Calcolo Scientifico - laurea triennale Informatica
SOLUZIONE NUMERICA DI SISTEMI LINEARI

prof.ssa Rossana Vermiglio, dott. Dimitri Breda
Dipartimento di Matematica e Informatica
Università degli Studi di Udine
vermiglio,dbreda@dimi.uniud.it
http://www.dimi.uniud.it/∼rossana,dbreda

Indice

1 Soluzione del sistema lineare e condizionamento

Il problema che vogliamo affrontare riguarda la soluzione di un sistema lineare

$$Ax = b, \tag{1}$$

dove $A \in \mathbb{R}^{n\times n}$ è una matrice quadrata non singolare, $b \in \mathbb{R}^n$. La condizione che A sia non singolare ci assicura che il problema sia ben posto: esiste ed è unico il vettore $x \in \mathbb{R}^n$ soluzione del sistema (1).

Sono equivalenti

- A non singolare
- $\det(A)$ non nullo
- $\text{rango}(A) = n$
- esiste l'inversa A^{-1}

Per studiare il condizionamento del problema, introduciamo dapprima solamente una perturbazione $\delta b \in \mathbb{R}^n$ del termine noto b. Sia $x + \delta x \in \mathbb{R}^n$ la soluzione esatta del problema perturbato, i.e.

$$A(x + \delta x) = b + \delta b.$$

b δb
$Ax + A\delta x$
↳ ricavo $\delta x = A^{-1} \cdot \delta b$

Vale $\delta x = A^{-1}\delta b$, da cui si ottiene $\|\delta x\| \leq \|A^{-1}\| \cdot \|\delta b\|$. Definendo l'errore relativo $\epsilon_x = \frac{\|\delta x\|}{\|x\|}$ e $\epsilon_b = \frac{\|\delta b\|}{\|b\|}$, si ottiene

$$\epsilon_x \leq \text{cond}(A)\epsilon_b \tag{2}$$

dove

$$\text{cond}(A) = \|A\|\|A^{-1}\|$$

↳ di quanto cambia la soluzione se perturbo il termine noto?

è il **numero di condizionamento** e misura la sensibilità del problema a perturbazioni introdotte nel termine noto. Introduciamo ora anche una perturbazione $\delta A \in \mathbb{R}^{n\times n}$ nella matrice A ed indichiamo $x + \delta x \in \mathbb{R}^n$ la soluzione esatta del problema così perturbato (che supponiamo esistere), i.e.

↳ di quanto cambia la soluzione se perturbo il termine noto e la MATRICE A?

$$(A + \delta A)(x + \delta x) = b + \delta b.$$

Vale

$$\epsilon_x \leq \frac{\text{cond}(A)}{1 - \text{cond}(A)\epsilon_A}(\epsilon_b + \epsilon_A), \tag{3}$$

dove $\epsilon_A = \frac{\|\delta A\|}{\|A\|}$.

Il problema sarà mal condizionato, se $\text{cond}(A) \gg 1$, mentre sarà ben condizionato se $\text{cond}(A)$ è piccolo, cioè dell'ordine dell'unità. In quest'ultimo caso a piccoli errori nei dati corrisponderanno errori dello stesso ordine di grandezza nella soluzione.

2 Casi semplici: sistema diagonale e triangolare

Come primo passo verso la soluzione numerica del sistema lineare (1), cerchiamo di analizzare e risolvere dei "casi semplici".

2.1 Sistema diagonale

Se $A = \text{diag}(a_{i,i})_{i=1,\dots,n}$, con $a_{i,i} \neq 0$, allora la soluzione x di (1) si ottiene da

$$x_i = \frac{b_i}{a_{i,i}}, \quad i = 1, \dots, n, \tag{4}$$

e richiede n operazioni.

2.2 Sistema triangolare

Supponiamo che A sia una matrice triangolare inferiore (i.e. $a_{i,j} = 0$, $i = 1, \dots, n-1$, $j = i+1, \dots, n$.). La soluzione x di (1) si può calcolare con l'algoritmo di **sostituzione in avanti**:

$$x_1 = \frac{b_1}{a_{1,1}}$$

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} x_j}{a_{i,i}}, \quad i = 2, \dots, n. \tag{6}$$

Per valutare la sua complessità computazionale osserviamo che il calcolo di x_i richiede $i-1$ somme, $i-1$ moltiplicazioni ed una divisione. Segue pertanto che in totale sono effettuate

$$\sum_{i=1}^{n} 1 + 2\sum_{i=2}^{n}(i-1) = o(n^2)$$

operazioni.

Esercizio 1 *Descrivi ed analizza l'algoritmo di* **sostituzione all'indietro** *per la risoluzione di un sistema triangolare superiore, i.e. A è tale che $a_{i,j} = 0$, $i = 2, \dots, n$, $j = 1, \dots, i-1$.*

3 Fattorizzazione LU ed algoritmo di eliminazione di Gauss

3.1 L'algoritmo

Supponiamo ora che $A \in \mathbb{R}^{n \times n}$ sia una matrice quadrata con $\det(A) \neq 0$. Come possiamo ricondurci alla soluzione di sistemi triangolari che sappiamo risolvere bene? Supponiamo di poter fattorizzare la matrice nella seguente forma

$$A = LU \tag{7}$$

dove L, U sono, rispettivamente, una matrice triangolare inferiore con gli elementi diagonali uguali a uno (i.e. $l_{i,i} = 1$, $i = 1, \ldots, n$) e una matrice triangolare superiore. Tale matrici risultano essere invertibili. Nota (7), la risoluzione del sistema lineare (1) può essere ricondotta alla risoluzione di due sistemi triangolari

$$\begin{cases} Lz = b \\ Ux = z \end{cases} \tag{8}$$

Dobbiamo ora affrontare il problema dell'esistenza della fattorizzazione (7) e proporre ed analizzare un algoritmo per il calcolo di tali matrici. Un approccio diretto al problema consiste nel definire le relazioni che gli elementi $l_{i,j}$ e $u_{i,j}$ rispettivemente di L e U, devono verificare affinchè risulti (7) ed è noto in letteratura con il nome **tecnica compatta**.

NOTA 1 *Consideriamo il caso $n = 2$. Dalla relazione segue*

$$\begin{cases} a_{1,1} = l_{1,1}u_{1,1} \\ a_{1,2} = l_{1,1}u_{1,2} \\ a_{2,1} = l_{2,1}u_{1,1} \\ a_{2,2} = l_{2,1}u_{1,2} + l_{2,2}u_{2,2} \end{cases} . \tag{9}$$

Risolvendo le quattro equazioni rispetto le quattro incognite $l_{2,1}$, $u_{1,1}$, $u_{1,2}$, $u_{2,2}$ si ottengono i seguenti valori $u_{1,1} = a_{1,1}$, $u_{1,2} = a_{1,2}$, $l_{2,1} = a_{2,1}/a_{1,1}$, $u_{2,2} = a_{2,2} - a_{2,1} * a_{1,2}/a_{1,1}$.

Ci proponiamo di seguire un approccio altrenativo, che porterà alla definizione dell'**algoritmo di eliminazione di Gauss**.

L'idea base consiste nel ridurre la matrice A alla forma triangolare superiore attraverso un numero finito di trasformazioni, descritte da matrici triangolari inferiori facilmente invertibili. Tali matrici di trasformazione devono annullare (**eliminare**) gli elementi diversi da zero nella parte triangolare inferiore.

Dato un generico vettore $v \in \mathbb{R}^n$ con $v_k \neq 0$, possiamo annullare tutte le sue componenti dalla $(k+1)$-esima in poi mediante la trasformazione $G_k v$ dove

$$G_k = \begin{pmatrix} 1 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 1 & 0 & \ldots & 0 \\ 0 & \ldots & -\frac{v_{k+1}}{v_k} & 1 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & -\frac{v_n}{v_k} & 0 & \ldots & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_k \\ v_{k+1} \\ \vdots \\ v_n \end{pmatrix} \leftarrow v$$

e

$$G_k v = \begin{pmatrix} v_1 \\ \vdots \\ v_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} . \tag{11}$$

↳ questa parte, dopo il prodotto con G_k, si azzera

La matrice (10) si chiama **matrice elementare di Gauss**. È una matrice triangolare inferiore, facilmente invertibile. La sua inversa è infatti definita da

$$G_k^{-1} = \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & \frac{v_{k+1}}{v_k} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \frac{v_n}{v_k} & 0 & \dots & 1 \end{pmatrix}. \tag{12}$$

Osserviamo inoltre che la trasformazione $t = G_k w$ di un generico vettore $w \in \mathbb{R}^n$ è descritta dalle relazioni

parte che non viene toccata — valori modificati

$$t_i = w_i,\ i = 1, \dots, k, \quad t_i = w_i - \frac{v_i * w_k}{v_k},\ i = k+1, \dots, n. \tag{13}$$

L'**algoritmo di eliminazione di Gauss** lavora sulle colonne della matrice A eliminando passo dopo passo tutti gli elementi nella parte triangolare inferiore. Posta $A^0 = A$, la trasformazione $A^k = G_k A^{k-1}$ al k-esimo passo, $k = 1, 2, \dots, n-1$, è descritta dalla matrice elementare

$$G_k = \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & -l_{k+1,k} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -l_{n,k} & 0 & \dots & 1 \end{pmatrix}, \tag{14}$$

dove $l_{i,k} = \frac{a_{i,k}^{k-1}}{a_{k,k}^{k-1}}, i = k+1, \dots, n$. Nota che G_k risulta definita se l'elemento pivotale $a_{k,k}^{k-1} \neq 0$, $k = 1, \dots, n-1$. In forma matriciale si ottiene

$$U = A^{n-1} = (G_{n-1} \cdots G_2 G_1) A = GA$$

dove $G = (G_{n-1} \cdots G_2 G_1)$ risulta essere una matrice triangolare inferiore con uno sulla diagonale principale. La sua inversa $G^{-1} = (G_1^{-1} G_2^{-1} \cdots G_{n-1}^{-1})$ è ancora una matrice triangolare inferiore con uno sulla diagonale ed è facilmente calcolabile usando (12) ed osservando che il prodotto di tali matrici è essenzialmente la loro "unione". Ponendo pertanto $L = G^{-1}$ otteniamo la fattorizzazione

$$LU = A.$$

L'uso delle matrici elementari di Gauss è proposto per fornire una descrizione formale del processo di fattorizzazione e **non** sono implementate esplicitamente. Utilizzando (13) si ottiene la seguente procedura per l'algoritmo di eliminazione

$$v = \begin{pmatrix} 2 \\ -3 \\ 4 \\ -5 \end{pmatrix}$$

$$G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{3}{2} & 1 & 0 & 0 \\ -\frac{4}{2} & 0 & 1 & 0 \\ \frac{5}{2} & 0 & 0 & 1 \end{pmatrix} \quad G_1 v = \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

PIVOT = 2

di Gauss :

$$
\begin{aligned}
&k = 1, \ldots, n-1,\ i = k+1, \ldots, n,\\
&l_{i,k} = a_{i,k}^{k-1} / a_{k,k}^{k-1},\\
&a_{i,j}^{k} = a_{i,j}^{k-1} - l_{i,k} a_{k,j}^{k-1},\ j = k+1, \ldots, n.
\end{aligned}
$$

elementi della matrice Gaussiana ← $l_{i,k}$; $a_{k,k}^{k-1}$ → PIVOT

Gli elementi $l_{i,j}$, $i = 2, .., n$, $j = 1, .., i-1$, e $l_{i,i} = 1$, $i = 1, \ldots, n$, definiscono la matrice L mentre $U = (a_{i,j}^{n-1})$, $i = 1, \ldots, n$, $j = i, \ldots, n$.

L'algoritmo di eliminazione di Gauss e la fattorizzazione LU sono semplicemente due modi di esprimere lo stesso procedimento.

Per poter applicare tale algoritmo gli **elementi pivotali** $a_{k,k}^{k-1}$ devono essere non nulli ad ogni passo e solo a queste condizioni possiamo concludere che la fattorizzazione LU esiste. Purtroppo non è sufficiente la non singolarità della matrice, vale infatti il seguente risultato:

Teorema 1 *Data una matrice $A \in \mathbb{R}^{n \times n}$, la fattorizzazione LU esiste se le sottomatirci A_k principali di testa di A di ordine k, $k = 1, 2, \ldots, n$, sono non singolari. Tale fattorizzazione è unica.*

Le matrici **simmetriche definite positive** e le **matrici a dominanza diagonale stretta** verificano le ipotesi di tale teorema.

Il caso della matrici simmetriche definite positive sarà trattato ampiamente nel paragrafo dedicato all'algoritmo di Cholesky.

NOTA 2 *Applicando le trasformazioni elementari non solo alla matrice A ma anche al termine noto $= b$, i.e.*

$$z = b^{n-1} = (G_{n-1} \cdots G_2 G_1) b = Gb, \quad \text{e } G = L^{-1} \Rightarrow b = Lz$$

si ottiene la soluzione del sistema triangolare $Lz = b$. L'ultimo passo per calcolare la soluzione x del sistema (1) consiste nel risolvere il sistema $Ux = z$. In pratica applicando l'algoritmo di Gauss ad A e b, abbiamo trasformato il sistema iniziale $Ax = b$ in un sistema equivalente (i.e. con la stessa soluzione) $Ux = z$ con matrice triangolare superiore. Osserviamo infine che la procedura viene così modificata

$$
\begin{aligned}
&k = 1, \ldots, n-1,\ i = k+1, \ldots, n,\\
&l_{i,k} = a_{i,k}^{k-1} / a_{k,k}^{k-1},\\
&\boxed{b_i^k = b_i^{k-1} - l_{i,k} b_k^{k-1},} \quad \text{→ trasformaz. di } b\\
&a_{i,j}^{k} = a_{i,j}^{k-1} - l_{i,k} a_{k,j}^{k-1},\ j = k+1, \ldots, n.
\end{aligned}
$$

La complessità computazionale dell'algoritmo di Gauss è pari a circa $\frac{2n^3}{3}$ operazioni addittive e moltiplicative.

3.2 Applicabilità e stabilità dell'algoritmo

L'algoritmo di eliminazione di Gauss nella versione descritta nel precedente paragrafo si ferma ad un certo passo k se l'elemento pivotale $a_{k,k}^{k-1}$ è nullo. Questo problema può essere risolto con uno scambio di righe: la riga k-esima viene permutata con una riga successiva della matrice che ha in colonna k-esima un elemento non nullo. Ciò è possibile perchè se $a_{i,k}^{k-1} = 0$, $i = k, \ldots, n$, la matrice A risulterebbe singolare, contro la nostra ipotesi.

In linea di principio qualsiasi elemento $a_{i,k}^{k-1}$, $i = k+1, .., n$, non nullo potrebbe essere portato in posizione pivotale. In pratica per migliorare la stabilità dell'algoritmo, evitando la crescita degli errori di arrotondamento, le righe vengono scambiate in modo da portare in posizione pivotale l'elemento $a_{i_k,k}^{k-1}$ tale che

$$|a_{i_k,k}^{k-1}| = \max_{i=k,\ldots,n} |a_{i,k}^{k-1}|. \quad (15)$$

→ max in val. ass. della colonna (a partire dal pivot)

Tale strategia è chiamata **pivot parziale**.

Con questa tecnica risulta $|l_{i,k}| \leq 1$, $i = k, \ldots, n$, $k = 1, .., n-1$, e $\max_{i,j} |u_{i,j}| = \max_{i,j} |a_{i,j}^{n-1}| \leq 2^{n-1} \max_{i,j} |a_{i,j}|$.

La stabiltà dell'algoritmo viene stuidiata con l'analisi dell'errore all'indietro. La soluzione effettivamente calcolata $\tilde{x}$ con l'algoritmo di eliminazione di Gauss in aritmetica di macchina con dati A, b i cui elementi sono numeri di macchina, risulta essere la soluzione esatta di un problema perturbato

$$(A + E)\tilde{x} = b, \quad (16)$$

dove per la matrice E vale la limitazione

$$|E| \leq 4nu(|A| + |\tilde{L}||\tilde{U}|) + O(u^2)$$

e $\tilde{L}, \tilde{U}$ sono le matrici della fattorizzazione calcolate in artimetica di macchina.

NOTA 3 *Ricorda che* $|A| = (|a_{i,j}|)$ *e la disequazione va letta elemento per elemento.*

Tale analisi conferma che la strategia del pivot parziale migliora le proprietà di stabilità dell'algoritmo perchè permette di controllare il modulo degli elementi delle matrici L ed U ed è pertanto essenziale per un'implementazione numericamente stabile dell'algoritmo di Gauss.

In forma matriciale può essere descritta da

matrice di PERMUTAZIONE delle righe che mi porta il pivot cmax (della colonna 1) in prima posiz.

$$U = A^{n-1} = (G_{n-1}P_{n-1} \cdots G_2P_2G_1P_1)A = GA$$

dove P_k sono opportume matrici di permutazione che scambiano la riga k-esima con la riga i_k per cui vale (15). La matrice $G = G_{n-1}P_{n-1} \cdots G_2P_2G_1P_1$ in questo caso non risulta triangolare. Si può facilmente verificare che vale

$$PA = LU \quad (17)$$

con L triangolare inferiore con uno sulla diagonale e $P = P_{n-1} \cdots P_2 P_1$. Quindi il metodo di eliminazione di Gauss con la variente del pivot parziale mi fornisce una fattorizzione LU di un'opportuna permutazione della matrice A nella solo ipotesi di A non singolare. La risoluzione del sistema lineare $Ax = b$ può essere ora ricondotta alla risoluzione di due sistemi triangolari

$$\begin{cases} Lz = Pb \\ Ux = z \end{cases} . \tag{18}$$

NOTA 4 *Applicando le trasformazioni elementari e le permutazioni di righe non solo alla matrice A ma anche al termine noto b, i.e.*

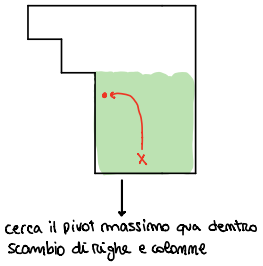
$$z = b^{n-1} = (G_{n-1}P_{n-1} \cdots G_2 P_2 G_1 P 1)b = Gb,$$

si ottiene la soluzione del sistema triangolare $Lz = Pb$. L'ultimo passo per calcolare la soluzione x del sistema originario consiste nel risolvere il sistema $Ux = z$.

Il metodo di eliminazione di Gauss può essere usato per calcolare il determinante di una matrice. Da $PA = LU$ si ricava $\underbrace{\det(P)}_{\pm 1} \det(A) = \underbrace{\det(L)}_{=1} \det(U) = \prod_{k=1}^{n} u_{k,k}$ e quindi, poichè $\det(P) = \pm 1$, il determinante di A è dato dal prodotto degli elementi principali di U a meno del segno. $\Rightarrow$ $\det(A) = \det(U)$

NOTA 5 *Il metodo di eliminazione di Gauss può essere implementato anche nella variante del* **pivot totale***: ad ogni passo viene portato in posizione pivotale l'elemento a_{i_k,j_k}^{k-1} tale che*

$$|a_{i_k,j_k}^{k-1}| = \max_{i,j=k,\ldots,n} |a_{i,j}^{k-1}|. \tag{19}$$



Tale strategia richiede non solo uno scambio di righe ma anche di colonne e porta alla seguente fattorizzazione

$$PAQ = LU, \tag{20}$$

con P e Q sono opportune matrici di permutazione. La risoluzione del sistema lineare $Ax = b$ può essere ora ricondotta alla risoluzione di due sistemi triangolari e ad riordinamento delle varie componenti descritto dalla matrice Q:

$$\begin{cases} Lz = Pb \\ Uy = z \\ x = Qy \end{cases} . \tag{21}$$

La stabilità numerica dell'algoritmo con il pivot totale è teoricamente migliore di quella del pivot parziale. Ma la sua implementazione richiede $(n-k+1)^2$, $k = 1, 2, \ldots, n-1$, confronti contro gli $n-k$, $k = 1, 2, \ldots, n-1$, del pivot parziale ed è quindi più costosa. In pratica la proprietà di stabilità numerica dell'algoritmo con il pivot parziale è più che adeguata per la risoluzione di un sistema lineare con l'algoritmo di eliminazione di Gauss.

NOTA 6 *Se la matrice A del sistema è singolare il metodo di Gauss con il pivot parziale si può ancora applicare. Se risulta $a_{i,k}^{k-1} = 0$, $i = k, \ldots, n$, al k-esimo passo non c'è nessuna operazione da compiere, i.e. $G_k = I$, $P_k = I$, e si presegue con la colonna successiva. La matrice $U = A^{n-1}$ ha l'elemento $u_{k,k}$ nullo e risulta singolare. Ma le ipotesi di consistenza del sistema ci permettono ugualmente di determinare una soluzione con l'algoritmo di sostituzione all'indietro.*

3.3 Aspetti implementativi

Qualora non sia necessario mantenere le informazioni sulla matrice A, nell'implementare al calcolatore il metodo di eliminazione di Gauss, l'area di memoria riservata per la matrice A può essere utilizzata per memorizzare la matrice L nella parte strettamente triangolare inferiore e la matrice U nella parte triangolare superiore.

Nel caso della variante del pivot parziale, non è necessario eseguire effettivamente gli scambi di riga. La posizione della i-esima riga può essere individuata utilizzando un vettore $riga$ che inizialmente ha componenti $riga_i = i$, $i = 1, \ldots, n$. Quando due righe k e ℓ vengono scambiate, esse rimangono nella locazione originaria ed il vettore $riga$ tiene conto dell'ordine, scambiando fra loro $riga(\ell)$ e $riga(k)$.

4 Algoritmo di Cholesky

Nel caso in cui $A \in \mathbb{R}^{n\times n}$ sia simmetrica e definita positiva, i.e. $A^T = A$ e $x^T Ax > 0$ per ogni $x \in \mathbb{R}^n \setminus \{0\}$ allora si può trovare una fattorizzazione speciale di A

$$A = HH^T \tag{22}$$

con H triangolare inferiore con elementi principali positivi. Gli elementi di H possono essere ottenuti con l'algoritmo di Cholesky descritto dall'algoritmo (per colonne):

$$\begin{aligned} &j = 1, \ldots, n, \\ &h_{j,j} = \sqrt{a_{j,j} - \sum_{k=1}^{j-1} h_{j,k} h_{j,k}}, \quad \longrightarrow \text{diagonale} \\ &h_{i,j} = \frac{1}{h_{j,j}} (a_{i,j} - \sum_{k=1}^{j-1} h_{i,k} h_{j,k}), \ i = j+1, \ldots, n. \end{aligned}$$

L'algoritmo di Cholesky è sempre numericamente stabile e non sono necessari scambi di riga. La sua complessità computazionale è circa $O(\frac{n^3}{6})$ ed n estrazioni di radice.

5 Algoritmo di Gauss-Jordan

L'algortimo di Gauss-Jordan vuole portare la matrice A in n passi alla forma diagonale. Il passo k-esimo, $A^k = J_k A^{k-1}$, $k = 1, \dots, n$, è descritto dalla matrice

$$J_k = \begin{pmatrix} 1 & \dots & -l_{1,k} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -l_{k-1,k} & 0 & \dots & 0 \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & -l_{k+1,k} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -l_{n,k} & 0 & \dots & 1 \end{pmatrix}, \tag{23}$$

dove $l_{i,k} = \frac{a_{i,k}^{k-1}}{a_{k,k}^{k-1}}$, $i = 1, \dots, n$. In forma matriciale si ottiene

$$D = A^n = (J_n \dots J_2 J_1) A$$

dove D risulta essere una matrice diagonale. Applicando le stesse trasformazioni anche al termine noto b, i.e.

$$d = b^n = (J_n \dots J_2 J_1) b,$$

si ottiene un sistema diagonale $Dx = d$ equivalente a quello di partenza, che si risolve in n operazioni.

Esercizio 2 *Scrivi una pseudocodifica per l'algoritmo di Gauss-Jordan.*

La sua stabiltà può essere migliorata applicando la tecnica del pivot parziale nella parte inferiore della matrice. In questo caso risulta che l'algoritmo di Gauss-Jordan con pivot parziale è stabile quanto quello di Gauss con pivot parziale.

L'algoritmo di Gauss-Jordan richiede circa $\frac{n^3}{2}$ moltiplicazioni.

Può essere applicato per il calcolo dell'inversa.

6 Errore e residuo

Il **residuo** di una soluzione approssimata $\tilde{x}$ è definito da

$$r = b - A\tilde{x}. \tag{24}$$

Poichè, se A è una matrice non singolare, vale che l'errore (assoluto) $\|x - \tilde{x}\| = 0$ se e soltando se $\|r\| = 0$, si potrebbe pensare di stimare l'errore relativo valutando la norma del **residuo relativo** $\frac{\|r\|}{\|b\|}$. In pratica un "piccolo" residuo relativo mi assicura un "piccolo" errore relativo nella soluzione solo se la matrice è ben condizionata. Vale infatti

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \mathrm{cond}(A) \frac{\|r\|}{\|b\|}. \tag{25}$$

Volendo analizzare l'errore relativo rispetto alla soluzione approssimata si ottiene la seguente stima

$$\frac{\|x-\tilde{x}\|}{\|\tilde{x}\|} \leq \mathrm{cond}(A)\frac{\|r\|}{\|A\|\cdot\|\tilde{x}\|}, \tag{26}$$

che conferma il ruolo del condizionamento della matrice $\mathrm{cond}(A)$ nella stima del'errore attraverso la valutazione del residuo e che un piccolo" residuo non mi assicura una soluzione accurata.

NOTA 7 *Ricordando che la soluzione calcolata $\tilde{x}$ è soluzione esatta di un problema perturbato (16) si può facilemente ottenere la seguente stima*

$$\frac{\|r\|}{\|A\|\cdot\|\tilde{x}\|} \leq \frac{\|E\|}{\|A\|}. \tag{27}$$

La sua analisi ci permette di concludere che

- *un residuo "grande" implica un "grande" errore (all'indietro) nella matrice A che mi dice che l'algoritmo usato per calcolare $\tilde{x}$ è instabile (all'indietro);*
- *un algoritmo stabile (all'indietro) fornisce un "piccolo" errore nel residuo. D'altro canto (26) non ci garantisce che un "piccolo" errore nel residuo sia indice di soluzione accurata.*

7 Raffinamento iterativo

Sia $\tilde{x}$ una soluzione approssimata della soluzione x in un sistema lineare calcolata, per esempio, con l'algoritmo di eliminazione di Gauss. Come già osservato nel paragrafo precedente, il vettore errore $e = x - \tilde{x}$ è soluzione (esatta) del sistema $Ae = r$ e la soluzione si ottiene con $x = \tilde{x} + e$. Risolvendo il sistema in aritmetica di macchina si otterrà una stima dell'errore con cui "raffinare" la soluzione iniziale. Sia $x_0 = \tilde{x}$ in maniera *iterativa*, l'algoritmo di *raffinamento iterativo* si propone di migliorare la stima iniziale potenzialmente fino alla precisione di macchina u nel seguente modo:

```
x_{0}=\tilde{x}
d=x_{0}
for k=1,2,\ldots,k_{max}
    r_{k-1}=b-Ax_{k-1}
    ‘‘in doppia precisione’’ risolvi
    A(e_{k-1})=r_{k-1}
    x_{k}=x_{k-1}+e_{k-1}
    if \|e_{k-1}\|\cdot\|x_{k-1}\|\geq \|d\|\cdot\|x_{k}\|
        then stop
        ‘‘l’iterazione non converge’’
    end
```

```
    if \|e_{k-1}\|\leq u |x_{k}\|
        then stop iteration
        ‘‘stima ottenuta’’
    end
    d=e_{k-1}
end
```

Tale algoritmo richiede la memorizzazione sia della matrice iniziale A per calcolare ad ogni passo iterativo il residuo, sia delle matrici L, U, P della fattorizzazione per risolvere in maniera efficiente ad ogni passo il sistema lineare per calcolare le correzioni e_{k-1}. Inoltre, poichè il calcolo dei residui comporta la cancellazione, risulta indispensabile che siano calcolati con precisione più elevata per ottenere un effettivo miglioramento. Si può dimostrare che se la matrice A non è troppo malcondizionata il test di arresto viene verificato. In pratica se in poche iterazioni non è soddisfatto è inutile proseguire: la matrice è molto malcondizionata e può capitare che le norme delle correzioni $\|e_{k-1}\|$ vadano aumentando.

8 Istruzioni MATLAB

L'istruzione fondamentale per la risoluzione di sistemi lineari $Ax = b$, con $A \in \mathbb{R}^{n\times n}$ non singolare e $b \in \mathbb{R}^n$ vettore dei termini noti, è l'operatore *backslash* "\". Anche se qui ci occupiamo di sistemi lineari identificati da una matrice quadrata, tale operatore si utilizza anche per sistemi sovra- e sotto-determinati (matrici rettangolari). La soluzione $x \in \mathbb{R}^n$ si ottiene mediante l'istruzione MATLAB

```
>>x=A\b
```

calcolata mediante la fattorizzazione LU di A con pivoting parziale ($PA = LU$) e la risoluzione dei due sistemi triangolare inferiore con matrice L (sostituzione in avanti) e triangolare superiore con matrice U (sostituzione all'indietro).

La fattorizzazione $PA = LU$ può essere calcolata direttamente con l'istruzione *lu*. La sintassi

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

restituisce la matrice triangolare inferiore L con diagonale unitaria, la matrice triangolare superiore U e la matrice di permutazione P. La sintassi

```
>>[LT,U]=lu(A)
```

restituisce al posto della matrice triangolare inferiore L con diagonale unitaria la sua permutazione $LT = P^T L$.

Dietro all'istruzione $x = A \setminus b$ ci sono quindi le istruzioni

```
>>[L,U,P]=lu(A);
>>x=U\(L\(P*b));
```

oppure

```
>>[L,U]=lu(A);
>>x=U\(L\b);
```

In realtà, alla chiamata $x = A \setminus b$, MATLAB tenta di fattorizzare la matrice A mediante il metodo di Cholesky per matrici simmetriche definite positive. Se la fattorizzazione ha successo viene usata per risolvere il sistema, altrimenti usa la fattorizzazione $PA = LU$. La fattorizzazione $A = HH^T$ di Cholesky può essere calcolata direttamente con l'istruzione *chol*. La sintassi

```
>>R=chol(A)
```

restituisce la matrice triangolare superiore $R = H^T$ tale che $A = R^T R$. La sintassi

```
>>[R,p]=chol(A)
```

restituisce anche un intero p. Se $p = 0$, significa che la fattorizzazione è riuscita, ovvero A è effettivamente simmetrica definita positiva. Se $p > 0$, significa che A non è fattorizzabile secondo Cholesky. In quest'ultimo caso la matrice R che viene restituita non ha significato ai fini della fattorizzazione.

8.1 Norma

L'istruzione MATLAB per il calcolo della norma p di un vettore $x \in \mathbb{R}^n$ è *norm*. La sintassi

```
>>norm(x,p)
```

restituisce

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

per $p \leq 1 < \infty$, mentre per $p = \infty$ (*Inf* in MATLAB) restituisce

$$\|x\|_\infty = \max_{i=1}^{n} |x_i|.$$

Il default è $p = 2$, ovvero la norma euclidea.

L'istruzione *norm* calcola anche la norma p di una matrice $A \in \mathbb{R}^{m \times n}$ definita da

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}.$$

In particolare, la chiamata

```
>>norm(A,p)
```

supporta $p = 1$, $p = 2$, $p = Inf$ e $p =' fro'$ rispettivamente per $\|A\|_1$ (massima somma delle colonne), $\|A\|_2$ (massimo valore singolare, vedi *svd* di MATLAB), $\|A\|_\infty$ (massima somma delle righe) e la norma di Frobenius

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}.$$

Quando il calcolo di $\|A\|_2$ è troppo costoso, la funzione *normest* con sintassi

```
>>normest(A,tol)
```

può essere usata per ottenerne una stima con una tolleranza *tol* (default: $tol = 10^{-6}$).

8.2 Numero di condizionamento

L'istruzione MATLAB per il calcolo del numero di condizionamento nella risoluzione di un sistema lineare $Ax = b$ con $A \in \mathbb{R}^{n\times n}$ non singolare è *cond*. La sintassi

```
>>{\rm cond}(A,p)
```

restituisce

$$\mathrm{cond}_p(A) = \|A\|_p \|A^{-1}\|_p \geq 1$$

e supporta $p = 1$, $p = 2$, $p = Inf$ e $p =' fro'$.

In generale il calcolo del numero di condizionamento è costoso perchè richiede la computazione esplicita dell'inversa. In questo caso MATLAB provvede le funzioni *condest* e *rcond*. In particolare

```
>>condest(A)
```

stima $\mathrm{cond}_1(A) \geq 1$ mentre

```
>>rcond(A)
```

stima $1/\mathrm{cond}_1(A) \in [0, 1]$.

8.3 Altre istruzioni per matrici

Una matrice $A \in \mathbb{R}^{n\times n}$ è non singolare quando $\det A \neq 0$. L'istruzione MATLAB per il calcolo del determinante di una matrice quadrata è *det* con sintassi

```
>>det(A)
```

L'inversa di una matrice $A \in \mathbb{R}^{n\times n}$ è la matrice $X \in \mathbb{R}^{n\times n}$ tale che $AX = XA = I$, dove I è la matrice identità in $\mathbb{R}^{n\times n}$ ($I = eye(n)$). L'istruzione MATLAB per il calcolo esplicito dell'inversa è *inv* con sintassi

```
>>inv(A)
```

L'inversa di una matrice A viene formata mediante la fattorizzazione $PA = LU$. Si osservi che è raramente necessario il calcolo esplicito dell'inversa. Per esempio, la soluzione del sistema lineare $Ax = b$ mediante $x = A \setminus b$ è più veloce che utilizzare $x = inv(A) * b$. **È normalmente possibile riformulare computazioni coinvolgenti l'inversa di una matrice in termini di risoluzione di sistemi lineari, cosicchè si può evitare il calcolo esplicito dell'inversa**.

Si riporta, infine, per completezza l'istruzione MATLAB *eig* per il calcolo degli autovalori di una matrice $A \in \mathbb{R}^{n \times n}$. La sintassi è

```
>>eig(A)
```

Se la matrice è sparsa è conveniente utilizzare *eigs* anzichè *eig*.

NOTA 8 *Per tutti i comandi precedenti si consiglia di consultare l'help e la documentazione (doc) in linea di MATLAB.*

9 Esercizi svolti

Esercizio 3 *(vedi anche* 09/12/02 − ♯2*) Considera la seguente matrice non singolare*

$$A = \begin{pmatrix} 0 & 4 & 0 \\ 0 & 2 & 1 \\ 2 & 3 & 1 \end{pmatrix}.$$

Si può applicare l'algoritmo di eliminazione di Gauss per determinare la fattorizzazione LU di A? Perchè? Applica la variante del pivot parziale alla matrice A calcolando P, L e U. Infine calcola il determinate di A usando la fattorizzazione ottenuta.

Soluzione. L'algoritmo di Gauss per la fattorizzazione LU di A non funziona dato che il primo elemento sulla diagonale prinicipale di A è nullo, ovvero A non è fattorizzabile secondo $A = LU$, con L triangolare inferiore con diagonale unitaria e U triangolare superiore, a meno che prima non si permutino opportunamente le sue righe. Bisogna dunque applicare l'algoritmo con la strategia del pivot parziale, ovvero eseguire la fattorizzazione $PA = LU$ dove P è la matrice di permutazione. Si determina dunque P_1 in modo che il primo elemento sulla diagonale sia il massimo (in valore assoluto) tra quelli della prima colonna:

$$P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \Rightarrow P_1 A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 4 & 0 \\ 0 & 2 & 1 \\ 2 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 \\ 0 & 2 & 1 \\ 0 & 4 & 0 \end{pmatrix}.$$

Si calcola poi la prima matrice moltiplicatore M_1 per eliminare gli elementi della prima colonna a partire dal secondo: tali elementi sono già nulli e quindi

$$M_1 = I \Rightarrow M_1 P_1 A = \begin{pmatrix} 2 & 3 & 1 \\ 0 & 2 & 1 \\ 0 & 4 & 0 \end{pmatrix}.$$

Si procede poi con la seconda permutazione per avere come secondo elemento sulla diagonale principale quello massimo (sempre in valore assoluto) tra gli elementi della seconda colonna a partire dal secondo in poi, dunque

$$\begin{aligned} P_2 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \Rightarrow \\ \Rightarrow P_2 M_1 P_1 A &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & 3 & 1 \\ 0 & 2 & 1 \\ 0 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 \\ 0 & 4 & 0 \\ 0 & 2 & 1 \end{pmatrix}. \end{aligned}$$

Infine si calcola la seconda e ultima matrice moltiplicatore M_2

$$\begin{aligned} M_2 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{pmatrix} \Rightarrow \\ \Rightarrow M_2 P_2 M_1 P_1 A &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{pmatrix} \begin{pmatrix} 2 & 3 & 1 \\ 0 & 4 & 0 \\ 0 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix} = U. \end{aligned}$$

In questo modo si è ottenuta la fattorizzazione $A = \tilde{L}U$ dove, ricordando che per P matrice di permutazione è $P^{-1} = P^T$,

$$\begin{aligned} \tilde{L} &= (M_2 P_2 M_1 P_1)^{-1} = P_1^{-1} M_1^{-1} P_2^{-1} M_2^{-1} = \\ &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 1 \end{pmatrix} = \\ &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1/2 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \end{aligned}$$

$\tilde{L}$ non è una matrice triangolare inferiore ma una permutazione di quest'ultima: $\tilde{L} = P^T L$ con $P = P_1^T P_2^T$. Risulta infatti

$$\begin{aligned} LU &= PA = P\tilde{L}U \Rightarrow L = P\tilde{L} = P_2 P_1 \tilde{L} = \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1/2 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 1 \end{pmatrix} \end{aligned}$$

che è triangolare inferiore con diagonale unitaria.

Si supponga ora di voler calcolare il determinante di A. A questo scopo si può sfruttare la fattorizzazione $PA = LU$ osservando che

- $\det PA = \pm \det A$ in quanto il determinante di una matrice cambia segno se la permutazione P è dispari,
- $\det L = 1$ in quanto per matrici triangolari il determinante è il prodotto degli elementi sulla diagonale principale e nel caso di L questi sono tutti uno.

Risulta perciò (P è pari, cioè si sono eseguite due permutazioni)

$$\det A = \det L \det U = \det U = 2 \cdot 4 \cdot 1 = 8.$$

L'istruzione *lu* di MATLAB consente, a seconda della sintassi usata, di ottenere entrambe le fattorizzazioni $A = \tilde{L}U$ e $PA = LU$:

```
>>A=[0,4,0;0,2,1;2,3,1]
A=
    0     4     0
    0     2     1
    2     3     1
>>[LT,U]=lu(a)
LT=
         0    1.0000         0
         0    0.5000    1.0000
   1.0000         0         0
U=
    2     3     1
    0     4     0
    0     0     1
>>[L,U,P]=lu(a)
L=
   1.0000         0         0
         0    1.0000         0
         0    0.5000    1.0000
U=
    2     3     1
    0     4     0
    0     0     1
P=
    0     0     1
    1     0     0
    0     1     0
```

Se utilizzata invece con un solo argomento di output, *lu* restituisce una matrice B composita di U e di L in cui non è memorizzata la diagonale unitaria di quest'ultima:

```
>>B=lu(A)
B=
   2.0000    3.0000    1.0000
         0    4.0000         0
         0    0.5000    1.0000
```

Infine MATLAB usa *det* per calcolare il determinante di una matrice quadrata, basandosi proprio sulla fattorizzazione $PA = LU$:

```
>>det(A),det(P*A),det(L),det(U)
ans=
    8
ans=
    8
ans=
    1
ans=
    8
```

■

Esercizio 4 *(vedi anche* $04/07/02 - \sharp 3$*) Sia* $A \in \mathbb{R}^{n\times n}$ *una matrice non singolare. Risolvi in maniera efficiente, usando l'algoritmo di eliminazione di Gauss con il pivot parziale, il problema seguente*

$$A^k x = b$$

dove k *è un intero positivo fissato e* $b \in \mathbb{R}^n$*. In particolare descrivi l'algoritmo da te proposto, presenta una pseudocodifica (simile al linguaggio MATLAB) senza scrivere l'algoritmo di eliminazione di Gauss e stima il numero di operazioni (FLOPS) richieste.*

Soluzione. La via più conveniente è quella di determinare un'unica volta la fattorizzazione $PA = LU$ (costo $\mathcal{O}(n^3/3)$) e sfruttarla per risolvere successivamente k sistemi lineari, ciascuno scomposto in uno triangolare inferiore (risolto per sostituzione in avanti, costo $\mathcal{O}(n^2/2)$) e uno triangolare superiore (risolto per sostituzione all'indietro, costo $\mathcal{O}(n^2/2)$). Una possibile implementazione dell'algoritmo si basa sull'uso della *lu* di MATLAB e dell'operatore \ per la risoluzione di sistemi lineari. Il corpo della funzione è

```
[L,U,P]=lu(A);
y(1)=b;
for i=1:k
    y(i+1)=U\(L\(P*y(i)));
end
x=y(k+1)
```

e all'i-esimo passo risolve il sistema lineare

$$Ay_{i+1} = y_i$$

mediante i due triangolari (inferiore e superiore)

$$\begin{cases} Lz_i = Py_i \\ Uy_{i+1} = z_i \end{cases}$$

e partendo da $y_1 = b$ si ottiene

$$b = y_1 = Ay_2 = A^2 y_3 = \cdots = A^k y_{k+1} = A^k x.$$

Il numero di moltiplicazioni C_* e di addizioni C_+ è dato rispettivamente da:

$$\begin{aligned} C_* &= \frac{1}{3}n(n^2-1) + kn(n+1) = \mathcal{O}\left(\frac{n^3}{3}\right) \\ C_+ &= n\left(\frac{1}{3}n^2 - \frac{1}{2}n + \frac{1}{6}\right) + kn(n-1) = \mathcal{O}\left(\frac{n^3}{3}\right) \end{aligned}$$

La risoluzione diretta del sistema lineare calcolando il prodotto A^k non è affatto conveniente dato che il prodotto di k-matrici piene ha un costo computazionale di $\mathcal{O}\left((k-1)n^3\right)$. ■

Esercizio 5 *(vedi anche 24/09/02 − ♯3) Sia $A \in \mathbb{R}^{n\times n}$ una matrice non singolare. Scrivi un algoritmo efficiente, usando il metodo di eliminazione di Gauss con il pivot parziale, per valutare*

$$y = c^T A^{-1} b$$

dove $b, c \in \mathbb{R}^n$. In particolare descrivi l'algoritmo da te proposto, presenta una pseudocodifica (simile al linguaggio MATLAB) senza scrivere l'algoritmo di eliminazione di Gauss e stima il numero di operazioni (FLOPS) richieste.

Soluzione. Il calcolo esplicito dell'inversa di una matrice non è conveniente, in quanto corrisponde alla risoluzione di n sistemi lineari

$$Ax^{(i)} = e^{(i)}$$

dove $e^{(i)}$ è l'i-esimo vettore della base canonica in $\mathbb{R}^n$ e $x^{(i)}$ è l'i-esima colonna della matrice inversa A^{-1}. Nota la fattorizzazione di A, il costo computazionale di tale processo è $\mathcal{O}(n^3)$. Risulta perciò più efficiente il seguente algoritmo:

```
[L,U,P]=lu(A);
z=L\(P*b);
x=U\z;
y=c'*x;
```

che fornisce la fattorizzazione $PA = LU$ con complessità $\mathcal{O}(n^3/3)$, risolve il sistema lineare $Ax = b$ mediante i due triangolari $Lz = Pb$ e $Ux = z$ con complessità $\mathcal{O}(n^2)$ e calcola il prodotto scalare $y = c^T x$ con complessità $\mathcal{O}(n)$. ■

Esercizio 6 *(vedi anche 24/03/03 − ♯2) Date due matrici $A, B \in \mathbb{R}^{n\times n}$ con A non singolare e due vettori $c, d \in \mathbb{R}^n$, proponi un algoritmo efficiente per determinare i vettori $x, y \in \mathbb{R}^n$ che soddisfano entrambe le relazioni*

$$A^T y + Bx = d, \quad Ax = c$$

supponendo di poter calcolare $A = LU$ con l'algoritmo di eliminazione di Gauss. In particolare descrivi l'algoritmo da te proposto, presenta una pseudocodifica

(simile al linguaggio MATLAB) senza scrivere l'algoritmo di eliminazione di Gauss e stima il numero di operazioni (FLOPS) richieste. Come cambia l'algoritmo se la matrice A è fattorizzabile in $PA = LU$ (metodo di eliminazione di Gauss con il pivot parziale)?

Soluzione. Un modo efficiente per risolvere il problema è implementato nel seguente algoritmo:

```
[L,U]=lu(A);
z=L\c;
x=U\z;
e=d-B*x;
v=U'\e;
y=L'\v;
```

che fornisce la fattorizzazione $A = LU$ con complessità $\mathcal{O}(n^3/3)$, risolve il sistema lineare $Ax = c$ mediante i due triangolari $Lz = c$ e $Ux = z$ con complessità $\mathcal{O}(n^2)$, calcola il vettore $e = d - Bx$ con complessità $\mathcal{O}(n^2)$ e infine risolve il sistema lineare $A^T y = e$ mediante i due triangolari $U^T v = e$ e $L^T y = v$ con complessità $\mathcal{O}(n^2)$ senza dover fattorizzare A^T in quanto $A^T = U^T L^T$.

Qualora A non sia fattorizzabile secondo $A = LU$, si deve ricorrere alla strategia del pivot parziale, ovvero alla fattorizzazione $PA = LU$. In tal caso l'algoritmo precedente diventa

```
[L,U,P]=lu(A);
z=L\(P*c);
x=U\z;
e=d-B*x;
v=U'\e;
y=P'*(L'\v);
```

in quanto vanno risolti i sistemi lineari $Lz = Pc$ e $L^T Py = v$. L'ultimo in particolare segue da

$$PA = LU \Rightarrow (PA)^T = (LU)^T \Rightarrow A^T P^T = U^T L^T \Rightarrow A^T = U^T L^T P^{-T}$$

per cui vanno risolti i sistemi triangolari $U^T v = e$ e $L^T Py = v$ (si ricordi che $P^T = P^{-1}$). ■

Esercizio 7 *(vedi anche* $10/07/03 - \sharp 3$*) Siano date due matrici $A, B \in \mathbb{R}^{n\times n}$ con A non singolare e due vettori $c, d \in \mathbb{R}^{n\times 1}$. Assumendo di aver già calcolato la fattorizzzazione LU di A, come la useresti per determinare i due vettori $x, y \in \mathbb{R}^{n\times 1}$ che soddisfano entrambe le relazioni*

$$Ay + Bx = d, \quad A^T x = c$$

dove A^T è la trasposta della matrice A? Come cambia l'algoritmo conoscendo la fattorizzazione $PA = LU$ (metodo di eliminazione di Gauss con il pivot parziale)?

Soluzione. Un modo efficiente per risolvere il problema è implementato nel seguente algoritmo:

```
[L,U]=lu(A);
z=U'\c;
x=L'\z;
e=d-B*x;
v=L\e;
y=U\v;
```

che fornisce la fattorizzazione $A = LU$ con complessità $\mathcal{O}(n^3/3)$, risolve il sistema lineare $A^T x = c$ (senza dover fattorizzare A^T in quanto $A^T = U^T L^T$) mediante i due triangolari $U^T z = c$ e $L^T x = z$ con complessità $\mathcal{O}(n^2)$, calcola il vettore $e = d - Bx$ con complessità $\mathcal{O}(n^2)$ e infine risolve il sistema lineare $Ay = e$ mediante i due triangolari $Lv = e$ e $Uy = v$ con complessità $\mathcal{O}(n^2)$.

Qualora A non sia fattorizzabile secondo $A = LU$, si deve ricorrere alla strategia del pivot parziale, ovvero alla fattorizzazione $PA = LU$. In tal caso l'algoritmo precedente diventa

```
[L,U,P]=lu(A);
z=U'\c;
x=P'*(L'\z);
e=d-B*x;
v=L\(P*e);
y=U\v;
```

in quanto vanno risolti i sistemi lineari $U^T L^T P^{-T} x = c$ e $LUy = PAy = Pd - PBx$ (si ricordi che $P^T = P^{-1}$). ■

Esercizio 8 *(vedi anche* 24/07/03−♯2*) Sia data una matrice non singolare* $A \in \mathbb{R}^{n \times n}$ *e un vettore* $b \in \mathbb{R}^n$*. Assumendo di aver già calcolato la fattorizzzazione* LU *di* A*, come la useresti per calcolare in maniera efficiente la soluzione* $x \in \mathbb{R}^{n \times 1}$ *del sistema*

$$(A^k)^T x = b,$$

dove $k \geq 1$ *è un prefissato intero e* M^T *indica la trasposta di una matrice* M*? Analizza la complessità computazionale dell'algoritmo da te proposto. Come cambia l'algoritmo conoscendo la fattorizzazione* $PA = LU$*?*

Soluzione. La via più conveniente è quella di determinare un'unica volta la fattorizzazione $A = LU$ (costo $\mathcal{O}(n^3/3)$) e sfruttarla per risolvere successivamente k sistemi lineari, ciascuno scomposto in uno triangolare inferiore (risolto per sostituzione in avanti, costo $\mathcal{O}(n^2/2)$) e uno triangolare superiore (risolto per sostituzione all'indietro, costo $\mathcal{O}(n^2/2)$). Si tenga conto infatti che $A^T = U^T L^T$ e $(A^k)^T = (A^T)^k$. Una possibile implementazione dell'algoritmo si basa sull'uso della *lu* di MATLAB e dell'operatore \ per la risoluzione di sistemi lineari. Il corpo della funzione è

```
[L,U]=lu(A);
y(1)=b;
for i=1:k
    y(i+1)=L'\(U'\y(i));
end
x=y(k+1);
```

e all'i-esimo passo risolve il sistema lineare

$$A^T y_{i+1} = y_i$$

mediante i due triangolari (inferiore e superiore)

$$\begin{cases} U^T z_i = y_i \\ L^T y_{i+1} = z_i \end{cases}$$

e partendo da $y_1 = b$ si ottiene

$$b = y_1 = A^T y_2 = (A^T)^2 y_3 = \cdots = (A^T)^k y_{k+1} = (A^T)^k x = (A^k)^T x.$$

Il numero di moltiplicazioni C_* e di addizioni C_+ è dato rispettivamente da:

$$\begin{aligned} C_* &= \frac{1}{3}n(n^2-1) + kn(n+1) = \mathcal{O}\left(\frac{n^3}{3}\right) \\ C_+ &= n\left(\frac{1}{3}n^2 - \frac{1}{2}n + \frac{1}{6}\right) + kn(n-1) = \mathcal{O}\left(\frac{n^3}{3}\right) \end{aligned}$$

La risoluzione diretta del sistema lineare calcolando il prodotto A^k non è affatto conveniente dato che il prodotto di k-matrici piene ha un costo computazionale di $\mathcal{O}\left((k-1)n^3\right)$.

Qualora A non sia fattorizzabile secondo $A = LU$, si deve ricorrere alla strategia del pivot parziale, ovvero alla fattorizzazione $PA = LU$. In tal caso l'algoritmo precedente diventa

```
[L,U,P]=lu(A);
y(1)=b;
for i=1:k
    y(i+1)=P'*(L'\(U'\y(i)));
end
x=y(k+1);
```

poichè $A^T = (P^{-1}LU)^T = U^T L^T P^{-T} = U^T L^T P$. ■

Esercizio 9 *(vedi anche* $16/09/03 - \sharp 3$*) Siano date le matrici* $A, B, C \in \mathbb{R}^{n\times n}$ *con* A, C *non singolari ed il vettore* $d \in \mathbb{R}^{n\times 1}$*. Descrivi i passi da compiere per determinare il vettore* $x \in \mathbb{R}^{n\times 1}$ *definito dalla seguente relazione*

$$x = A^{-1}(B+I)C^{-1}d,$$

senza calcolare le matrici inverse. Fornisci una valutazione del costo computazionale dell'algoritmo proposto.

Soluzione. Il seguente algoritmo risolve efficientemente il problema senza calcolare le matrici inverse. Si suppone che A e C siano fattorizzabili senza l'uso del pivot parziale, ovvero $A = L_A U_A$ e $C = L_C U_C$.

```
[LC,UC]=lu(C);
y=L\d;
z=U\y;
v=(B+I)z;
[LA,UA]=lu(A);
w=L\v;
x=U\w;
```

che risolve il sistema lineare $Cz = d$ trovando $z = C^{-1}d$ (costo $\mathcal{O}(n^3/3)+\mathcal{O}(n^2)$), effettua la moltiplicazione $v = (B + I)z$ (costo $\mathcal{O}(n^2)$) e infine risolve il sistema lineare $Ax = v$ trovando $x = A^{-1}v$ (costo $\mathcal{O}(n^3/3) + \mathcal{O}(n^2)$). Il costo totale (per le moltiplicazioni) risulta $\mathcal{O}(2n^3/3) + \mathcal{O}(3n^2)$. ■

10 Esercizi da svolgere

Esercizio 10 *Parla del condizionamento nella risoluzione di un sistema lineare con A matrice quadrata di dimensione n non singolare. Dopo aver definito il numero di condizionamento di A, valuta come cambia per le seguenti matrici:*

- *αA, con α scalare non nullo;*
- *A^{-1};*
- *PA, con P matrice di permutazione.*

Esercizio 11 *Descrivi dettagliatamente l'algoritmo di eliminazione di Gauss per risolvere un sistema lineare $Ax = b$ con A matrice quadrata non singolare di dimensione n. In particolare*

- *scrivi una pseudocodifica;*
- *analizza la sua complessità computazionale;*
- *descrivi la tecnica di pivoting, elencando due motivi per applicarla.*

Il costo computazionale ci permette di stimare il tempo di esecuzione di un algoritmo. Rispetto ad un sistema di dimensione n, di quanto stimi l'aumento del tempo di esecuzione dell'algoritmo di eliminazione di Gauss per un sistema di dimensione $10*n$? Dovendo risolvere in modo efficiente dieci volte un sistema di dimensione n con la stessa matrice A ma con termini noti b diversi, quale è la stima del costo computazionale?

Esercizio 12 *Sia data la seguente matrice*

$$A = \begin{pmatrix} -3 & 0 & -1 \\ -12 & 8 & -12 \\ -1 & 0 & -3 \end{pmatrix}.$$

Dopo aver verificato che

$$B = \frac{1}{8}\begin{pmatrix} -3 & 0 & 1 \\ -3 & 1 & -3 \\ 1 & 0 & -3 \end{pmatrix}$$

è la sua inversa, fornisci una limitazione dell'errore inerente della soluzione del sistema $Ax = b$, supponendo di introdurre un'errore relativo nel termine noto b per cui vale la seguente limitazione:

$$\frac{\|\delta b\|_\infty}{\|b\|_\infty} \leq 10^{-4}.$$

Esercizio 13 *Descrivi dettagliatamente un algoritmo per risolvere un sistema lineare $Lx = b$ con L triangolare inferiore, analizzando anche la sua complessità computazionale. Date le matrici L_1, L_2, $B \in \mathbb{R}^{n\times n}$ con L_1, L_2 non singolari e triangolari inferiori e due vettori c, $d \in \mathbb{R}^n$, descrivi i passi da compiere per determinare i vettori x, $y \in \mathbb{R}^n$ che soddisfano entrambe le relazioni*

$$Bx + L_2 y = d, \quad L_1 x = c.$$

Fornisci anche una stima del costo computazionale complessivo dell'algoritmo proposto.

Esercizio 14 *Calcola per la seguente matrice*

$$A = \begin{pmatrix} 4 & -8 & 1 \\ 6 & 5 & 7 \\ 0 & -10 & -2 \end{pmatrix}$$

la fattorizzazione $PA = LU$ mediante l'algoritmo di eliminazione di Gauss con pivot parziale. Come useresti tale fattorizzazione per risolvere il sistema $A^T x = b$ per $b = (10, -13, 6)^T$?

Esercizio 15 *Considera la seguente matrice non singolare*

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 1 \\ 2 & 3 & 1 \end{pmatrix}.$$

Si può applicare l'algoritmo di eliminazione di Gauss per determinare la fattorizzazione LU di A ? Perchè? Descrivi la variante del pivot parziale e applicala alla matrice A calcolando P, L, ed U. Infine calcola il determinate di A usando la fattorizzazione ottenuta.

11 Domande di verifica

- Definisci il numero di condizionamento di una matrice e spiega il suo significato.

- Perchè si applica la tecnica del pivot parziale?
- Esiste sempre la fattorizzazione LU di una matrice non singolare?
- Quale algoritmo si usa per risolvere un sistema lineare la cui matrice è simmetrica e definita positiva?
- Il residuo fornisce una buona stima dell'accuratezza della soluzione?

Riferimenti bibliografici

[QS02] **testo consigliato:** Quarteroni, A., Saleri, F. (2002) *Introduzione al Calcolo Scientifico*, Springer, Milano.

[BBCM92] Bevilacqua, R., Bini, D., Capovani, M. e Menchi, O. (1992) *Metodi Numerici*, Zanichelli.

[Mon98] Monegato, G. (1998) *Fondamenti di Calcolo Numerico*, Clut.

[NPR01] Naldi, G., Pareschi, L. e Russo, G. (2001) *Introduzione al Calcolo Scientifico*, Mc Graw Hill Ed.

[Hea02] Heath, M.T. (2002) *Scientific Computing*, Mc Graw Hill Ed.

[Ste96] Stewart, G.W. (1996) *Afternotes on Numerical Analysis*, SIAM Ed.