Es:
$$A = \begin{bmatrix} 3 & & 1 \\ 1 & \ddots & \vdots \\ \vdots & & 1 \\ 1 & & 3 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

- $H_J = -\frac{1}{3}\begin{bmatrix} 0 & & 1 \\ 1 & \cdots & \vdots \\ \vdots & & 1 \\ 1 & & 0 \end{bmatrix}, \quad c_J = \frac{1}{3} b$

- $\rho(H_J) = \frac{1}{3}$ (ci si aspetta : $\| e_k \| \sim \left(\frac{1}{3}\right)^k$)

- $$x_{k+1} = H_J x_k + c_J$$
$$= \begin{bmatrix} -\frac{1}{3} x_k(N) + c_J(1) \\ -\frac{x_k(1) + x_k(N)}{3} + c_J(2) \\ \vdots \\ -\frac{x_k(1) + x_k(N)}{3} + c_J(N-1) \\ -\frac{1}{3} x_k(1) + c_J(N) \end{bmatrix}$$
($x_k(j)$ = j-esima componente di $x_k \in \mathbb{R}^N$)

$k = 0, 1, 2, \ldots$

- posto x_* = la soluz di $Ax = b$, $e_k = x_k - x_*$:

$$e_{k+1} = H_J e_k \qquad k = 0, 1, \ldots$$

(1) $$e_1 = H_J e_0 = -\frac{1}{3} \boxed{} e_0 = -\frac{1}{3} \begin{bmatrix} e_0(N) \\ e_0(1) + e_0(N) \\ \vdots \\ e_0(1) + e_0(N) \\ e_0(1) \end{bmatrix}$$

(2) $$e_2 = H_J e_1 = -\frac{1}{3} \boxed{} e_1 = \frac{1}{9} \begin{bmatrix} e_0(1) \\ e_0(1) + e_0(N) \\ \vdots \\ e_0(1) + e_0(N) \\ e_0(N) \end{bmatrix}$$

da cui:

$$\| e_k \| = \frac{1}{3^k} \sqrt{e_0(1)^2 + \left(e_0(1) + e_0(N)\right)^2 + \dots + e_0(N)^2}$$

$$\xrightarrow[k \to \infty]{} 0 \quad \text{come } \left(\frac{1}{3}\right)^k$$

Es (vedere le due pagine seguenti) ...

$$\log_{10} NE(k) = -ck + d$$

$$\Rightarrow \quad NE(k) = 10^{-ck+d} = A\, 10^{-ck}$$

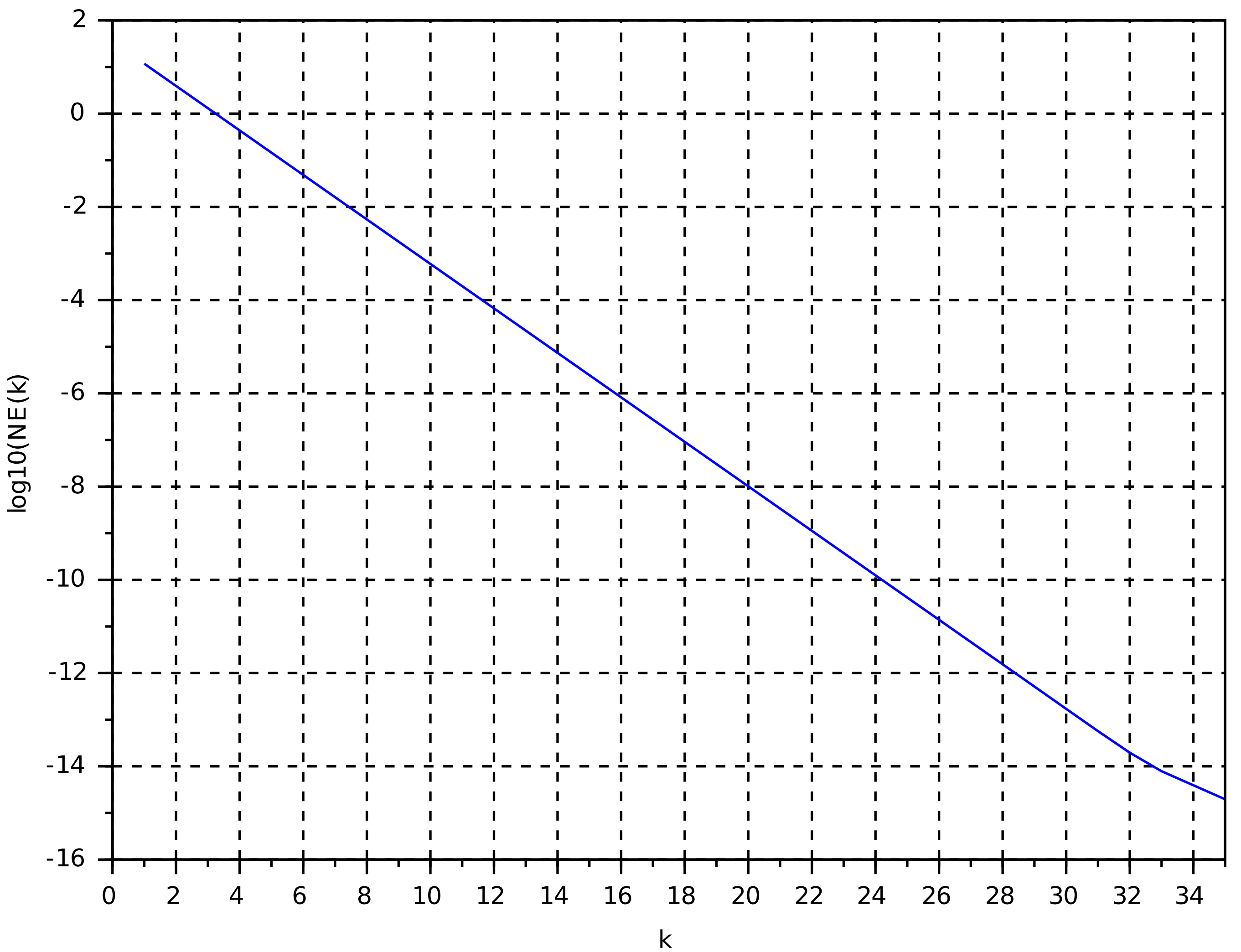
$$= A\, (10^{-c})^k .$$

dalle figure: $\quad c \simeq \frac{1}{2}$,

$$10^{-c} \simeq \frac{1}{\sqrt{10}} \simeq 0{,}31$$

Sistemi con matrice <u>SPARSA</u>: vedere esempi nella parte finale della pagina di help del comando backslash.

```
//
clear;
//
Percorso = "/home/ciampa/Lavoro/Scilab/Lavoro/Sistemi Lineari/";
exec(Percorso + "/Metodi/EGPP.sci");
exec(Percorso + "/Metodi/SA.sci");
exec(Percorso + "/Metodi/SI.sci");
//
T_J = 0;
T_EG = 0;
NE = [];
//
N = 5000;
ripetizioni = 50;
            //  100:   TJ = 8.4d-3   TG = 4.2d-2   EJ = 1.6d-16   EG = 3.3d-17
            //  500:   TJ = 4.0d-2   TG = 2.7
            // 1000:   TJ = 7.9d-2   TG = 18.8
            // 2000:   TJ = 1.5d-1   troppa memoria necessaria per EGPP
            // 3000:   TJ = 2.3d-1   troppa memoria necessaria per EGPP
            // 4000:   troppa memoria per costruire A; senza costruire A:
            //         TJ = 3.1e-1
            // 5000:   senza costruire A:
            //         TJ = 3.9e-1
//
//A = 3*eye(N,N); A(2:N,1) = ones(N-1,1); A(1:N-1,N) = ones(N-1,1);
b = ones(N,1);
//
c = b/3;
//
x_esatta = ones(N,1)/6; x_esatta(1) = 1/4; x_esatta(N) = 1/4;
tic();
for j=1:ripetizioni,
    x = zeros(N,1);
    xn = x;
    for k = 1:35,
        s = -(x(1)+x(N))/3;
        for i = 2:N-1, xn(i) = s + c(i); end;
            xn(1) = c(1) - x(N)/3;
            xn(N) = c(N) - x(1)/3;
            x = xn;
            NE(k) = norm(x-x_esatta);
    end;
end;
T_J = toc()/ripetizioni;
//
for k = find(NE == 0), NE(k) = %eps; end;
printf("\nerrore relativo J = %4.3e\n",norm(x - x_esatta)/norm(x_esatta));
RES = zeros(N,1);
RES(1) = 3*x(1) + x(N) - b(1);
for i = 2:N-1, RES(i) = 3*x(i) + x(1) + x(N) - b(i); end;
RES(N) = x(1) + 3*x(N) - b(N);
printf("\nmisura residuo = %4.3e\n",norm(RES)/norm(b));
printf("\ntempo per Jacobi (N = %d) = %4.3e\n",N, T_J);
//
scf(0);clf();plot([1:35]',log10(NE));xgrid();
//
tic();
//for j=1:ripetizioni,
    [S,D,P,info] = EGPP(A);
    [c,info] = SA(S,P*b,"disabilita");
    [x_EG, info] = SI(D,c,"disabilita");
//end;
T_EG = toc();
//
printf("\nerrore relativo EG = %4.3e\n",norm(x_EG - x_esatta)/norm(x_esatta));
printf("\ntempo per Gauss (N = %d)= %4.3e\n", N, T_EG);
//
```



log10(NE(k))
k
2
0
-2
-4
-6
-8
-10
-12
-14
-16
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34

```
-->tic();
   res = umfpack(A,'\',b);
   mprintf('\nTime with umfpack: %.3f\n',toc());

Time with umfpack: 0.081

-->tic();
   res = linsolve(A,b);
   mprintf('\nTime with linsolve: %.3f\n',toc());

Time with linsolve: 48.661

-->tic();
   res = A\b;
   mprintf('\nTime with backslash: %.3f\n',toc());

Time with backslash: 7.971
```

(22) Si considera il sistema (omogeneo!) $\mathtt{A}x = \mathtt{b}$ con `A` matrice 3562×3562 ad elementi reali, *sparsa*. Quest'ultimo termine si riferisce al *tipo* di struttura di `A` (*matrice sparsa*, appunto). Se M è una matrice (più tecnicamente: un *oggetto di tipo matrice*), Scilab ne memorizza *tutte* le componenti e tutte le manipolazioni di M avvengono in modo "ovvio." Invece, se M è una matrice sparsa (un *oggetto di tipo matrice sparsa*), Scilab memorizza soltanto *la posizione* (riga e colonna) e *il valore* di ciascun elemento *non nullo*, e le manipolazioni di M sono eseguite utilizzando procedure *ad hoc* per matrici sparse. L'uso della struttura matrice sparsa è tanto più vantaggioso (rispetto alla struttura matrice) quanto più elevata è la percentuale di elementi nulli in M. Per ulteriori informazioni sulle matrici sparse in Scilab vedere [4]. La matrice sparsa è creata dall'istruzione `ReadHBSparse`, che legge la matrice dal file specificato. Per dettagli su come la matrice è codificata nel file, vedere [5].

(23) Si determinano approssimazioni della soluzione del sistema con *tre* procedure differenti e si misurano *i tempi* di esecuzione. La prima approssimazione è ottenuta dall'istruzione `umfpack(A,'\',b)`. Come specificato nella descrizione, questa istruzione utilizza una procedura *specifica* per sistemi con matrice sparsa. La seconda approssimazione è ottenuta dall'istruzione `linsolve(A,b)`. Questa istruzione, come specificato nella descrizione, *può* operare su sistemi con matrice sparsa e, in tal caso, utilizza procedure specifiche.[4] Si osservi che la funzione determina *l'intera struttura* dell'insieme delle soluzioni del sistema, un compito *di complessità più grande* rispetto alla sola determinazione di una soluzione. Infine, la terza approssimazione è ottenuta dall'istruzione `A\b`. Nella descrizione precedentemente riportata dell'istruzione, *non si fa cenno* alla possibilità che essa possa operare su matrici

[4]Per approfondire, vedere il codice sorgente della funzione (tecnicamente una *compiled macro*), accessibile con l'istruzione `editor(get_function_path('linsolve'))`, [3], Capitolo 4.

sparse (cosa, invece, esplicitamente menzionata nella descrizione della funzione `linsolve`). Il risultato ottenuto indica come la funzione *sia in grado* di operare su matrici sparse (non viene generato alcun errore a causa del *tipo* di operandi) e, coerentemente con quanto riportato nella Sezione 3.2 di [4], che anch'essa utilizza procedure specifiche per sistemi con matrice sparsa.[5]

(24) I tempi riportati, in *secondi*, corrispondono al tempo di volta in volta trascorso tra l'esecuzione dell'istruzione `tic()` e l'esecuzione dell'istruzione `toc()` (vedere la descrizione delle istruzioni). Si osservi che ripetendo le istruzioni ciascuno dei tempi potrà risultare *diverso*.

Riferimenti

[1] Aceto L. e Ciampa M, *Decomposizione ai valori singolari*, 2009.[6]

[2] Anderson E. ecc, *LAPACK Users' Guide*, Third Edition, 1999.[7]

[3] Baudin M, *Programming in Scilab*, 2011.[8]

[4] Baudin M, *Introduction to sparse matrices in Scilab*, 2011.[9]

[5] Duff I, *User's guide for the Harwell-Boeing Sparse Matrix Collection*, 1992.[10]

[6] Lambers J, *MAT 610: Numerical Linear Algebra*, 2014.[11]

[5] Per l'operatore `backslash` è definita una *overloading function*: `%sp_l_sp` (vedere la descrizione della funzione `overloading` per capire l'origine dello strano *nome* della funzione). Il codice sorgente (accessibile con l'istruzione `editor(get_function_path('%sp_l_sp'))`, conferma quanto espresso nella Sezione 3.2 di [4].

[6] `http://users.dma.unipi.it/ciampa/Didattica_10-11/x-AlgLinFondGeo_Materiale.php`

[7] `http://www.netlib.org/lapack/lug/lapack_lug.html`

[8] `http://forge.scilab.org/index.php/p/docprogscilab/downloads/`

[9] `http://forge.scilab.org/index.php/p/docscisparse/downloads/`

[10] `http://www.cs.colostate.edu/ mroberts/toolbox/c++/sparseMatrix/hbsmc.pdf`

[11] `http://www.math.usm.edu/lambers/mat610/book610.pdf`

Oss: per lo studio della convergenza di un metodo sono spesso disponibili condizioni SUFFICIENTI. Ad es:

Teo: $A \in \mathbb{R}^{n \times n}$ a PDF per righe. Allora il m. it di Jacobi è convergente.

Teo: $A \in \mathbb{R}^{n \times n}$

- a PDF per righe

oppure

- simm def pos

Allora il m. it di Gauss-Seidel è converg

* CRITERI di ARRESTO *

- $Ax = b$, $A \in \mathbb{R}^{n \times n}$ invertibile , $b \neq 0$.

(1) <u>SE</u> $\|r_k\| \leq E \|b\|$ <u>ALLORA</u> STOP

con $r_k = Ax_k - b$ ...

- "calcolabile" e "certam verif dopo un numero FINITO di iterazioni"
- se verificato, allora: $\boxed{\dfrac{\|x_k - x^*\|}{\|x^*\|} \leq c(A) \dfrac{\|r_k\|}{\|b\|}}$

SE $c(A)$ grande (sistema mal condiz)

ALLORA: $\dfrac{\|r_k\|}{\|b\|}$ piccolo $\not\Rightarrow$ err rel piccolo.

(2) <u>se</u> $\|x_k - x_{k-1}\| < E$ <u>allora</u> STOP

- "calcolabile" e "certam verif ..." (perché ?)
- se verif allora:

<u>SE</u> $\|H\| < 1$ <u>ALLORA</u>

$$\|x_k - x^*\| \leq \frac{\|H\|}{1 - \|H\|} \|x_k - x_{k-1}\|$$

SE $\|H\| \simeq 1$ ALLORA: $\|x_k - x_{k-1}\|$ piccolo $\not\Rightarrow$ err assoluto piccolo

* Soluz di un sist NEL SENSO dei M.Q. *

def: $A \in \mathbb{R}^{n \times k}$, $b \in \mathbb{R}^n$

$x^* \in \mathbb{R}^k$ soluz di $Ax = b$ nel senso dei mq

se

$$\forall x \in \mathbb{R}^k, \quad \| Ax - b \|^2 \geq \| Ax^* - b \|$$

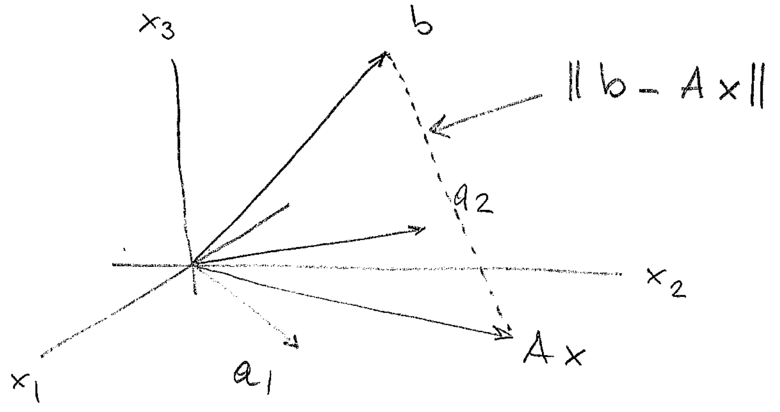
Oss: x^* è uno degli elem che rende minimo il valore della funzione

$$F(x) = \| Ax - b \|^2 : \mathbb{R}^n \to \mathbb{R}$$

Es: $A = \begin{bmatrix} 1 & -1 \\ 1 & 2 \\ 0 & 0 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix}$

- $x \in \mathbb{R}^2$; $Ax = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} x_1 + \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix} x_2$

- tra tutte le $x \in \mathbb{R}^2$ si cercano quelle che rendono minima la norma del vettore $Ax - b$ ~ tra tutte le comb lin delle colonne di A si cercano quelle (i coeff di quelle) che rendono minima la norma del vettore $Ax - b$ ...



- la norma $\|b-Ax\|$ è minima (ovvero $\|Ax-b\|^2$ è minima)

quando Ax è la <u>pr ortogonale di b</u> sul piano delle colonne di A

- $\forall b$, $\exists!$ pr ort di b sullo spazio gen dalle colonne di A <u>MA</u>:

1) <u>SE</u> colonne di A <u>lin indip</u>: $\exists!$ comb lin che vale la pr ort, ovvero: $\exists!$ soluz del sist $Ax=b$ nel senso dei mq

2) <u>SE</u> colonne di A <u>lin dip</u>: $\exists$ infinite comb lin che valgono la pr ort, ovvero: $\exists$ infinite soluz del sist $Ax=b$ nel senso dei mq.