Risoluzione di sistemi lineari

Il comando \ (*backslash*)

- Se A è una matrice $n \times n$ e $\mathbf{b}$ è un vettore colonna di dimensione n

  `x=A \ b`

  calcola la soluzione del sistema lineare

  $$A\,\mathbf{x} = \mathbf{b}\,.$$

- Non calcola l'inversa.
- Risolve il sistema lineare usando l'algoritmo più efficiente, effetuando dei test preliminari sulla matrice: ad esempio, se A è triangolare inferiore usa il metodo della sostituzione in avanti.

Il comando `inv`.

- Calcola esplicitamente l'inversa della matrice A.
- Non è conveninte usare questo comando per risolvere un sistema lineare perche è più costoso.

Il comando `lu` e il comando `chol`

`[L,U,P]=lu(A)`

- $L\,U = P\,A$
- `L` triangolare inferiore con gli elementi diagonali uguali a 1.
- `U` triangolare superiore.
- `P` matrice di permutazioni.

`[L,U]=lu(A)`

- $L\,U = A$
- `L` non è triangolare inferiore ma una permutazione di righe la rende triangolare inferiore con tutti gli elementi diagonali uguali a 1.
- `U` triangolare superiore.

`R=chol(A)`

- A deve essere simmetrica definita positiva
- $R^T\,R = A$
- `R` triangolare superiore con tutti gli elementi diagonali positivi.

Altri comandi per matrici (e vettori)

- `det(A)` calcola in determinante della matrice A.
- `v=diag(A)` se A è una matrice calcola il vettore con la diagonale di A.
- `D=diag(v)` se $\mathbf{v}$ calcola la matrice diagonale che ha nella diagonale i valori di $\mathbf{v}$.
- `M=tril(A)` restituisce la parte triangolare inferiore di A. (Provare anche `tril(A,1)` o `tril(A,-2)`).
- `M=triu(A)` restituisce la parte triangolare superiore di A. Analogo a `tril`.
- `v=eig(A)` calcola gli autovalori di A.
- `[M,v]=eig(A)` calcola gli autovettori e gli autovalori di A.
- `norm(A)` calcola la norma 2 di A.
- `cond(A)` calcola il condizionamento in norma 2 di A.

Esercizio

Data la matrice

$$\begin{bmatrix} 9 & 14 & -3 \\ 14 & 24 & -6 \\ -3 & -6 & 5 \end{bmatrix}$$

- Verificare che è simmetrica definita positiva.
- Verificare che il metodo di Jacobi non converge.
- Verificare che il metodo di Richardson

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha P^{-1}(\mathbf{b} - A\mathbf{x}^{(k)})$$

con $P = D$ e $\alpha = 0.85$ converge.

Esempio di matrice mal condizionata

La matrice di Hilbert: $a_{i,j} = \frac{1}{i+j-1}$.

```
A=hilb(10)
x=ones(10,1)
b=A*x
x=A \ b
cond(A)
```

<u>Esercizio:</u>
Fare un grafico che rappresenti il numero di condizionamento della matrice di Hilbert in funzione della dimensione.

```
y=[];
for n=2:12
    H=hilb(n);
    c=cond(H)
    y=[y c];
end
semilogy([2:12],y)
```

Operatori logici

&	*and*, congiunzione
$\vert$	*or*, unione
$\sim$	*not*, negazione
$==$	*equal to* , uguaglianza
$\sim=$	*not equal to* , non uguaglianza
$>=$	*greater or equal to* , $\geq$
$<=$	*lower or equal to* , $\leq$

Esercizi

- Scrivere una funzione di Matlab che implementi il metodo iterativo di Gauss-Seidel per l'approssimazione della soluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$. Usare il test d'arresto basato sull'incremento.
- Scrivere una funzione di Matlab che implementi il metodo iterativo del gradiente per l'approssimazione della soluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$.Usare il test d'arresto basato sul residuo.

Gauss-Seidel

```
function [x,nit]=GS(A,b,toll,nmax,x0)
x=x0;
P=tril(A); N=P-A; % A=P-N
nit=0;
inc=toll+1;
while (nit< nmax) & (inc> toll)
    xnew=P\(b+N*x); % P xnew = b + N x
    inc=norm(x-xnew);
    x=xnew;
    nit=nit+1;
end
return
```

Gradiente

```
function [x,nit,flag]=gradiente(A,b)
nmax=1000;
toll=1.e-10;
x=zeros(size(b));
flag=1;
tolleff=toll*norm(b);
res=b-A*x;
for nit=1:nmax
    alpha=(res'*res)/(res'*A*res);
    x=x+alpha*res;
    res=res-alpha*A*res;
    if norm(res)<tolleff, flag=0; return, end;
end
```