

Laboratorio di Analisi Numerica

Lezione 3

Gianna Del Corso <delcorso@di.unipi.it>

Federico Poloni <fpoloni@di.unipi.it>

16 Ottobre 2012

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!

1 Sottomatrici e determinanti

Utilizzando l'operatore `:`, in Octave è possibile selezionare un'intera sottomatrice di una matrice:

```
octave:1> A=[1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```
 1 2 3
 4 5 6
 7 8 9
```

```
octave:2> A(1:2,2:3)
```

```
ans =
```

```
 2 3
 5 6
```

```
octave:3> A(2:end,1:end-2)
```

```
ans =
```

```
 4
 7
```

```

octave:4> A(1:end,1:end)
ans =

    1  2  3
    4  5  6
    7  8  9

octave:5> A(1,:)
ans =

    1  2  3

```

La sintassi `a:b` seleziona tutte le righe/colonne comprese tra `a` e `b` (estremi inclusi). Il valore `end` viene sostituito con il massimo indice disponibile. Il solo `:` è un'abbreviazione per `1:end`.

Possiamo anche assegnare un valore a una sottomatrice selezionata in questo modo:

```

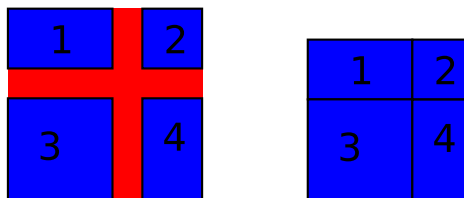
octave:5> A(1:2,1:2)=eye(2)
A =

    1  0  3
    0  1  6
    7  8  9

```

Ovviamente le dimensioni devono essere compatibili: non posso selezionare una sottomatrice 2×2 e assegnarle il valore `eye(3)`!

La seguente function ritorna la *matrice minore* di (i, j) in A , cioè la matrice che si ottiene eliminando la i -esima riga e la j -esima colonna di A .



```

function B=minor(A,i,j)
X=A(1:i-1,1:j-1);
Y=A(1:i-1,j+1:end);
Z=A(i+1:end,1:j-1);
W=A(i+1:end,j+1:end);
B=[X Y; Z W];
endfunction

```

Abbiamo già visto che se X, Y, Z, W sono numeri, la riga di codice `B=[X Y; Z W]` crea la matrice

$$\begin{bmatrix} X & Y \\ Z & W \end{bmatrix};$$

ora vediamo che la stessa sintassi funziona anche se X, Y, Z, W sono matrici di dimensioni “compatibili” e crea la matrice formata accostando i quattro blocchi.

Esercizio 1. Creare una **function** `d=mydet(A)` che calcoli il determinante di una matrice quadrata A utilizzando la formula di Laplace sulla prima riga, cioè

$$\det(A) = A_{11} \det A^{(11)} - A_{12} \det A^{(12)} + A_{13} \det A^{(13)} - \dots + (-1)^{n+1} A_{1n} \det A^{(1n)}$$

dove A_{ij} è l'elemento di A nella posizione (i, j) e $A^{(ij)}$ è la matrice minore di A rispetto a (i, j) . Hint: la funzione può essere *ricorsiva*, cioè chiamare sé stessa al suo interno. Occhio a definire un caso base!

Poi testarla su alcune matrici, confrontandola con la funzione `det` di Octave, per esempio le matrici di Vandermonde `V=vander(1:2)`, `V=vander(1:3)`, `V=vander(1:4)`...

2 Tempi di calcolo

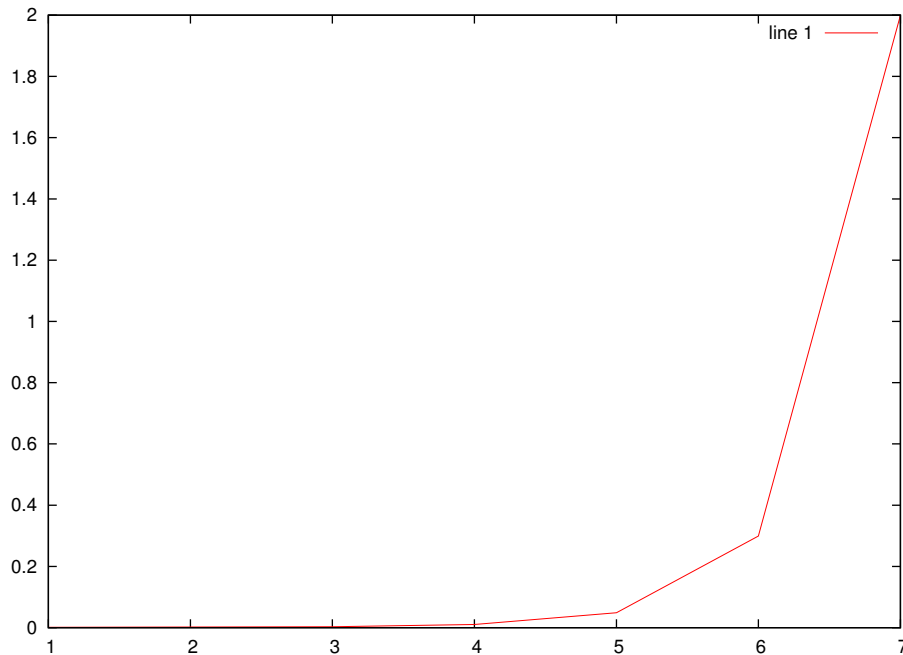
Le funzioni `tic` e `toc` misurano il tempo necessario ad eseguire più istruzioni. La prima fa partire il cronometro, la seconda lo ferma e restituisce il valore ottenuto. Per esempio, le istruzioni

```
tic;
x=10;
myexp2(x,500);
t=toc;
```

salvano in `t` il tempo necessario ad eseguire le due righe centrali.

La seguente funzione disegna un grafico del tempo impiegato per calcolare i determinanti delle matrici `vander(1:n)` con `n=1:7`.

```
function plottimes();
n=7;
tempi=zeros(n,1); %prepara un vettore vuoto con i tempi
for i=1:n
    A=vander(1:i); %la matrice viene generata prima del ‘tic’:perche?
    tic;
    d=mydet(A);
    tempi(i)=toc;
endfor
plot(1:n,tempi);
endfunction
```



I tempi di calcolo crescono molto velocemente. Già per $n = 9$ la funzione è praticamente inutilizzabile. Difatti, con questo algoritmo, per calcolare un determinante di dimensione n dobbiamo calcolarne n di dimensione $n - 1$; quindi vale

$$\text{tempo}(n) \approx n \cdot \text{tempo}(n - 1)$$

da cui $\text{tempo}(n) \approx n! \cdot \text{tempo}(1)$. Il nostro algoritmo quindi non è adatto a calcolare il determinante in modo efficiente.

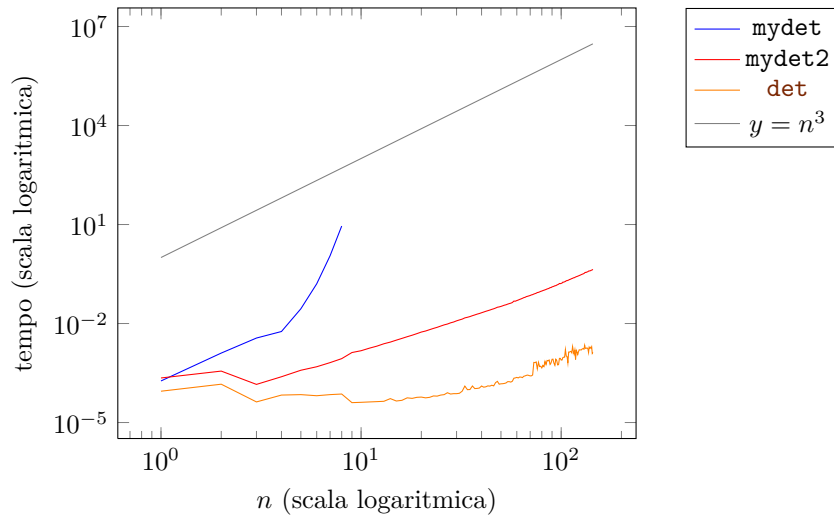
3 Determinante con l'eliminazione di Gauss

Esercizio 2. Scrivete una **function** `d=mydet2(A)` che calcoli il determinante utilizzando l'eliminazione di Gauss. Suggestioni:

- utilizzate espressioni del tipo `A(i,:)=A(i,:)+A(j,:)` per sommare due righe di una matrice, invece di scrivere un ciclo `for`.
- per ora, ignorate il fatto che i pivot possono essere zero; se incontrate un pivot nullo, terminate con un messaggio di errore (potete farlo con l'istruzione `error('ho incontrato un pivot nullo')`).
- per controllare che tutto vada bene, in una prima fase fatevi scrivere a schermo il valore di `A` dopo ogni passo di eliminazione di Gauss

Esercizio 3. Disegnare un grafico analogo a quello più sopra per i tempi di calcolo di `mydet2`.

Esercizio 4. Ecco un grafico più completo, disegnato in scala logaritmica.



Per riferimento abbiamo riportato anche il grafico di $y = n^3$, sempre sulla stessa scala logaritmica. Qual è la pendenza della retta che corrisponde ad esso? Come potete determinare dal grafico che `mydet2` richiede $O(n^3)$ operazioni mentre `mydet` è asintoticamente più costoso? La funzione `det` di Octave dovrebbe essere anch'essa cubica, ma il suo comportamento è più complicato da determinare per via delle ottimizzazioni usate; i tempi esatti dipendono fortemente dall'architettura del calcolatore.

Esercizio 5. Testate la funzione `mydet2` su alcune matrici per cui dovrebbe generare un errore a causa del pivot nullo. Per esempio, generate usando `rand` una matrice 10×10 in cui tutte le entrate sono casuali, tranne la nona riga che è la somma delle otto righe precedenti. Cosa succede? Incontrate effettivamente un errore? Perché?

Esercizio 6 (facoltativo). Provare a disegnare un grafico simile a quello qui sopra, che riporti i tempi dei tre diversi algoritmi in scala logaritmica.

Esercizio 7 (facoltativo). Scrivete una funzione che scambi opportunamente le righe quando incontra un pivot nullo, in modo da poter continuare l'algoritmo. Occhio: cosa succede al determinante quando scambio due righe?

Testate la funzione su alcune matrici, per esempio `A=vander(1:10)`, confrontandola con `det()` di Octave.