Funzioni per la costruzione di matrici

Lucia Gastaldi

Dipartimento di Matematica,
http://lucia-gastaldi.unibs.it

Indice

eye

Il comando eye serve per costruire la matrice identità, ossia

$$I = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

`eye(N)` fornisce la matrice identità di dimensione $\mathtt{N} \times \mathtt{N}$
`eye(M,N)` oppure `eye([M,N])` è una matrice di dimensione $\mathtt{M} \times \mathtt{N}$ con 1 sulla diagonale principale e 0 altrove.
`eye(SIZE(A))` è una matrice identità con le stesse dimensioni di A.

zeros e ones

`ones` e `zeros`

I comandi `ones` e `zeros` permettono di costruire un array che ha componenti tutte uguali ad 1 oppure a 0.

`b=ones(10,1)` costruisce il vettore colonna b di 10 componenti tutte uguali a 1.

`c=zeros(3,3)` costruisce una matrice 3×3 di elementi tutti uguali a 0.

DIAG

Il comando **diag** ha due funzioni diverse:

Costruzione di una matrice con una sola diagonale non nulla

Sia `V` un vettore di `N` componenti.
`diag(V,K)` è una matrice quadrata di ordine `N+ABS(K)` che ha gli elementi di `V` sulla diagonale `K`-esima.

se `K`$= 0$ è la diagonale principale
se `K`> 0 si trova sopra la diagonale principale
se `K`< 0 si trova sotto la diagonale principale

Estrazione di una diagonale da una matrice

Sia `A` una matrice di dimensione $n \times n$ il comando

```
d=diag(A,k)
```

fornisce il vettore
$d = [a_{1,1+k}, \ldots, a_{n-k,n}]^t$ se $k \geq 0$
$d = [a_{1+k,1}, \ldots, a_{n,n-k}]^t$ se $k < 0$.

tril e triu

I comandi `tril` e `triu` servono per estrarre la parte **triangolare inferiore** e **triangolare superiore** di una matrice.

Data una matrice `A` di dimensione nxn
`tril(A)` è la sottomatrice triangolare inferiore di `A`
`tril(A,K)` fornisce la sottomatrice di `A` formata dagli elementi che si trovano sotto o sulla diagonale `K`-esima.
`triu(A)` è la sottomatrice triangolare superiore di `A`.
`triu(A,K)` fornisce la sottomatrice di `A` formata dagli elementi che si trovano sopra o sulla diagonale `K`-esima.

Norme di vettore

La function `norm` calcola la norma euclidea di un vettore ossia

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

La sintassi del comando è: `nv=norm(v)`
essendo `v` il nome del vettore di cui si vuole calcolare la norma e `nv` il nome della variabile a cui si assegna il valore della norma del vettore.

Il comando `norm(v,Inf)` calcola la norma del massimo cioè
$\|v\|_\infty = \max_{1\le i\le n} |v_i|$.

Il comando `norm(v,p)` calcola la seguente norma
$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p}$.

Norme di matrice

Nel caso di matrici, la function `norm` calcola la seguente norma di matrice associata alla norma euclidea di vettore:

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

essendo $\rho(B)$ il **raggio spettrale** della matrice B ossia il massimo autovalore in modulo:

$$\rho(B) = \max_{1 \leq \lambda_i \leq n} |\lambda_i| \quad \text{dove } \lambda_i,\ i = 1, \ldots, n \text{ sono gli autovalori di } B.$$

La sintassi del comando è: `nA=norm(A)`
essendo `A` il nome della matrice di cui si vuole calcolare la norma e `nA` il nome della variabile a cui si assegna il valore della norma della matrice.

Norme di matrice

Il comando `norm(A,p)` fornisce il valore delle seguenti norme di matrice:

- $p = 1$ $\qquad \|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|;$
- $p = 2$ $\qquad \|A\|_2 = \sqrt{\rho(A^T A)};$
- $p = \texttt{Inf}$ $\qquad \|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|;$
- $p = \texttt{'fro'}$ $\qquad \|A\|_F = \left(\sum_{i,j=1}^{n} |a_{ij}|^2 \right)^{1/2}.$

Numero di condizionamento

Il **numero di condizionamento** di una matrice A è dato da:

$$K(A) = \|A\| \, \|A^{-1}\|.$$

Il comando `cond(A,p)` fornisce il numero di condizionamento di A nella norma `p`.
`p` assume i seguenti valori: `1`, `2`, `Inf`, `'fro'`.

`rcond(A)` è una stima del reciproco del numero di condizionamento di A calcolato nella norma 1 mediante un estimatore di *LAPACK*.
Se la matrice A è ben condizionata allora `rcond(A)` è vicino a 1, se A è mal condizionata allora `rcond(A)` è vicino a `eps`.

Formato `sparse`

Il formato sparse è utilizzato in Matlab per ridurre i costi di memorizzazione della matrice.

`S=sparse(A)` converte la matrice in formato full in una matrice in formato sparse tenendo in memoria solo gli elementi diversi da zero.

`S=sparse(m,n)` genera una matrice in formato sparse con tutti gli elementi nulli.

`S = sparse(i,j,s,m,n)` usa i vettori `i`, `j` e `s` per costruire la matrice di dimensione $m \times n$ tale che `S(i(k),j(k)) = s(k)`.

Il comando `spy(A)` mostra in un grafico quali sono gli elementi di A non nulli ed il loro numero `nnz`.

Il comando `spdiags`

Il comando `spdiags` generalizza il comando `diag`.

Sono disponibili quattro operazioni differenti.

- `B = spdiags(A)` estrae tutte le diagonali non nulle dalla matrice A. Le p colonne di B sono le diagonali di A. `[B,d] = spdiags(A)` fornisce anche il vettore d di lunghezza p, i cui valori specificano le diagonali di A.
- `B = spdiags(A,d)` estrae le diagonali specificate da d.
- `A = spdiags(B,d,A)` sostituisce le diagonali specificate da d con le colonne di B.
- `A = spdiags(B,d,m,n)` crea una matrice sparsa $m \times n$ prendendo le colonne di B e mettendole al posto delle diagonali specificate da d.

Matrici a blocchi

Matrici a blocchi

Se una matrice è decomposta in matrici di dimensioni più piccole allora si parla di **matrice a blocchi**. Ad esempio la matrice $A \in \mathbb{R}^{m \times n}$ è data da

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1p} \\ A_{21} & A_{22} & \dots & A_{2p} \\ \dots & \dots & \dots & \dots \\ A_{q1} & A_{q2} & \dots & A_{qp} \end{bmatrix}$$

dove le matrici A_{ij} hanno dimensione $m_i \times n_j$ essendo $\sum_{i=1}^{q} m_i = m$ e $\sum_{j=1}^{p} n_j = n$.

La costruzione in Matlab di una matrice a blocchi viene fatta come nel caso di una matrice di scalari.

Matrici a blocchi

Esercizio

Costruire la matrice

$$A = \begin{bmatrix} 100 & 10 & 10 & 10 & 100 & 100 \\ 10 & 2 & -1 & 0 & 20 & 20 \\ 10 & -1 & 2 & -1 & 20 & 20 \\ 10 & 0 & -1 & 2 & 20 & 20 \\ 100 & 10 & 10 & 10 & 100 & 100 \end{bmatrix}$$

usando opportunamente i comandi `ones` e `diag`, in modo tale che la matrice possa essere costruita con una dimensione n arbitraria della matrice tridiagonale interna.