Note Title 2020-04-21

$Ax=b$, A quadrata invertibile

$A=M-N$ M arbitraria invertibile $N=M-A$

$$b=Ax=(M-N)x=Mx-Nx$$

$$Mx=Nx+b \rightarrow x=M^{-1}(Nx+b)=\underbrace{M^{-1}Nx+M^{-1}b}_{f(x)}$$

$x^{(0)}$ arbitrario

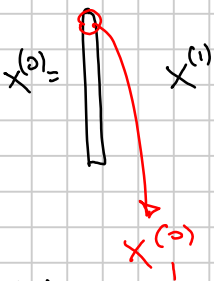
$$x^{(1)}=M^{-1}Nx^{(0)}+M^{-1}b$$
$$x^{(2)}=M^{-1}Nx^{(1)}+M^{-1}b$$
$$\vdots$$
$$x^{(k+1)}=\underbrace{M^{-1}N}_{P}x^{(k)}+\underbrace{M^{-1}b}_{q}$$

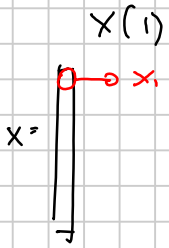
Qualche volta, $x^{(k)} \rightarrow x$ quando $k \rightarrow \infty$

Quando questo succede, il limite risolve il sis. lineare $Ax=b$.

Esempio: $M=I$ $N=I-A$ $M^{-1}=I$

$$x=(I-A)x+b \qquad x^{(k+1)}=(I-A)x^{(k)}+b$$





In Matlab, $x=(I-A)x+b$ sovrascrive la variabile x (che inizialmente valeva $x^{(k)}$) con $x^{(k+1)}$

1) k iterazioni:

```
for i=1:K
    x=Nx+b
end
```

2) fino a condizione di arresto:

```
while norm(A*x-b)>eps
    x=Nx+b
end
```

Δ non dice quante iterazioni faccio

Cosa succede in un esempio in cui il metodo non converge (diverge a $+\infty$)?

$x^{(1)}, x^{(2)}, \dots$ $\quad x^{(15)}, \dots$ $\quad x^{(20.000)}$ $\quad x^{(23278)}$

↑ diventano sempre più grandi

$\uparrow$ SS 10^{307} $\quad \begin{bmatrix} -Inf \\ * \\ * \end{bmatrix}$

$$Ax^{(23278)} = \begin{bmatrix} -Inf \\ NaN \\ -Inf \end{bmatrix}$$

$norm(Ax^{(23278)} - b) \geq eps$ restituisce falso.

oppure

$norm(Ax-b) \geq eps$ è sempre vera $\Rightarrow$ loop infinito (ciclo)

Per uscire: ctrl + C

3) si ferma se $\|Ax-b\| \leq \varepsilon$ oppure dopo k passi

nello while, continuiamo se $\|Ax-b\| \geq \varepsilon$ e iter $\leq k$

```
while norm(..) && iter <= k
    ;
end
```

oppure:

```
for i = 1 .. k
    x = Nx + b
    if norm(..)
        break
    end
end
```

⚠ Posso avere loop infinito con metodo 2 se la tolleranza richiesta non si può raggiungere causa errori nelle operazioni di macchina.

Costi: metodo 1: (Sia una matrice $A \in \mathbb{R}^{n \times n}$)

$N = \text{eye}(n) - A$ ← n^2 operazioni

for $i = 1 : K$

$x = N * x + b$ ← $2n^2 + O(n)$ (prodotto matrice-vettore + somma vettore)

end

Costo: $2n^2 + O(n)$ per passo

Metodo 2:

$N = I - A$ ← n^2 op.

while norm$(A*x - b, 1) \geq$ eps

$x = N * x + b$ ← $2n^2 + O(n)$ op.

end

La riga while( ) costa: $2n^2 + O(n)$ per il prodotto

n per la differenza

$\text{norm}(v, 1) = |v_1| + |v_2| + \dots + |v_n|$ $n + O(n)$ addizioni

Costo: $4n^2 + O(n)$ per passo

Ci sarebbe un modo di evitare uno di questi due prodotti:

Ad ogni passo, stiamo calcolando prima Ax

e poi $Nx = (I - A)x = x - Ax$

Salviamo in una variabile $y = Ax$

metodo2veloce

$x = x0$

$y = A * x$

while norm$(y - b, 1) <$ eps

$x = (x - y) + b$

$y = A * x$

end

Se la matrice A (e la matrice N) la moltiplico,

posso avere metodi più furbi di fare il prodotto

es.

$$N = \begin{bmatrix} 0 & 0 & v_1 \\ 0 & 0 & v_2 \\ 0 & 0 & \vdots \\ & & v_n \end{bmatrix}$$

$N * x$ ci costa sempre $2n^2 + O(n)$

$$N * x = \begin{bmatrix} 0 & 0 & v_1 \\ 0 & 0 & v_2 \\ & & \vdots \\ & & v_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} v_1 x_n \\ v_2 x_n \\ v_3 x_n \\ \vdots \\ v_n x_n \end{bmatrix}$$

↳ per questa matrice particolare, so calcolare le entrate di Nx in $O(n)$ operazioni

---

Dalla volta scorsa:

fattorizz. LU di una matrice della forma

$$A = \begin{bmatrix} 1 & & & & v_1 \\ & 1 & & & v_2 \\ & & 1 & & \vdots \\ & & & \ddots & v_{n-1} \\ v_1 & v_2 & \cdots & v_{n-1} & 1 \end{bmatrix}$$

+ sol. sist. lineare

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ -v_1 & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & v_1 \\ & 1 & & v_2 \\ & & \ddots & \vdots \\ & & 1 & v_{n-1} \\ v_1 & \cdots & v_{n-1} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & v_1 \\ & 1 & & v_2 \\ & & \ddots & \vdots \\ & & 1 & v_{n-1} \\ 0 & v_2 & v_3 \cdots & v_n \; 1 - v_1^2 \end{bmatrix}$$

$-v_1 \cdot$ I r.g.

$L \quad A \quad = \quad A_1$

$$\begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ -\frac{\gamma_1}{\alpha_1} & & & \end{bmatrix} \begin{bmatrix} \alpha_1 & & & \beta_1 \\ & \alpha_2 & & \beta_2 \\ & & \ddots & \vdots \\ & & & \beta_{n-1} \\ \gamma_1 & \gamma_2 & \cdots & \gamma_{n-1} \; \alpha_n \end{bmatrix} = \begin{bmatrix} \alpha_1 & & & \beta_1 \\ & \ddots & & \\ & & & \alpha_{n-1} \; \beta_{n-1} \\ 0 & \gamma_2 & \cdots & \gamma_{n-1} \; \alpha_n - \frac{\beta_1 \gamma_1}{\alpha_1} \end{bmatrix} = \delta_1$$

L_1 $\qquad A \qquad\qquad A_1$

$$\begin{bmatrix} 1 & & & \\ & \ddots & & \\ 0 & -\frac{\gamma_2}{\alpha_2} & 0 \cdots 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 & \beta_1 \\ \gamma_2 & \alpha_2 & \beta_2 \\ & \ddots & \ddots & \vdots \\ 0 & \gamma_2 & \cdots & \alpha_{n-1} & \beta_{n-1} \\ & & & \gamma_{n-1} & \delta_{n-1} \end{bmatrix} = \begin{bmatrix} \alpha_1 & \beta_1 \\ & \alpha_2 & \ddots & \vdots \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ 0 & 0 & \gamma_3 \gamma_4 \cdots & \gamma_{n-1} & \delta_2 \end{bmatrix}$$

(n-esima riga − $\frac{\gamma_2}{\alpha_2}$ volte 1ª riga)

$\delta_2 = \delta_1 - \frac{\gamma_2}{\alpha_2}\beta_2$

L_2 A_1 $=$ A_2

Ad ogni passo si ripete la stessa situazione, $\delta_k = \delta_{k-1} - \frac{\gamma_k \beta_k}{\alpha_k}$ (dove $\frac{\gamma_k}{\alpha_k} = \varepsilon_k$)

$$L_k = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & -\frac{\gamma_k}{\alpha_k} & & \end{bmatrix}$$

↑ k-esimo elemento

Alla fine, dopo $n-1$ passi, otteniamo

$$U = \begin{bmatrix} \alpha_1 & \beta_1 \\ & \alpha_2 & \ddots & \vdots \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ 0 & & & & \delta_{n-1} \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ \frac{\gamma_1}{\alpha_1} & \frac{\gamma_2}{\alpha_2} & \cdots & \frac{\gamma_{n-1}}{\alpha_{n-1}} & 1 \end{bmatrix} = L_1^{-1} L_2^{-1} L_3^{-1} \cdots L_{n-1}^{-1}$$

ε_1 ε_2 $\cdots$ ε_{n-1}

```
epsilon(1) = gamma(1)/alpha(1);
delta(1) = alpha(n) - epsilon(1)*beta(1);
for k = 2:n-1
    epsilon(k) = gamma(k)/alpha(k);
    delta(k) = delta(k-1) - epsilon(k)*beta(k);
end
```

$$L = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ \varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_{n-1}\ 1 \end{bmatrix} \qquad U = \begin{bmatrix} \alpha_1 & & & \beta_1 \\ & \alpha_2 & & \beta_2 \\ & & \ddots & \vdots \\ & & & \alpha_n\ \beta_{n-1} \\ & & & \delta_{n-1} \end{bmatrix}$$

Se posso partire da

$$\begin{bmatrix} \alpha_1 & \beta_1 & \cdots & \beta_{n-1} \\ \gamma_1 & \alpha_2 & & \\ \gamma_2 & & \ddots & \\ \vdots & & & \\ \gamma_{n-1} & & & \alpha_n \end{bmatrix},$$

già dopo il primo passo, la matrice è piena $\Rightarrow$ mi servono n^2 elementi per memorizzarla