n+1} = u_n + h\, a(u_{n+1}, v_n), \tag{O.24}$$
$$v_{n+1} = v_n + h\, b(u_{n+1}, v_n). \tag{O.25}$$

Questo schema viene chiamato *Metodo Simplettico di Eulero* [HLW02]. Si prova che questo schema preserva la simpletticità, mentre Eulero Esplicito ed Eulero Implicito non la preservano. Questo comporta che la dinamica *a lungo termine* del sistema modellizzato non viene correttamente approssimata usando gli ultimi due schemi.

O.2.1 Compiti

Risolvere l'esercizio del pendolo utilizzando il Metodo Simpettico di Eulero. Utilizzare i valori $h = 0.3$ s, $p_0 = 0$ e $q_0 = 0.7, 1.4, 2.1$. Disegnare un grafico che mostra l' evoluzione temporale del sistema nelle coordinate p e q, $0 \leq t \leq 100$ s. Disegnare anche i grafici che mostrano l'andamento di ϕ in funzione di t. Confrontare i nuovi grafici con quelli ottenuti precedentemente. Qual è più corretto?

Appendice P

Test dello Shock Tube

P.1 Descrizione del problema

Supponiamo di avere un tubo (vedi figura P.1) diviso in due parti uguali da una membrana[1]. Nella metà sinistra è contenuto un fluido l, caratterizzato da alta densità, d_l, e pressione, p_l. Nella metà destra, invece, vi è fluido r, di densità $d_r \ll d_l$, e pressione $p_r \ll p_l$.

Al tempo $t_0 = 0$, la membrana viene rimossa ed il fluido l inizia a diffondere entro il fluido r, originando uno shock che si propaga verso la parte destra del tubo, mentre un'onda, chiamata *onda di rarefazione*, si propaga in direzione opposta, alla velocità locale del suono.

Si vengono a creare cinque distinte regioni (vedi figura P.2):

- il fluido imperturbato l;
- l'onda di rarefazione;
- una regione di densità e pressione costante (interfaccia);
- lo shock;
- il fluido imperturbato r.

[1]Sezione sviluppata con la collaborazione di C. Gheller, R. Favaretto, A. Missio, M. Guidolin.

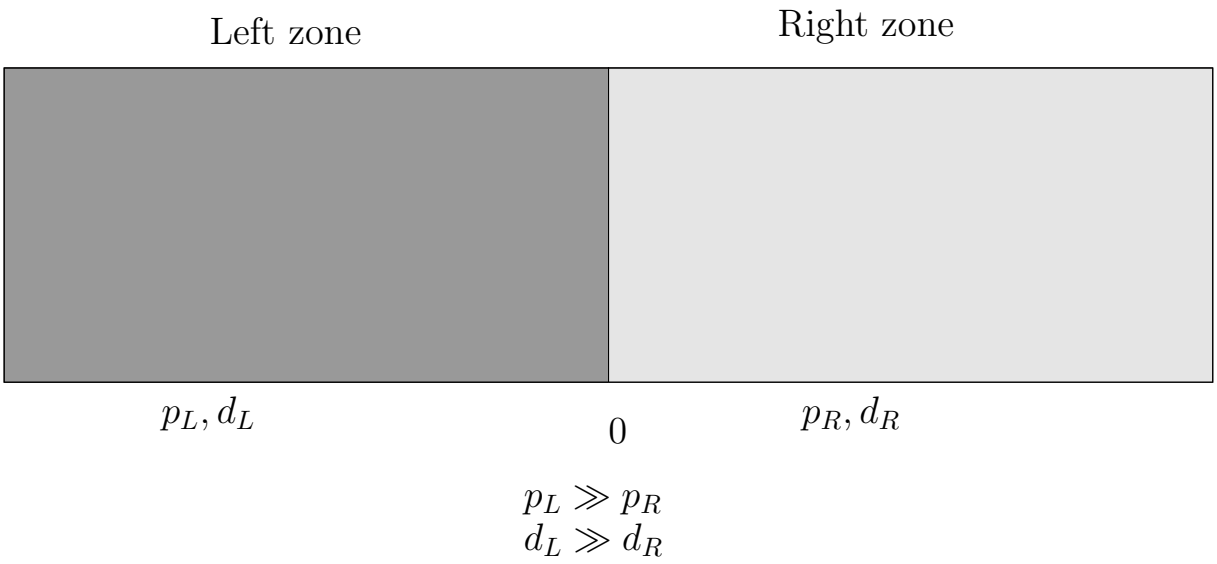


Figura P.1: Rappresentazione del tubo.

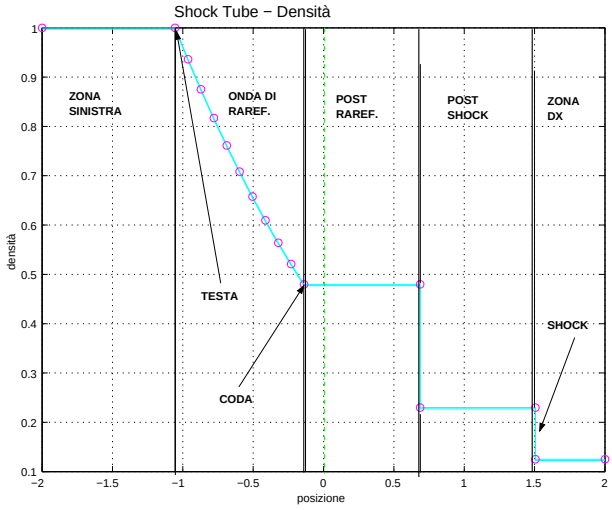


Figura P.2: Regioni che si vengono a creare dopo la rimozione della membrana.

P.2 Sistema fluidodinamico ideale

L'evoluzione di un sistema fluidodinamico ideale in una dimensione viene descritta dalle seguenti equazioni idrodinamiche [CF48, LL87]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial [\rho v]}{\partial x} = 0, \tag{P.1}$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial [\rho v^2 + p]}{\partial x} = 0, \tag{P.2}$$

$$\frac{\partial E}{\partial t} + \frac{\partial [(E+p)v]}{\partial x} = 0, \tag{P.3}$$

e dall'equazione di stato:

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2, \tag{P.4}$$

dove E indica l'energia, ρ la densità, p la pressione e γ è la costante adiabatica; $\gamma = 5/3$ per un fluido barionico non relativistico.

Ponendo $w_1 = \rho$, $w_2 = \rho\, v$, $w_3 = E$, e $f_1 = \rho\, v = w_2$, $f_2 = \rho v^2 + p$, $f_3 = (E+p)v$, le equazioni si possono riscrivere

$$\frac{\partial w_k}{\partial t} + \frac{\partial f_k}{\partial x} = 0, \quad k = 1, 2, 3. \tag{P.5}$$

Per determinare l' evoluzione del sistema, bisogna quindi risolvere le equazioni (P.5).

P.3 Metodi di integrazione numerica

Descriveremo due metodi numerici, lo *schema di Lax* e quello di *Lax-Wendroff*, che useremo per risolvere il test dello shock tube.

Entrambi gli schemi fanno uso di metodi di approssimazione alle differenze finite.

Sia x_i la coordinata dell'i-esimo punto della griglia.

Per determinare l'evoluzione temporale di w_k possiamo considerarne lo sviluppo in serie di Taylor rispetto al tempo di riferimento $\bar{t}$:

$$w_k(x_i, t) = w_k(x_i, \bar{t}) + \left[\frac{\partial w_k(x_i, t)}{\partial t}\right]_{t=\bar{t}} (t - \bar{t}) + \tag{P.6}$$

$$+ \frac{1}{2}\left[\frac{\partial^2 w_k(x_i, t)}{\partial t^2}\right]_{t=\bar{t}} (t - \bar{t})^2 + ... \tag{P.7}$$

Siano: t_n l'n-esimo istante temporale; $\bar{t} = t_n$; $\Delta t = t_{n+1} - t_n$.
Scrivendo la (P.7) al tempo t_{n+1} otteniamo:

$$w_k(x_i, t_{n+1}) = w_k(x_i, t_n) + \left[\frac{\partial w_k(x_i,t)}{\partial t}\right]_{t=t_n} (t - t_n) + \quad \text{(P.8)}$$

$$+ \frac{1}{2}\left[\frac{\partial^2 w_k(x_i,t)}{\partial t^2}\right]_{t=t_n} (t - t_n)^2 + ... \quad \text{(P.9)}$$

Usando la (P.5), otteniamo

$$w_k(x_i, t_{n+1}) = w_k(x_i, t_n) - \left[\frac{\partial f_k(x,t)}{\partial x}\right]_{x=x_i} \Delta t - \quad \text{(P.10)}$$

$$+ \frac{1}{2}\left[\frac{\partial}{\partial t}\left[\frac{\partial f_k(x,t)}{\partial x}\right]_{x-x_i}\right]_{t=t_n} \Delta t^2 + ... \quad \text{(P.11)}$$

Trascurando i termini di ordine superiore al primo e approssimando la derivata con le differenze centrali, otteniamo

$$w_k(x_i, t_{n+1}) = w_k(x_i, t_n) - \frac{f_k(x_{i+1}, t_n) - f_k(x_{i-1}, t_n)}{2\Delta x}\Delta t \quad \text{(P.12)}$$

P.4 Integrazione temporale

La scelta di un Δt opportuno si rivela estremamente delicata dato che, se è grande si ha instabilità numerica, se è troppo piccolo, il costo computazionale diventa alto e si verificano altri problemi numerici.

Consideriamo una griglia unidimensionale caratterizzata da N punti spaziali, definita sull'intervallo $[0, L]$, $x_i = i\Delta x$, $\Delta x = L/N$, $i = 0, \ldots, N$.

Se $|v|$ è la velocità con cui si propaga la perturbazione, per garantire stabilità numerica, il passo temporale deve soddisfare alla cosiddetta *condizione di Courant*

$$\Delta t \leq C\frac{\Delta x}{|v|}. \quad \text{(P.13)}$$

dove $0 < C \leq 1$ è una costante, chiamata *coefficiente di Courant*.

In questo modo durante il tempo Δt il fenomeno non si può propagare più di Δx nello spazio, quindi la sua variazione è abbastanza limitata. Poniamo

$$\alpha = \frac{\Delta x}{|v_m|}, \quad |v_m| = \max_i |v_i|. \quad \text{(P.14)}$$

Poiché possiamo avere velocità piccole nella simulazione, fissiamo un valore $\beta > 0$ che dipende dal problema in esame, e poniamo

$$\Delta t = \begin{cases} C\alpha, & \text{se } \alpha < \beta, \\ C\beta, & \alpha \geq \beta \end{cases} \quad \text{(P.15)}$$

P.4.1 Metodo di Lax

Il metodo di Lax è uno schema alle differenze finite del primo ordine. A partire dalla eq. (P.12) si pone

$$w_k(x_i, t_{n+1}) = \frac{w_k(x_{i+1}, t_n) + w_k(x_{i-1}, t_n)}{2} - \tag{P.16}$$

$$+ \frac{f_k(x_{i+1}, t_n) - f_k(x_{i-1}, t_n)}{2\Delta x}\Delta t \tag{P.17}$$

In letteratura si prova che pregi di questo metodo sono la limitata spesa computazionale e l'assenza di fenomeni come oscillazioni non fisiche o la perdita di positività di grandezze che fisicamente sono positive.

P.4.2 Il metodo di Lax-Wendroff

Questo è uno schema alle differenze finite del secondo ordine. L'eq. (P.7) si può riscrivere

$$w_k(t_{n+1}) = w_k(t_n) + \left[\frac{\partial w_k}{\partial t}\right]_{t=\bar{t}} \Delta t + \frac{1}{2}\left[\frac{\partial^2 w_k}{\partial t^2}\right]_{t=\bar{t}} \Delta t^2 \tag{P.18}$$

Usando l'equazione (P.5) possiamo scrivere:

$$\frac{\partial^2 w_k}{\partial t^2} = -\frac{\partial}{\partial t}\frac{\partial f_k}{\partial x} = -\frac{\partial}{\partial t}\left(\frac{\partial x}{\partial t}\frac{\partial f_k}{\partial x}\right). \tag{P.19}$$

Ma dall'equazione (P.5) si ha anche che:

$$\frac{\partial x}{\partial t} = -\frac{\partial f_k}{\partial w_k}. \tag{P.20}$$

Sostituendo nella (P.19) otteniamo

$$\frac{\partial^2 w_k}{\partial t^2} = \frac{\partial}{\partial x}\left(\frac{\partial f_k}{\partial w_k}\frac{\partial f_k}{\partial x}\right). \tag{P.21}$$

Possiamo, quindi, riscrivere la (P.18) come

$$w_k(t_{n+1}) = w_k(t_n) - \frac{\partial f_k}{\partial t}\Delta t + \frac{1}{2}\frac{\partial}{\partial x}\left(\frac{\partial f_k}{\partial w_k}\frac{\partial f_k}{\partial x}\right)\Delta t^2. \tag{P.22}$$

È stato dimostrato che tale approssimazione può essere sostituita da uno schema a due step, detto *schema di Lax-Wendroff-Richtmyer* [RM67]. Poniamo $w_{k,n}^{(i)} := w_k(x_i, t_n)$.

Quantità	Dimensioni (SI)	Parte sinistra	Parte destra
densità	M/L	1.0	0.125
velocità	L/T	0.0	0.0
pressione	M/T^2	2/3	2/30

Tabella P.1: Condizioni iniziali.

1. Si calcolano i valori delle grandezze ausiliarie $\tilde{w}_k^{(i)}$ al tempo $t_{n+1/2} = t_n + \Delta t_n/2$, $\Delta t_n = t_{n+1} - t_n$,

$$\tilde{w}_{k,n+1/2}^{(i+1)} = \frac{1}{2}\left[w_{k,n}^{(i+1)} + w_{k,n}^{(i)}\right] - \frac{\Delta t}{2\Delta x}\left[f_{k,n}^{(i+1)} - f_{k,n}^{(i)}\right]. \tag{P.23}$$

   A partire dai $\tilde{w}_{k,n+1/2}^{(i)}$, si calcolano i valori $\tilde{f}_{k,n+1/2}^{(i)}$. Ad esempio, $f_1 = \rho\, v$, quindi

$$\tilde{f}_{1,n+1/2}^{(i)} = \tilde{\rho}_{n+1/2}^{(i)}\, \tilde{v}_{n+1/2}^{(i)}, \quad i = 0, \ldots, N.$$

2. All' istante t_{n+1} abbiamo

$$w_{k,n+1}^{(i)} = w_{k,n}^{(i)} - \frac{\Delta t}{\Delta x}\left(\tilde{f}_{k,n+1/2}^{(i+1)} - \tilde{f}_{k,n+1/2}^{(i)}\right). \tag{P.24}$$

Pregio di questo metodo è la maggior accuratezza rispetto al metodo di Lax. In particolare, è molto contenuto il fenomeno della diffusione numerica, anche se, in prossimità delle zone in cui compaiono forti gradienti, compaiono delle oscillazioni non fisiche. La spesa rispetto allo schema di Lax è maggiore, sia come occupazione di memoria (si devono mantenere in memoria le grandezze al tempo $t_{n+1/2}$), sia come costo computazionale (l'algoritmo richiede due steps ad ogni passo temporale).

P.5 Compiti

Implementare gli schemi di Lax e Lax–Wendroff, confrontando i risultati con quelli analitici, forniti dal codice Matlab `shock.m`, reperibile nella cartella `$CN/Esercitazioni/Shocktube`.

Porre $\gamma = 5/3$, $\Delta t = 0.05$. Le condizioni iniziali sono riportate nella tabella P.1. L'energia interna iniziale a destra e a sinistra si puø calcolare usando l' equazione (P.4). Effettuare i confronti negli istanti temporali $t = 1, 2, 4, 8, 16$, ponendo alternativamente $C = 0.9, 0.8, 0.6$. Il passo temporale soddisfa la condizione di Courant?

Quanto vale α?

Bibliografia

[AM88] J. C. Alexander e J. H. Maddocks. On the maneuvering of vehicles. *SIAM J. Appl. Math.*, 48:38–51, 1988.

[Atk78] K. E. Atkinson. *An introduction to Numerical Analysis*. John Wiley & Sons, New York, 1978.

[Bal99] S. Balsamo. Sistemi di elaborazione dell' informazione: valutazione delle prestazioni di sistemi. Dispense per il corso di *Sistemi di Elaborazione dell' Informazione*, Università di Venezia, Corso di Laurea in Informatica. Disponibile all' URL `http://www.dsi.unive.it/~balsamo/dispense.html`, ultimo accesso: 17 sttembre 2007, 1999.

[BCM88] D. Bini, M. Capovani, e O. Menchi. *Metodi Numerici per l' Algebra Lineare*. Zanichelli, Bologna, 1988.

[BDJ99] M. W. Berry, Z. Drmač, e E. R. Jessup. Matrices, vector spaces and information retrieval. *SIAM Review*, 41(2):335–362, 1999.

[Ber94] M. Bertsch. *Istituzioni di Matematica*. Bollati Boringhieri, Torino, 1994.

[BR02] A. Brandt e D. Ron. Multigrid solvers and multilevel optimization strategies. Manuscript, 2002.

[Bra77] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31(138):333–390, April 1977.

[Bra89] A. Brandt. Rigorous local mode analysis of multigrid. In J. Mandel, S. F. McCormick, J. E. Dendy, C. Farhat, G. Lonsdale, S. V. Parter, J. W. Ruge, e K. Stüben, (a cura di), *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods*, pagina 438, Philadelphia, PA, USA, 1989. SIAM.

[Bra94] A. Brandt. Rigorous quantitative analysis of multigrid, I: Constant coefficients two-level cycle with L_2-norm. *SIAM J. Num. Anal.*, 31(6):1695–1730, December 1994.

[BRS02] S. C. Brenner e L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer Verlag, new York, ii edizione, 2002.

[Car91] G. Cariolaro. *La teoria unificata dei segnali*. UTET, Torino, 1991.

[CF48] R. Courant e K. O. Friedrichs. *Supersonic Flow and Shock Waves*. Interscience, New York, 1948.

[CLRS05] T. H. Cormen, C. E. Leiserson, R. L. Rivest, e C. Stein. *Introduzione agli algoritmi e strutture dati*. McGraw-Hill, 2005.

[Com91] V. Comincioli. *Analisi Numerica*. McGraw-Hill Italia, Milano, 1991.

[D$\ddot{9}$6] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM Journal of Numerical Analysis*, 33(3):1106–1124, 1996.

[Dem97] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[EEHJ96] K. Eriksson, D. Estep, P. Hansbo, e C. Johnson. *Computational Differential Equations*. Cambridge University Press, Cambridge MA, 1996.

[FvDFH90] J. D. Foley, A. van Dam, S. K. Feiner, e J. F. Hughes. *Computer Graphics. Principles and Practice*. Addison-Wesley, Reading, MA, ii edizione, 1990.

[Gam94] G. Gambolati. *Lezioni di Metodi Numerici per Ingegneria e Scienze Applicate*. Cortina, Padova, 1994.

[Gan75] F. R. Gantmacher. *Lectures in Analytical Mechanics*. MIR Publishers, Moscow, 1975.

[Gol91] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

[Gv89] G. Golub e C. F. van Loan. *Matrix Computation*. The Johns Hopkins University Press, Baltimore, ii edizione, 1989.

[Hab98] R. Haberman. *Mathematical Models*. SIAM, Philadelphia, PA, 1998. Unabridged republication of Prentice-Hall book, Englewood Cliffs, NJ, 1977.

[HLW02] E. Hairer, C. Lubich, e G. Wanner. *Geometric Numerical Integration*. Springer-Verlag, Berlin, 2002.

[HS93] W. Huang e D. M. Sloan. A new pseudospectral method with upwind features. *IMA J. Num. Anal.*, 13:413–430, 1993.

[JFH+98] P. J. Jessee, W. A. Fiveland, L. H. Howell, P. Colella, e R. B. Pember. An adaptive mesh refinement algorithm for the radiative transport equation. *Journal of Computational Physics*, 139:380–398, 1998.

[KA03] P. Knaber e L. Angermann. *Numerical Methods for Elliptic and Parabolic Partial Differential Equations*. Springer, New York, 2003.

[LL87] L. D. Landau e E. M. Lifschitz. *Fluid Mechanics*. Oxford Univ. Press & Butterworth–Heinemann, Oxford, 1987.

[Lyo95] L. Lyons. *All you wanted to know about mathematics*, volume 1. Cambridge University Press, Cambridge, 1995.

[MC96] D. Martin e K. Cartwright. Solving poisson's equation using adaptive mesh refinement. available on internet, october 1996.

[Min96] M. L. Minion. A projection method for locally refined grids. *Journal of Computational Physics*, 127:158–178, 1996.

[Mv96] J. D. Murray e W. vanRyper. *Encyclopedia of Graphics File Formats*. O'Reilly and Associates, Inc., Sebastopol CA, seconda edizione, 1996.

[Nie03] Y. Nievergelt. Scalar fused multiply-add instructions produce floating-point matrix arithmetic provably accurate to the penultimate digit. *ACM Transactions on Mathematical Software*, 29(1):27–48, March 2003.

[Par80] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs NJ, USA, 1980.

[PH05] D. A. Patterson e J. L. Hennessy. *Computer organization and design. The hardware/software interface.* Elsevier, 2005.

[PZ92] G. Pini e G. Zilli. *Esercizi di Calcolo Numerico.* Imprimitur, Padova, 1992.

[QS06] A. Quarteroni e F. Saleri. *Introduzione al Calcolo Scientifico.* Springer Verlag Italia, iii edizione, 2006.

[RM67] R. D. Richtmyer e K. W. Morton. *Difference Methods for Initial-Value Problems.* Interscience, New York, ii edizione, 1967.

[SB80] J. Stoer e R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, New York, 1980.

[SM83] G. Salton e M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[SP02] F. Sartoretto e M. Putti. *Introduzione alla Programmazione per elaborazioni numeriche.* Edizioni Libreria Progetto, Padova, 2002.

[Sta04] W. Stallings. *Architettura e organizzazione dei calcolatori.* Pearson, Milano, sesta edizione, 2004.

[The06] The MathWorks Inc., Natick, MA. *MATLAB. Complete documentation.*, 2006. Reperibile all' URL: `http://www.mathworks.com/access/helpdesk/help-/techdoc/matlab.html`, ultimo accesso: 1 settembre 2006.

[TOS00] U. Trottenberg, C. W. Oosterlee, e A. Schüller. *Multigrid.* Academic Press, London, 2000.

[TS64] A. N. Tychonov e A. A. Samarsky. *Partial Differential Equations of Mathematical Physics.* Holden-Day, San Francisco, 1964.

[Wei01] R. Weinands. *Extended Local Fourier Analysis for Multigrid: Optimal Smoothing, Coarse Grid Correction and Preconditioning.* PhD thesis, University of Köln, 2001.

[Wes92] P. Wesseling. *An Introduction to Multigrid Methods.* John Wiley and Sons Ltd, New York, 1992. Corrected Reprint. Philadelphia: R. T. Edwards Inc., 2004.

[Wik08] Wikipedia. Logistic function. Reperibile all' URL `http://en.wikipedia.org/wiki/Logistic_function`, Ultimo accesso: 8 marzo 2008.