CN - lab. lezione 18

Metodo di Jacobi:

```
clear
clc

A = [7 6 9; 4 5 -4; -7 -3 8];
sol = ones(3,1); %soluzione esatta
b = A * sol; %calcolo del vettore b corrispondente
%Ax=b cond x=[1;1;1]

%scomosizione della matrice A
D = diag(diag(A));
C = A-D;

tol = 10^-5;
x_old = [1:3]'; %vettore iniziale x_0 (può essere inizializzato con un vettore
qualunque)
x_new = x_old * 2; %x_new deve essere diverso da x_old in modo da entrare
correttamente nel ciclo

mat_iter = -inv(D)*C;
raggio_spett = max(abs(eig(mat_iter)));

count = 0;
while norm(b - A * x_new) > tol
x_old = x_new;
x_new = mat_iter * x_old + inv(D)*b;
count = count + 1;
end

disp('n iter')
disp(count)
disp('x')
disp(x_new)
```

```
raggio_spett = max(abs(eig(mat_iter)));
```

calcolo del raggio spettrale della matrice di iterazione :

$$M = -D^{-1} \cdot C$$

Serve per capire **se il metodo converge**: se

$$\rho(M) < 1$$

allora converge.

```
while norm(b - A * x_new) > tol
```

si continua finché il residuo $||Ax - b||$ è maggiore della tolleranza.

```
x_new = mat_iter * x_old + inv(D)*b;
```

formula iterativa:

$$x^{(k+1)} = -D^{-1} \cdot Cx^{(k)} + D^{-1}b$$

Metodo Gauss-Seidel

```
x_old = [1:3]';
x_new = x_old .* 2;
[sol, n_iter] = gauss_seidel(A, x_new, b, tol)

function [x, n_iter] = gauss_seidel(A, x0, b, tolleranza)
    %decomposizione A
    DE = tril(A);
    F = triu(A, 1);

    mat_iter = -inv(DE)*F;
    raggio_spett= max(abs(eig(mat_iter)));

    if (raggio_spett > 1)
        disp('il metodo non converge')
        x = [];
        n_iter = 0;
        return
    end

    x = x0;
    n_iter = 0;

    while true
        xNew = mat_iter * x + inv(DE) * b;

    if (norm(xNew-x) < tolleranza)
        x = xNew;
        return
    end

    x = xNew;
    n_iter = n_iter + 1;
    end
end
```

```
x = x0;
n_iter = 0;
```

- `x = x0` imposta il primo vettore di approssimazione della soluzione (quello iniziale passato alla funzione);
- `n_iter = 0` inizializza il contatore delle iterazioni a zero.

```
while true
```

è un ciclo infinito che continuerà fino a quando la condizione di arresto sarà soddisfatta.

```
xNew = mat_iter * x + inv(DE) * b;
```

calcolo della nuova approssimazione `xNew`:

- `mat_iter` è la matrice di iterazione $-DE^{-1}F$
- `inv(DE)*b` è il termine noto trasformato
- implementazione della formula iterativa:

$$x^{(k+1)} = -(DE)^{-1} \cdot F \cdot x^{(k)} + (DE)^{-1} \cdot b$$

```
if (norm(xNew-x) < tolleranza)
    x = xNew;
    return
end
```

controllo della convergenza:

- `norm(xNew - x)` calcola quanto è cambiata l'approssimazione rispetto all'iterazione precedente utilizzando il criterio dell'incremento:

$$||x^{(k+1)} - x^k|| < \epsilon$$

- se questo cambiamento è più piccolo della tolleranza, allora il metodo converge: aggiorna `x` e esce dalla funzione.

```
x = xNew;
n_iter = n_iter +1;
```

preparazione per l'iterazione successiva:

- aggiorna il vettore `x` con il nuovo vettore `xNew`;
- aumenta il numero di iterazioni `n_iter`.
  e il ciclo riparte.