Laboratorio Calcolo Numerico

Esercizio 1 Si scriva una function `jacobi.m` che implementi il metodo di Jacobi per la risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$. La function dovrà avere la seguente intestazione:

```
function [x,k,flag] = jacobi (A,b,x0,toll,kmax)
% JACOBI Metodo di Jacobi per la risoluzione di un sistema lineare
%        con test di arresto sulla norma della differenza di due
%        iterate successive
%
% Uso:
%   [x,k,flag] = jacobi (A,b,x0,toll,kmax)
%
% Dati di ingresso:
%   A    matrice dei coefficienti
%   b    vettore colonna dei termini noti
%   x0   vettore colonna iniziale
%   toll tolleranza per il test di arresto
%   kmax numero massimo di iterazioni
%
% Dati di uscita:
%   x    array che contiene per colonne le iterate (vettori) del metodo
%   k    numero delle iterazioni effettuate
%   flag vale 0 se e' stato possibile costruire la matrice di iterazione
%        vale 1 se la matrice M=D risulta singolare
```

In uscita, la variabile `x` sarà un array (matrice) le cui colonne corrispondono alle successive iterate. Pertanto `x(:,1)` conterrà $\mathbf{x}_0$, `x(:,2)` conterrà $\mathbf{x}_1$, e così via. In caso di convergenza, l'ultima colonna di `x`, estraibile con `x(:,end)`, conterrà la soluzione approssimata. Il test di arresto del ciclo `while` relativo alle iterate deve essere basato sulla norma della differenza tra due iterate successive $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|$.
Il metodo deve essere implementato utilizzando le funzionalità di trattamento delle matrici e dei vettori del Matlab, ovvero si costruisca la matrice di iterazione $B_J = D^{-1}(E + F)$, il vettore $\mathbf{q} = D^{-1}\mathbf{b}$ e si costruiscano le iterate utilizzando la relazione del metodo stazionario $\mathbf{x}_{k+1} = B_J\mathbf{x}_k + \mathbf{q}$. A tal fine serviranno le funzioni Matlab `diag` e `inv`. Nota: come far calcolare in Matlab la matrice $N = E + F$?
Il parametro di uscita `flag` risulta essere uguale ad 1 se la matrice $M = D$ è singolare. All'interno si deve quindi prevedere un controllo su tale eventualità.

Si scriva poi uno uno script `jacobiscript.m` che risolva il sistema seguente utilizzando la function `jacobi`. Si definisca la seguente matrice di ordine $n = 50$

$$A = \begin{pmatrix} 4 & -2 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & -2 \\ & & 1 & 4 \end{pmatrix}$$

con le istruzioni

```
n = 50;
A = diag(4*ones(n,1)) + diag(-2*ones(n-1,1),1) + diag(ones(n-1,1),-1)
```

Si definisca il vettore termine noto $\mathbf{b}$ in modo che la soluzione del sistema sia il vettore unitario $\mathbf{x} = (1, \ldots, 1)^T$, con il comando

```
sol = ones(n,1);
b = A*sol;
```

Si parta dal vettore iniziale nullo e si utilizzi una tolleranza di 10^{-5}.

Ottenuti i risultati si rappresenti su di un grafico a scala logaritmica, per ogni iterata, la norma euclidea dell'errore (calcolabile in questo caso perchè è nota la soluzione esatta!). Si utilizzi il comando `norm` per costruire il vettore da utilizzare nel grafico con un ciclo `for` che, per `j=1:k`, calcoli

```
err(j) = norm(x(:,j)-sol);
```

Esercizio 2

Si scriva, sempre utilizzando le operazioni tra matrici e vettori del Matlab, una function `gaussseidel.m` che implementi il metodo di Gauss-Seidel per la risoluzione di sistemi lineari. Le specifiche saranno simili a quanto indicato per l'Esercizio 1 sul metodo di Jacobi. Per la costruzione della matrice di iterazione B_G serviranno anche i comandi `triu` e `tril`.

Si scriva uno script di utilizzo di tale function e si provi a risolvere lo stesso sistema con tale metodo, producendo un grafico come indicato nell'Esercizio 1.

Esercizio 3

Al fine di confrontare i risultati ottenuti con i due metodi, si scriva uno script che calcoli le soluzioni approssimate ottenute con entrambi i metodi e produca un grafico unico che contenga le norme degli errori calcolati con i due metodi.

Esercizio 4 Prendendo spunto dalla function `gaussseidel.m`, si desidera scrivere una function `SOR.m` che implementi il metodo SOR per la risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$, con il fattore di rilassamento. La function dovrà avere la seguente intestazione:

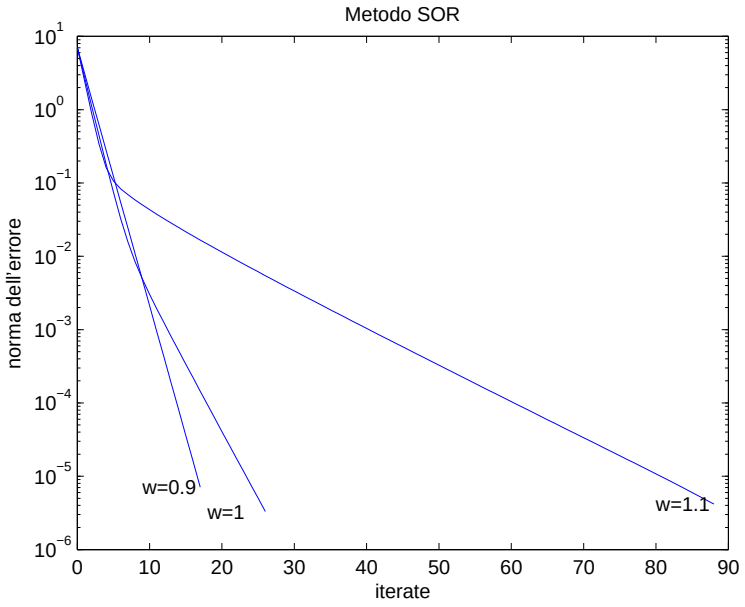
```
function [x,k,flag] = SOR (A,b,w,x0,toll,kmax)
% SOR Metodo SOR per la risoluzione di un sistema lineare
%     con test di arresto sulla norma della differenza di due
%     iterate successive
%
% Uso:
%   [x,k,flag] = SOR (A,b,w,x0,toll,kmax)
%
% Dati di ingresso:
%   A    matrice dei coefficienti
%   b    vettore colonna dei termini noti
%   w    fattore di rilassamento
%   x0   vettore colonna iniziale
%   toll tolleranza per il test di arresto
%   kmax numero massimo di iterazioni
%
% Dati di uscita:
%   x    array che contiene per colonne le iterate (vettori) del metodo
%   k    numero delle iterazioni effettuate
%   flag vale 0 se e' stato possibile costruire la matrice di iterazione
%        vale 1 se la matrice M risulta singolare
```

In uscita, la variabile `x` sarà un array (matrice) le cui colonne corrispondono alle successive iterate. In caso di convergenza, l'ultima colonna di `x` conterrà la soluzione approssimata. Il test di arresto del ciclo `while` relativo alle iterate deve essere basato sulla norma della differenza tra due iterate successive $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|$. Il parametro di uscita `flag` risulta essere uguale ad 1 se la matrice M è singolare.

Si scriva poi uno uno script `SORscript.m` che risolva lo stesso sistema indicato nell'esercizio 1, utilizzando la function `SOR`, e prevedendo di poter inserire durante la sua esecuzione il valore del parametro di rilassamento `w`.
Lo script deve verificare che i valori inseriti appartengano all'intervallo definito dal teorema di Kahan. Si parta dal vettore iniziale nullo e si utilizzi una tolleranza di 10^{-5} ed un `kmax` $= 100$. Si ripeta l'esecuzione dello script più volte e, come parametri `w` si utilizzino per prova i valori $0.9, 1, 1.1$.

Esercizio 5

Si scriva poi uno uno script `SORscriptiter.m` che preveda un ciclo `while` per poter inserire durante la sua esecuzione un certo numero di diversi valori del parametro di rilassamento `w`.
Utilizzando i valori $0.9, 1, 1.1$ la figura risultante dovrebbe essere



Esercizio 6

Si utilizzi lo script dell'Esercizio 5, prendendo la stessa matrice, ma di dimensione $n = 250$ e tolleranza 10^{-15} e si cerchi di individuare (si guardi il grafico ed il numero di iterate) un intervallo $[a, b]$ a cui, presumibilmente, appartenga il valore w_{opt}.

Esercizio 7

Si modifichi la function del metodo SOR per utilizzare come test di arresto il vettore residuo

$$\|\mathbf{r}_k\|/\|\mathbf{b}\| < \texttt{toll}$$

e la si chiami `SORres`. Si modifichi lo script dell'esercizio 2 e si confronti il nuovo grafico con il precedente, ma si ponga nel nuovo script il valore di `toll` uguale a `toll*norm(b)` per rendere consistente il confronto. Sono variate il numero di iterazioni a parità di fattore `w`?