Laboratorio computazionale numerico
Lezione 4

Federico Poloni <f.poloni@sns.it>

2009-11-04

1 Dalla volta scorsa...

Esercizio 1. Testare i seguenti metodi di soluzione di un sistema lineare $Ax = b$:

- L'istruzione x=sys_solve(A,b), che utilizza la funzione contenuta nel file `http://poisson.dm.unipi.it/~poloni/dida/lcn09/lezione%204/sys_solve.m` (da scaricare!), che contiene gli algoritmi della sezione precedente (fattorizzazione LU senza pivoting + soluzione di due sistemi triangolari).
- Il comando di Octave x=**inv**(A)∗b, che calcola la matrice inversa e la moltiplica per b.
- Il comando di Octave x=A\b: il comando \ (barra rovesciata) serve proprio per risolvere sistemi lineari, ed è basato sulla fattorizzazione LU con pivoting parziale[1].

Per testarli, utilizzate le seguenti matrici:

- La matrice M1=9∗**eye**(10)+ones(10), che è dominante diagonale.
- La matrice M2=**rand**(10), che è una matrice con elementi casuali — può essere abbastanza mal condizionata! Potete controllare il condizionamento con il comando **cond**(M2).
- La matrice data da M3=M1;M3(9,1:9)=0, che ha una riga quasi tutta di zeri che rende la sottomatrice principale 9×9 singolare (e quindi non ammette fattorizzazione LU).
- La matrice data da M4=M2;M4(9,1:9)=**sum**(M4(1:8,1:9)):

  ```
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  ```

  gli elementi in rosso sono ognuno la somma degli otto elementi che stanno direttamente sopra di esso; quindi la sottomatrice principale 9×9 è singolare. (Ma lavorando con i numeri floating point...)

[1] Non vi ho fatto scrivere una fattorizzazione LU con pivoting parziale per mancanza di tempo, ma non è niente di particolarmente complicato.

- La matrice data da M5=M2;M5(10,1:10)=**sum**(M5(1:9,1:10)): l'ultima riga è la somma delle 9 precedenti, quindi la matrice è singolare. (Ma lavorando con i numeri floating point...)

Ponete v=(1:10)' (vettore colonna conentente i numeri da 1 a 10 in ordine); per ognuna di queste matrici, calcolate bi=v (per $i = 1, \ldots, 4$), e andate a risolvere il sistema Mi∗x=bi. La soluzione esatta di questo sistema è v; di quanto si discosta la soluzione calcolata?

2 Ancora determinanti

Esercizio 2. Scrivete una **function** d=mydet_gauss(A) che calcoli il determinante di una matrice riducendola in forma triangolare con l'eliminazione di Gauss. Per ora, ignorate il pivoting (scambi di righe quando incontrate uno zero).
Hint: invece di usare cicli **for** come

```
for j=1:n
  A(i1,j)=A(i1,j)+ alpha * A(i2,j);
endfor
```

usate singole istruzioni sui vettori come

```
A(i1,1:n)=A(i1,1:n) + alpha * A(i2,1:n);
```

Esercizio 3. Aggiungete il *pivoting* (scambi di righe per rendere più stabile l'algoritmo) all'esercizio precedente.
Hint: Questo ci porta a due piccoli problemi di programmazione che dovreste avere già affrontato in passato (laboratorio di C?).

Il primo è: come faccio a scambiare due righe (o più in generale il contenuto di due variabili a e b)? L'algoritmo classico utilizza un valore temporaneo *tmp*:

```
tmp=a;
a=b;
b=tmp;
```

Il secondo è: come faccio a trovare in quale colonna sta il massimo elemento di un vettore? Questo si fa con un ciclo **for**, scandendo il vettore e tenendo in un "accumulatore" temporaneo l'indice del massimo valore trovato finora:

```
max_pos=1; %indice del massimo elemento
for i=2:n
  if(v(i)>v(max_pos))
    max_pos=i;
  endif
endfor
```

(Se volete usare la funzione già presente in Octave [max_val,max_pos]=**max**(v), che fa la stessa cosa, occhio a da dove partono gli indici!)

3 Eliminazione di Gauss "senza L"

Se avete già finito e vi state annoiando...

Ignoriamo per il momento il pivoting e cominciamo a scrivere una function che risolva sistemi lineari con l'eliminazione di Gauss. Si possono riorganizzare

i calcoli in un modo in cui il calcolo di L non viene effettuato implicitamente (che probabilmente è come avete visto l'eliminazione di Gauss lo scorso anno ad Algebra Lineare, inizialmente). Lavoriamo sulle equazioni anziché sulle matrici:

$$\begin{array}{llll} a_{11}x_1 & +a_{12}x_2 & +a_{13}x_3 & = b_1. \\ a_{21}x_1 & +a_{22}x_2 & +a_{23}x_3 & = b_2. \\ a_{31}x_1 & +a_{32}x_2 & +a_{33}x_3 & = b_3. \end{array}$$

Cerchiamo di eliminare x_1 dalla seconda e terza equazione:

$$\begin{array}{llll} a_{11}x_1 & +a_{12}x_2 & +a_{13}x_3 & = b_1. \\ a_{21}x_1 & +a_{22}x_2 & +a_{23}x_3 - \frac{a_{21}}{a_{11}}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & = b_2 - \frac{a_{21}}{a_{11}}b_1. \\ a_{31}x_1 & +a_{32}x_2 & +a_{33}x_3 - \frac{a_{31}}{a_{11}}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & = b_3 - \frac{a_{31}}{a_{11}}b_1. \end{array}$$

Quindi reiteriamo, fino a trasformare A in una matrice triangolare.

Prendiamo come riferimento per l'analisi il primo passo. Come vengono modificati quindi la matrice dei coefficienti (A) e il termine noto (b) che stiamo tenendo in memoria? Sulla matrice A facciamo le stesse operazioni che abbiamo fatto la scorsa lezione nel calcolo della fattorizzazione LU. Come abbiamo visto, la quantità che dobbiamo sottrarre ad A è una matrice di rango 1:

$$\texttt{A(2:3,1:3)} = \texttt{A(2:3,1:3)} - \begin{bmatrix} \frac{a_{21}}{a_{11}} \\ \frac{a_{31}}{a_{11}} \end{bmatrix} * \begin{bmatrix} a_{11} & a_{12} & a_{13} \end{bmatrix}$$

In più, dobbiamo effettuare un calcolo simile anche su b:

$$\texttt{b(2:3)} = \texttt{b(2:3)} - \begin{bmatrix} \frac{a_{21}}{a_{11}} \\ \frac{a_{31}}{a_{11}} \end{bmatrix} * b_1$$

Possiamo anche interpretare i calcoli effettuati in termini di prodotti di matrici: nel calcolo della fattorizzazione LU ottenevamo L^{-1} come prodotto di matrici parziali $L_n L_{n-1} \cdots L_1$, ognuna della forma

$$\begin{bmatrix} I & & \\ & 1 & \\ & v & I \end{bmatrix},$$

ora quello che facciamo è effettuare subito il prodotto con la matrice parziale

$$\begin{bmatrix} 1 & & \\ -\frac{a_{21}}{a_{11}} & 1 & \\ -\frac{a_{31}}{a_{11}} & 0 & 1 \end{bmatrix}$$

in modo da trasformare il sistema nel sistema equivalente (che ha la stessa soluzione)

$$L_1 A x = L_1 b,$$

Al termine degli n passi, abbiamo trasformato il sistema in un sistema equivalente ma con matrice dei coefficienti triangolare, e questo lo sappiamo risolvere.

3.1 Pivoting parziale

Nel nostro algoritmo, effettuare pivoting parziale vuol dire, a ogni passo, scambiare tra loro le equazioni in modo da portare il cima quella con il valore maggiore del primo elemento a_{i1}. Non è necessario calcolare esplicitamente la matrice di permutazione, basta scambiare materialmente le righe nella matrice (e nel termine noto!).

4 Soluzione di un sistema lineare con fattorizzazione QR implicita (Householder)

se avete finito e vi state annoiando...

4.1 Trasformazioni elementari di Householder

Il vettore v che determina la riflessione $I - \frac{2vv^T}{\|v\|_2^2}$ che porta x in un multiplo di e_1 è

$$x \pm \|x\|_2 e_1$$

In particolare, v è uguale a x tranne per il primo elemento, che vale

$$v_1 = x_1 - \|x\|_2 = \frac{-(x_2^2 + \cdots + x_n^2)}{x_1 + \|x\|_2}$$

oppure

$$v_1 = x_1 + \|x\|_2 = \frac{-(x_2^2 + \cdots + x_n^2)}{x_1 - \|x\|_2}.$$

In una di queste, il denominatore soffre di errori di cancellazione; quale? (dipende dal segno di $x_1 \ldots$)

Scrivete una funzione v=householder_vector(x) che calcoli v con il segno giusto.

4.2 Mettere insieme tutto

Ora potete scrivere una funzione **function** x=qr_solve(A,b) che risolva un sistema lineare attraverso la fattorizzazione QR: per esempio, al primo passo, invece di trovare una matrice triangolare L_1 che mandi la prima riga di A in un multiplo di e_1 (come facevate in gepp), dovete trovare una matrice di Householder H_1 che faccia lo stesso lavoro e applicarla sia a A che a b per ottenere un sistema equivalente:

$$H_1 Ax = H_1 b.$$

Esercizio 4. Cosa succede se usate il segno sbagliato nella funzione householder_vector, o se usate sempre lo stesso segno? Testare provando a risolvere qualche sistema lineare.