

Laboratorio di Analisi Numerica

Lezione 6

Leonardo Robol <leonardo.robol@unipi.it>

Igor Simunec <igor.simunec@sns.it>

5 Novembre 2021

Quantità di esercizi: in questa dispensa ci sono *più esercizi* di quanti uno studente medio riesca a farne durante una lezione di laboratorio, specialmente tenendo conto anche degli esercizi facoltativi. Questo è perché è pensata 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!

1 Usare le fattorizzazioni

1.1 La fattorizzazione QR

La fattorizzazione QR è una decomposizione $A = QR$ dove

- Q è una matrice unitaria (i.e., $Q^T Q = I$).
- R è triangolare superiore

Reminder: Per calcolare la (o, per essere più precisi, una) fattorizzazione QR di A si possono usare le matrici elementari di Householder. Queste sono matrici del tipo

$$P = I - \beta uu^T \quad \beta = \frac{2}{u^T u}.$$

Come nel caso delle matrici di Gauss è possibile scegliere u in modo che $Pv = \alpha e_1$. In particolare funzionano le scelte $u = v \pm \|v\|_2 e_1$.

Esercizio 1. Scrivere una funzione `P = householder_matrix(v)` che, dato il vettore v , calcoli la matrice elementare di Householder che trasformi v in un multiplo di e_1 .

Esercizio 2. Realizzare una funzione `[Q,R] = my_qr(A)` che calcoli la fattorizzazione QR di una matrice A . Provare a sostituire questa fattorizzazione alla LU nella funzione `x = sys_solve(A,b)` realizzata la volta scorsa. Attenzione: la matrice Q non è triangolare inferiore, ma il sistema $Qy = b$ è ugualmente risolvibile a costo basso: come?

Esercizio 3 (facoltativo). La scelta del segno nel calcolo delle matrici di Householder è importante per evitare cancellazione. Qual è il criterio da usare? Provare a confrontare i risultati ottenuti da `sys_solve(A,b)` al variare di questa scelta.

1.2 Calcolare il determinante

Una delle applicazioni immediate delle fattorizzazioni QR ed LU è il calcolo del determinante. Dalle proprietà di L e Q nelle fattorizzazioni $A = LU = QR$ segue immediatamente che $\det(L) = |\det(Q)| = 1$ (perché?). Questo ci permette di calcolare

$$\det(A) = \det(U) = \text{segno}(\det(Q)) \det(R).$$

Ricordando che sia R che U sono triangolari superiore concludiamo che il determinante è il prodotto degli elementi diagonali (e quindi molto facile da calcolare!). Rimane da risolvere il problema del segno di $\det(Q)$ (come lo possiamo calcolare?).

Esercizio 4. Realizzare una funzione `d = mydet3(A)` che calcoli il determinante di una matrice A utilizzando la sua fattorizzazione LU .

Esercizio 5 (facoltativo). Scrivere una funzione `mydet4(A)` che calcoli il determinante utilizzando la fattorizzazione QR invece che quella LU . Quali sono i vantaggi (e gli svantaggi) di questa seconda implementazione?

Esercizio 6 (facoltativo). Realizzare una funzione `charpoly(A,x)` che valuti $\det(xI - A)$. Controllare la correttezza confrontando i risultati con quelli ottenuti valutando il polinomio caratteristico di A restituito da MATLAB (digitare `help poly` e `help polyval` per approfondire).

1.3 Sfruttare la struttura

Spesso le matrici ottenute da problemi reali hanno una particolare struttura. Quando possibile conviene cercare di sfruttare questa caratteristica per ottenere vantaggi computazionali.

Possiamo ad esempio considerare il caso di A “a banda”, ovvero tale che $a_{ij} = 0$ se $|i - j| > q$ per qualche q fissato.

Esercizio 7. Mostrare che se A ha la struttura sopra anche i fattori L ed U della sua fattorizzazione LU la possiedono.

Esercizio 8. Modificare la funzione `my_lu(A)` scritta la scorsa lezione in modo da sfruttare questa struttura. Trasformarla in una funzione `my_lu2(A,q)` che prenda come input anche la larghezza della banda q . Confrontate poi i tempi utilizzando `tic` e `toc`. Ad esempio, per una matrice 1000×1000 con ampiezza di banda 3:

```
>> n = 1000;
>> A = zeros(n);
>> for i = -3:3
>> A = A + diag(randn(n - abs(i),1), i)
>> end
```

```
>> tic; [L,U] = my_lu2(A, 3); toc
Elapsed time is 0.21765 seconds.
>> tic; [L,U] = my_lu(A); toc
Elapsed time is 3.8512 seconds.
```

1.4 Fattorizzazione QR con matrici di Givens

L'applicazione di una matrice di Householder ad A consente di annullare simultaneamente tutti gli elementi di una colonna di A al di sotto della diagonale principale. Vi sono però casi, ad esempio per matrici con specifiche proprietà di struttura, in cui è necessario annullare solo particolari elementi.

Esercizio 9. Si generi una matrice sparsa con il comando `sprand(n, m, density)` scegliendo $n = m = 10$ e `density = 0.5`. Si applichi una trasformazione di Householder a questa matrice per annullare gli elementi sottodiagonali della prima colonna. Cosa succede agli elementi che originariamente erano zero nella rimanente parte della matrice? (Si consiglia di utilizzare comando `full(A)` per trasformare la matrice sparsa A nell'usuale formato denso). Potete utilizzare la funzione `nnz(A)` per controllare quanti elementi in A sono non-zero. La quantità `nnz(A) / prod(size(A))` dovrebbe corrispondere con il parametro di densità da voi indicato. Controllate come varia invece lo stesso valore per la Q e la R ottenute con `[Q,R] = qr(full(A));`

Esercizio 10. Si consideri la seguente matrice “freccia” ottenuta tramite le seguenti istruzioni in MATLAB:

```
>> n = 12; % potete provare anche con altre dimensioni
>> A = diag(randn(1,n));
>> A(1,:) = randn(1,n);
>> A(:,1) = randn(n,1);
```

Questa matrice ha una notevole struttura. Osservatela con l'istruzione `spy(A)`.

Cosa succede dopo un passo di Householder? Provate a calcolare P in modo da pulire la prima colonna di A e poi eseguite `spy(P*A)`.

Hint: Il comando `spy` a volte è piuttosto pignolo su cosa considerare 0. Per pulire gli elementi piccoli provate a dare il comando `spy(P*A .* (abs(P*A) > 1e-12))`.

Esiste un'alternativa alle matrici di Householder che si comporta meglio con matrici sparse: le rotazioni di Givens. Queste si costruiscono a partire dalla seguente osservazione: se $v \in \mathbb{R}^2$ possiamo determinare una rotazione G tale che $Gv = \|v\|e_1$. In questo modo eliminiamo la seconda componente di v .

Se $v \in \mathbb{R}^n$ possiamo “pulire” una sua componente utilizzandone un'altra generalizzando la strategia sopra. Ad esempio, supponiamo di volere uno 0 in posizione j modificando v solo nella posizione i :

- Consideriamo lo spazio bidimensionale $V = \langle e_i, e_j \rangle$.
- Proiettiamo v sullo spazio V : otteniamo $P_V v = [v_i \ v_j]^T$. Esiste dunque una trasformazione G tale che $GP_V v = [\alpha \ 0]^T$.

- Consideriamo I_{V^\perp} l'identità su V^\perp e definiamo $G_{ij} = G \oplus I_{V^\perp}$. Questa matrice trasforma v “pulendo” la componente j modificando solo la componente i , come desideravamo.

Ricordiamo che una rotazione di \mathbb{R}^2 è rappresentata da una matrice del tipo

$$R = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, \quad c^2 + s^2 = 1.$$

Più concretamente, la matrice G_{ij} che si utilizza per eliminare un elemento non zero nella prima colonna di A è di questo tipo:

$$G_{ij} = \begin{bmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & c & & & -s & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & s & & & & & c & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix}$$

in cui c e s sono scelti in modo tale che $B = G_{ij}S$, che differisce da S solo per gli elementi nelle righe di indice i e j , abbia $b_{j1} = 0$. In particolare se $a_{j1} = 0$ si pone $c = \text{segno}(a_{i1})$, $s = 0$. Se $a_{i1} = 0$ si pone $c = 0$ e $s = -\text{sign}(a_{j1})$, altrimenti si calcolano s e c secondo il seguente schema

$$\begin{aligned} t &= \sqrt{a_{j1}^2 + a_{i1}^2} \\ c &= a_{i1}/t; \\ s &= -a_{j1}/t; \end{aligned}$$

Esercizio 11. Si scriva una funzione **function** `G=mygivens(v)` che prenda in input un vettore colonna con due componenti e costruisca la matrice di Givens G che trasforma v in αe_1 . Confrontate la vostra implementazione con la funzione `planerot` di MATLAB (esiste anche la funzione `givens` che funziona in modo simile).

Esercizio 12. Si utilizzi **function** `G=mygivens(v)` per eliminare gli elementi non nulli nella parte triangolare inferiore di A dell'esercizio 9. Si verifichi che le matrici di Givens sono matrici ortogonali. Hint: Non è necessario formare esplicitamente la matrice G_{ij} che contiene G : è sufficiente fare agire la matrice piccola sulle righe di A utilizzando la sintassi di selezione righe di MATLAB.

Esercizio 13. Si scriva una funzione **function** `[Q,R]=qr_givens(A)` per calcolare la fattorizzazione $A = QR$ di A mediante matrici di Givens. Come si ricavano R e Q dalle matrici di Givens e da A ? Provate a sostituire questa fattorizzazione alla LU nella funzione `x = sys_solve(A,b)`.