

# Laboratorio computazionale numerico

## Lezione 2

f.poloni@sns.it

2008-10-22

### 1 Vettori e matrici

Octave è pensato per lavorare con vettori e matrici; pertanto, ha una sintassi specifica e parecchi comandi dedicati, che rendono molto più semplice lavorare con i vettori rispetto a un linguaggio non specifico come il C.

#### 1.1 Creare vettori e matrici

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

```
 1  2  3  
 4  5  6
```

```
octave:1> zeros(3,2)  
ans =
```

```
 0  0  
 0  0  
 0  0
```

```
octave:2> ones(3,2)  
ans =
```

```
 1  1  
 1  1  
 1  1
```

```
octave:3> eye(3)  
ans =
```

```
 1  0  0  
 0  1  0  
 0  0  1
```

```
octave:4> rand(2,3)  
ans =
```

```
 0.615284  0.959462  0.755501  
 0.343727  0.098065  0.469369
```

## 1.2 Il range operator :

Con la sintassi `a:t:b` creiamo un vettore (riga) che contiene gli elementi `a`, `a+t`, `a+2t`... fino a `b` (o fino all'ultimo che sia minore o uguale a `b`). Se `t=1`, può essere omesso.

```
octave:6> 1:0.5:4
ans =
    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000

octave:7> 1:10
ans =
     1     2     3     4     5     6     7     8     9    10

octave:8> 1:2:10
ans =
     1     3     5     7     9
```

Bonus question: dove avete già usato l'operatore `:`?

## 1.3 Accedere agli elementi

```
octave:16> A=ones(2,3)
A =
     1     1     1
     1     1     1

octave:17> A(1,2)=2
A =
     1     2     1
     1     1     1

octave:18> A(1,2)
ans = 2
octave:19> A(5,10)
error: invalid row index = 5
error: invalid column index = 10
octave:19> A(5,10)=7
A =
     1     2     1     0     0     0     0     0     0     0
     1     1     1     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     0     0
     0     0     0     0     0     0     0     0     0     7
```

## 1.4 Operazioni su vettori

```
octave:20> a=1:3
a =
```

```

1 2 3
octave:21> b=4:6
b =
4 5 6
octave:22> a+b
ans =
5 7 9
octave:23> sin(a)
ans =
0.84147 0.90930 0.14112
octave:24> 2*a+1
ans =
3 5 7
octave:25> a.*b %operazioni elemento per elemento
ans =
4 10 18
octave:26> c=a' %matrice trasposta
c =
1
2
3
octave:27> a'*b %prodotto matrice-matrice
ans =
4 5 6
8 10 12
12 15 18

```

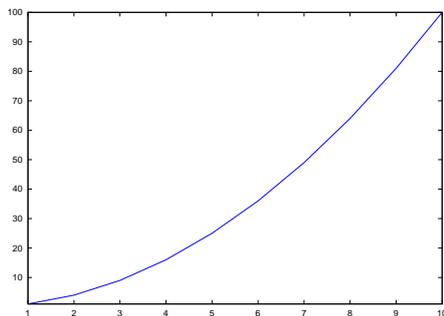
## 2 Grafici

Il comando `plot(x,y)` prende come argomenti due vettori della stessa lunghezza  $x$  e  $y$  e disegna sul piano cartesiano i punti  $x(i),y(i)$  collegandoli con una linea.

```

octave:28> r=1:10
r =
1 2 3 4 5 6 7 8 9 10
octave:30> plot(r,r.^2)

```



Il seguente comando disegna un cerchio.

```
octave:23> r=0:.1:2*pi;
octave:24> plot(cos(r),sin(r))
```

*Esercizio 1.* Scrivere una funzione `circle(z,r)` che, dato un complesso  $z$  e un reale  $r \geq 0$ , disegni il cerchio di centro  $\text{real}(z)$ ,  $\text{imag}(z)$  e raggio  $r$ .

### 3 Cerchi di Gerschgorin

La seguente funzione disegna gli autovalori e i cerchi di Gerschgorin di una matrice quadrata  $A$ .

```
function gg(A)
n=size(A,1);
clearplot; %elimina i disegni precedenti
hold on; %mostra il disegno seguente senza cancellare quello prima
axis("square"); %cerca di non distorcere il disegno
autoval=eig(A);
plot(real(autoval),imag(autoval),".*");
for k=1:n
center=A(k,k);
radius=0; %accumulatore
for j=1:n
if(j~=k)
radius=radius+abs(A(k,j));
endif
endfor
circle(center,radius);
endfor
endfunction
```

*Esercizio 2.* Testare la funzione `gg` su alcune matrici test: `rand(10)`, `randn(10)`, `rosser`, `hilb(10)`,

```
octave:133> A=diag(1:10)+diag(ones(9,1),1)
A =
```

1	1	0	0	0	0	0	0	0	0
0	2	1	0	0	0	0	0	0	0
0	0	3	1	0	0	0	0	0	0
0	0	0	4	1	0	0	0	0	0
0	0	0	0	5	1	0	0	0	0

0	0	0	0	0	6	1	0	0	0
0	0	0	0	0	0	7	1	0	0
0	0	0	0	0	0	0	8	1	0
0	0	0	0	0	0	0	0	9	1
0	0	0	0	0	0	0	0	0	10

*Esercizio 3.* Riuscite a trovare una matrice irriducibile con un autovalore sul bordo dei cerchi (terzo teorema di Gerschgorin)?

*Esercizio 4.* Scrivere una funzione `ggsecond(A)` che illustri come variano gli autovalori quando gli elementi fuori dalla diagonale di  $A$  si spostano verso lo zero, come nella dimostrazione del secondo teorema di Gerschgorin.

#### Soluzione dell'esercizio 4

```
function ggsecond(A)
n=size(A,1);
clearplot;
hold on;
axis("square");
for t=0:.01:1
    autoval=eig((1-t)*diag(diag(A))+t*A); %"help diag" per saperne di piu'
    plot(real(autoval),imag(autoval),"1.");
endfor
for k=1:n
    center=A(k,k);
    radius=0;
    for j=1:n
        if (j~=k)
            radius=radius+abs(A(k,j));
        endif
    endfor
    circle(center,radius);
endfor
endfunction
```

*Se vi stavate annoiando...* Poiché Octave è un linguaggio interpretato, se si riesce a riscrivere le funzioni utilizzando delle operazioni sui vettori invece che dei cicli for, il programma si esegue molto più velocemente. Per esempio, è molto più veloce

```
s=sum(abs(v));
```

rispetto a

```
s=0;
for k=1:size(v)
    s=s+v(k);
endfor
```

Questa operazione si chiama "vectorization". Riuscite a riscrivere i programmi della scorsa lezione vettorizzando i cicli for? Potrebbe esservi utile il manuale di Octave alla sezione 17.4 ("Sum and products").