

# Laboratorio di Analisi Numerica

## Lezione 4

Federico Poloni <f.poloni@sns.it>

17 Novembre 2010

**Quantità di esercizi:** in questa dispensa ci sono *più esercizi* di quanti uno studente medio riesce a farne durante una lezione di laboratorio, specialmente tenendo conto anche degli esercizi facoltativi. Questo è perché sono pensate per “tenere impegnati” per tutta la lezione anche quegli studenti che già hanno un solido background di programmazione. Quindi fate gli esercizi che riuscite, partendo da quelli *non* segnati come facoltativi, e non preoccupatevi se non li finite tutti!

*Esercizio 1* (di riscaldamento). Scrivere una funzione `function M=laplacian(n)` che crea la matrice di dimensione  $n \times n$  che ha 2 sulla diagonale e  $-1$  sulla sopra- e sotto-diagonale:

```
octave:1> laplacian(5)
ans =
  2 -1 0 0 0
 -1 2 -1 0 0
  0 -1 2 -1 0
  0 0 -1 2 -1
  0 0 0 -1 2
```

## 1 Soluzione di sistemi triangolari

### 1.1 Sistemi triangolari inferiori

$$\begin{pmatrix} L_{11} & 0 & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} & 0 \\ L_{51} & L_{52} & L_{53} & L_{54} & L_{55} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}$$

Possiamo risolverlo per sostituzione: a ogni passo, supponendo di avere calcolato  $x_1, \dots, x_{i-1}$  si ha

$$x_i = \frac{b_i - \sum_{j=0}^{i-1} L_{ij}x_j}{L_{ii}}$$

```
function x=inf_solve(L,b)
n=size(L)(1);
x=zeros(n,1);
for i=1:n
    p=b(i); % accumulatore
    for j=1:i-1
        p=p-L(i,j)*x(j);
    endfor
    x(i)=p/L(i,i);
endfor
endfunction
```

Testiamo la funzione con la matrice  $L=\text{tril}(\text{laplacian}(5))$  e il termine noto  $L*y$ , dove  $y$  è un vettore semplice.

```
octave:4> y=[1:5]'
```

```
y =
```

```
1
2
3
4
5
```

```
octave:5> y=[1:5]'
```

```
y =
```

```
1
2
3
4
5
```

```
octave:6> L=tril(laplacian(5))
```

```
L =
```

```
2 0 0 0 0
-1 2 0 0 0
0 -1 2 0 0
0 0 -1 2 0
0 0 0 -1 2
```

```

octave:7> inf_solve(L,L*y)
ans =

  1
  2
  3
  4
  5

```

*Esercizio 2.* Scrivere una `function` `sup_solve(U,b)` che risolva un sistema  $Ux = b$  con  $U$  triangolare superiore (hint: sostituire a partire dall'ultima riga). Testare su `U=triu(laplacian(5))`, `b=U*y` (con `y` vettore opportuno).

## 1.2 Soluzione di sistemi con la matrice di Laplace

Il comando `[L,U]=lu(laplacian(5))` restituisce due matrici, una  $L$  triangolare inferiore, e una  $U$  triangolare superiore, tali che  $L*U=laplacian(5)$ .

```

octave:9> [L,U]=lu(laplacian(5))
L =

  1.0000  0.0000  0.0000  0.0000  0.0000
-0.5000  1.0000  0.0000  0.0000  0.0000
  0.0000 -0.66667  1.0000  0.0000  0.0000
  0.0000  0.0000 -0.75000  1.0000  0.0000
  0.0000  0.0000  0.0000 -0.80000  1.0000

U =

  2.0000 -1.0000  0.0000  0.0000  0.0000
  0.0000  1.5000 -1.0000  0.0000  0.0000
  0.0000  0.0000  1.33333 -1.0000  0.0000
  0.0000  0.0000  0.0000  1.25000 -1.0000
  0.0000  0.0000  0.0000  0.0000  1.2000

octave:10> L*U-laplacian(5)
ans =

  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0

```

Utilizzando questa fattorizzazione e le funzioni `inf_solve` e `sup_solve` possiamo scrivere una funzione che risolve un sistema  $Ax = b$  con  $A=\text{laplacian}(5)$ : infatti,

$$A^{-1}b = (LU)^{-1}b = U^{-1}(L^{-1}b)$$

```
function x=lap_solve(n,b)
    [L,U]=lu(laplacian(n));
    y=inf_solve(L,b)
    x=sup_solve(U,y);
endfunction
```

Testiamo:

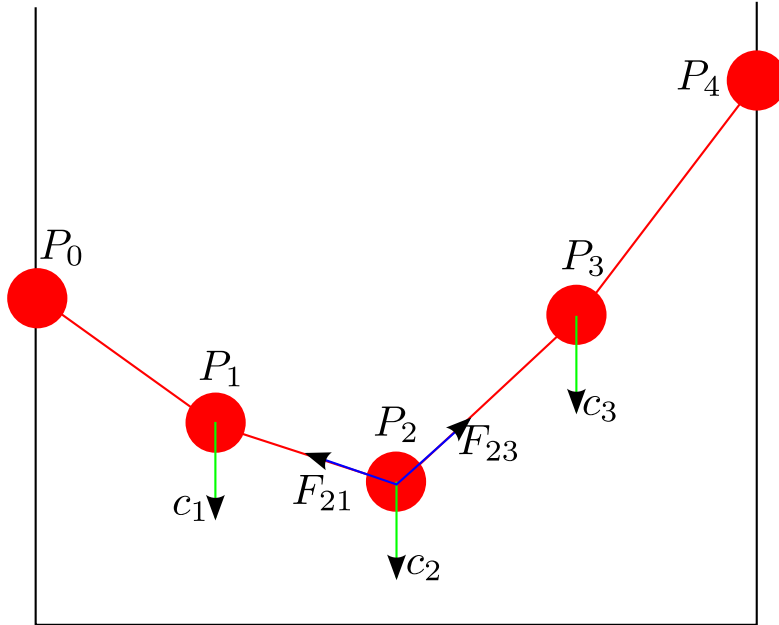
```
octave:20> A=laplacian(5); y=[1:5]';
octave:21> lap_solve(5,A*y)
ans =
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
```

Possiamo testarlo anche su matrici molto più grandi: per  $n = 1000$  sui computer del laboratorio dovrebbe impiegare meno di 20 secondi.

*Esercizio 3* (facoltativo). Qual è l'errore relativo (in norma-2) ottenuto sulla soluzione per diversi valori di  $n$ ? Come è legato al condizionamento della matrice `laplacian(n)`, che potete calcolare usando il comando `cond` di Octave?

### 1.3 Un problema fisico

Vogliamo determinare qual è la forma assunta da una corda (o da un elastico, o da un ponte) su cui sono appoggiati dei carichi. Modellizziamo la corda come una serie di "particelle"  $P_0, P_1, P_2, \dots, P_n, P_{n+1}$  disposte a intervalli regolari, a ognuna delle quali è applicato un certo peso. Supponiamo che  $x_0, y_0$  e  $x_{n+1}, y_{n+1}$  siano fissati. Supponiamo inoltre che le particelle possano muoversi solo in verticale, cioè che le  $x_i$  siano fissate a intervalli regolari.



Su ogni punto  $P_i$  interno alla corda agiscono tre forze:

- il carico  $c_i$  (verso il basso) che indica quanto peso viene appoggiato sul punto  $P_i$
- la forza  $F_{i,i+1}$  che “tiene incollato” il punto al punto alla sua destra; la sua componente verticale è circa uguale a  $k(y_{i+1} - y_i)$  (per un’opportuna costante elastica  $k$ )
- la forza  $F_{i,i-1}$  che “tiene incollato” il punto al punto alla sua sinistra; la sua componente verticale è circa uguale a  $k(y_{i-1} - y_i)$ .

L’equilibrio si ha quando  $c_i = k(y_{i+1} - y_i) + k(y_{i-1} - y_i)$ , o anche

$$-y_{i-1} + 2y_i - y_{i+1} = -c_i/k.$$

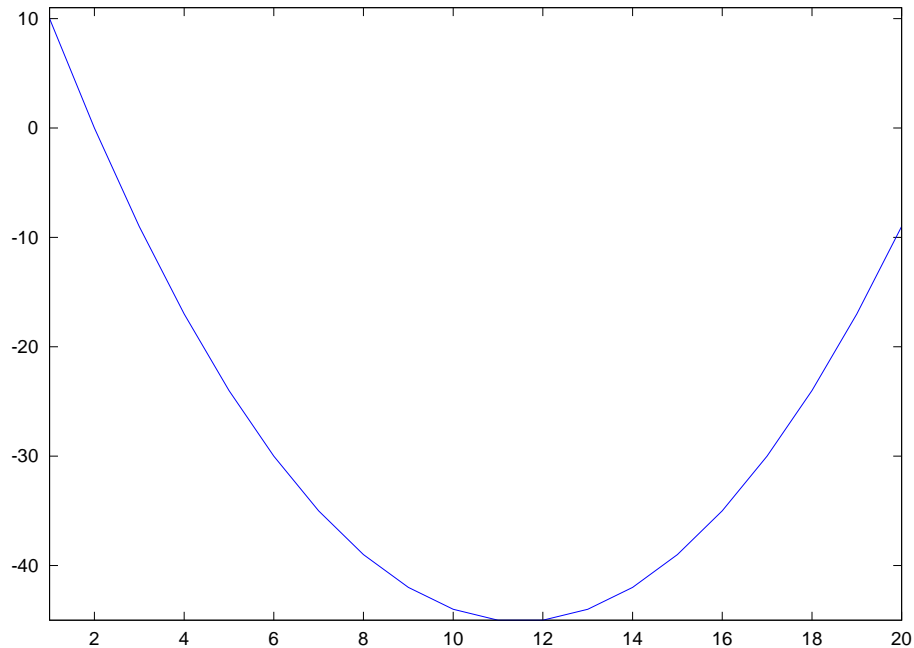
Possiamo scrivere queste equazioni come un sistema lineare  $Ly = b$ , dove  $L$  è la matrice di Laplace vista sopra,  $y = (y_1, y_2, \dots, y_n)$  sono le altezze dei punti interni al ponte, e

$$b = \begin{bmatrix} y_0 - c_1 \\ -c_2 \\ -c_3 \\ \vdots \\ -c_{n-1} \\ y_{n+1} - c_n \end{bmatrix}$$

(come mai il primo e l’ultimo termine sono diversi?). Possiamo porre  $k = 1$  (tanto dipende solo dalle unità di misura).

*Esercizio 4.* Ponete  $n = 20$ . Scrivete il termine noto  $b$  per i carichi  $c_i = 1$  per ogni  $i$ ,  $y_0 = y_{n+1} = 0$ . Risolvete il sistema per trovare le altezze dei punti, e disegnate il “ponte” con il comando `plot(1:20,y)`.

*Esercizio 5.* Provate a cambiare i carichi aggiungendo dei pesi in punti specifici del ponte. Provate a cambiare le altezze dei due estremi del “ponte”.



*Esercizio 6 (facoltativo).* Calcolate risolvendo un sistema simile anche le ascisse  $x$  dei punti. Qual è il risultato? Come ve lo spiegate?

## 2 Facoltativo: sfruttare la struttura

### 2.1 Matrici bidiagonali

Notiamo che le matrici  $L$  e  $U$  per le matrici `laplacian(n)` hanno una struttura particolare: tutti gli elementi sono nulli tranne quelli sulla diagonale principale e sulla sottodiagonale (per  $L$ ) o sopradiagonale (per  $U$ ).  $L$  e  $U$  sono dette *bidiagonali*. Possiamo sfruttare esplicitamente questo fatto nei nostri algoritmi.

*Esercizio 7 (facoltativo).* Scrivere due funzioni `inf_solve2` e `sup_solve2` che risolvano i sistemi  $Ly = b$  e  $Ux = y$  supponendo  $L$  e  $U$  bidiagonali (facendo meno calcoli di `inf_solve` e `sup_solve`!). Poi scrivere la funzione `lap_solve2` analoga a `lap_solve` ma che utilizza le due funzioni appena scritte.

Quanto tempo impiegano le due funzioni?

```
octave:27> b=laplacian(1000)*[1:1000]';  
octave:28> tic;lap_solve(1000,b); toc  
ans = 11.669  
octave:29> tic;lap_solve2(1000,b); toc  
ans = 2.1535
```

**Morale della favola:** quando una matrice ha una struttura dobbiamo sempre cercare di sfruttarla nei calcoli; il guadagno di tempo (e a volte anche di precisione dei risultati) può essere notevole.