Triangolazione a blocchi di matrici

Matrici quadrate sparse, cioè contenenti zeri, possono essere, sotto opportune ipotesi, ricondotte alla comoda forma "triangolare a blocchi" per mezzo di semplici permutazione righe-colonne.

1	2	1	0	0	0
2	1	5	0	0	0
1	-1	3	0	0	0
-6	5	3	1	1	2
1	-3	2	1	-1	-2
-9	7	1	1	2	1

A_1	0
A_{21}	A_2

La forma a blocchi consente un notevole risparmio di sforzo elaborativo per i più comuni e importanti calcoli di algebra lineare quali: sistemi lineari, determinanti, autovalori.

Risoluzione Sistema a Blocchi

Per esempio un sistema lineare del tipo

$\mathbf{A}\,\mathbf{x} = \mathbf{b}$

$$\begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 \\ 2 & 1 & 5 & 0 & 0 & 0 \\ 1 & -1 & 3 & 0 & 0 & 0 \\ -6 & 5 & 3 & 1 & 1 & 2 \\ 1 & -3 & 2 & 1 & -1 & -2 \\ -9 & 7 & 1 & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}$$

Può essere scritto nella forma

$$\mathbf{A}_1\,\mathbf{x}_1 = \mathbf{b}_1$$

$$\mathbf{A}_2\,\mathbf{x}_2 = \mathbf{b}_2 - \mathbf{c}_2$$

dove il vettore $\mathbf{c}_2$ è dato da: $\mathbf{c}_2 = \mathbf{A}_{21}\,\mathbf{x}_1$

In pratica, il sistema originale (6 x 6) è ricondotto alla risoluzione due sotto-sistemi (3 x 3)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 5 \\ 1 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & -1 & -2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} b_4 \\ b_5 \\ b_6 \end{bmatrix} - \begin{bmatrix} -6 & 5 & 3 \\ 1 & -3 & 2 \\ -9 & 7 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Calcolo Determinante

Anche il calcolo del determinante si avvantaggia della forma a blocchi.
Ad esempio il determinante della matrice (6 x 6) è dato dal prodotto dei determinanti delle due matrici (3 x 3) $\mathbf{A}_1$ e $\mathbf{A}_2$ della diagonale principale

$$\begin{vmatrix} 1 & 2 & 1 & 0 & 0 & 0 \\ 2 & 1 & 5 & 0 & 0 & 0 \\ 1 & -1 & 3 & 0 & 0 & 0 \\ -6 & 5 & 3 & 1 & 1 & 2 \\ 1 & -3 & 2 & 1 & -1 & -2 \\ -9 & 7 & 1 & 1 & 2 & 1 \end{vmatrix} = 18 \qquad \begin{vmatrix} 1 & 2 & 1 \\ 2 & 1 & 5 \\ 1 & -1 & 3 \end{vmatrix} = 3 \qquad \begin{vmatrix} 1 & 1 & 2 \\ 1 & -1 & -2 \\ 1 & 2 & 1 \end{vmatrix} = 6$$

Permutazioni

Diversamente dagli algoritmi di fattorizzazione (Gauss, LR, ecc...) la riduzione di matrici a blocchi deve usare solo operazioni di permutazioni di righe e colonne. Dal punto di vista formale tali operazioni possono essere assimilate ad una trasformazione per contragradienza.
Ad esempio, data una matrice **A** (6 x 6) lo scambio della riga 2 con la 5 seguito dallo scambio della colonna 2 con la 5 può essere scritto formalmente (ma solo formalmente) come:

$$\mathbf{B} = \mathbf{P}^T \mathbf{A} \mathbf{P} \quad , \qquad \text{dove la matrice di permutazione } \mathbf{P} = (e_1, e_5, e_3, e_4, e_2, e_6)$$

A

1	0	0	1	2	0
1	-1	1	2	-3	-2
-6	1	1	3	5	2
1	0	0	3	-1	0
2	0	0	5	1	0
-9	2	1	1	7	1

P

1	0	0	0	0	0
0	0	0	0	1	0
0	0	1	0	0	0
0	0	0	1	0	0
0	1	0	0	0	0
0	0	0	0	0	1

$\mathbf{P}^T \mathbf{A} \mathbf{P}$

1	2	0	1	0	0
2	1	0	5	0	0
-6	5	1	3	1	2
1	-1	0	3	0	0
1	-3	1	2	-1	-2
-9	7	1	1	2	1

Nota. Dal punto di vista del calcolo numerico invece il prodotto di matrice è un'operazione estremamente costosa e deve essere evitata; si usa lo scambio diretto righe-colonne o meglio ancora lo scambio degli indici.

Si deve mettere in evidenza che la trasformazione per contragradienza conserva gli autovalori della matrice. Quindi gli autovalori della matrice **A** sono gli stessi della matrice **B**

Ricerca di autovalori

Anche il calcolo degli autovalori può essere molto semplificato se la matrice è a blocchi.
Ad esempio la seguente matrice A (6 x 6) ha gli autovalori :

$$\lambda = [-7 , -1 , 1 , 2 , 3 , 5]$$

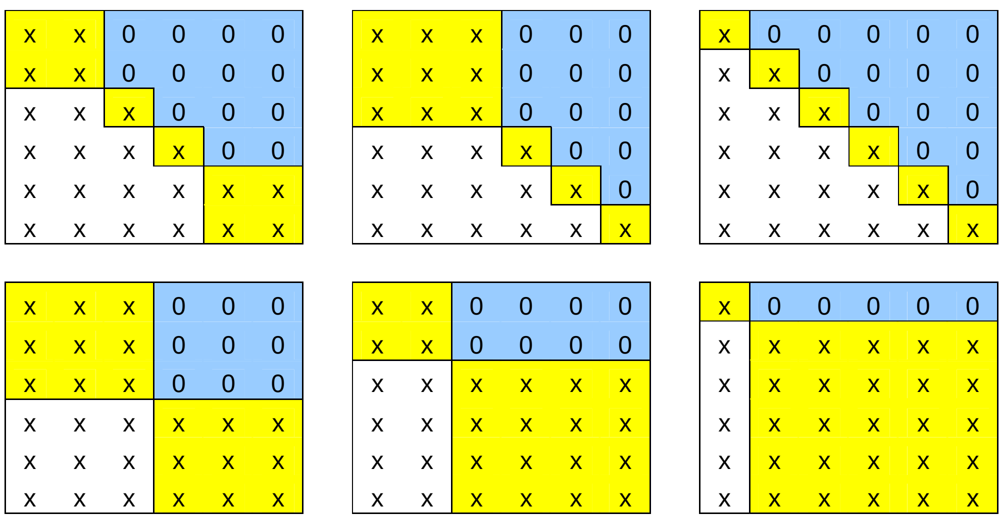
$$\mathbf{A} = \begin{bmatrix} -15 & 0 & -16 & 0 & 0 & 0 \\ 10 & 2 & 11 & 0 & 0 & 0 \\ 8 & 0 & 9 & 0 & 0 & 0 \\ 1 & 3 & 5 & 3 & 0 & -4 \\ 2 & 6 & 1 & 2 & 5 & 4 \\ -4 & 9 & -3 & -6 & -6 & -1 \end{bmatrix} \qquad \lambda = \begin{bmatrix} -7 \\ -1 \\ 1 \\ 2 \\ 3 \\ 5 \end{bmatrix}$$

$$\mathbf{A}_1 = \begin{bmatrix} -15 & 0 & -16 \\ 10 & 2 & 11 \\ 8 & 0 & 9 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} 3 & 0 & -4 \\ 2 & 5 & 4 \\ -6 & -6 & -1 \end{bmatrix} \qquad \lambda_1 = \begin{bmatrix} 1 \\ 2 \\ -7 \end{bmatrix} \quad \lambda_2 = \begin{bmatrix} -1 \\ 3 \\ 5 \end{bmatrix}$$

L'insieme degli autovalori della matrice (6 x 6) **A** è costituito dagli autovalori delle matrici (3 x 3) $\mathbf{A}_1$ [1 , 2 , -7] e $\mathbf{A}_2$ [-1 , 3 , 5].

Vari tipi di matrici a blocchi

Le forme finora viste sono solo un tipo di matrici a blocchi; in realtà i blocchi possono essere di diversa dimensione; al limite possono essere anche tutti di dimensione unitaria (matrice triangolare). Nella figura seguente sono mostrati alcuni esempi di matrici partizionate a blocchi (gialli)



Nota. Il risparmio di calcolo è tanto più efficace tanto più bassa è la massima dimensione dei blocchi della partizione. Nel primo caso la massima dimensione del blocco è 2, nel secondo è 3, nel terzo addirittura 1. E' evidente che il massimo guadagno si ha con la terza matrice. Per contro le ultime due forme di partizione danno un guadagno assai più limitato.

Matrici partizionabili

E' sempre possibile partizionare una matrice a blocchi? Sfortunatamente no. La possibilità di ridurre a blocchi una matrice dipende ovviamente dalla presenza di zeri ma anche dalla loro disposizione. Nascano due importanti problemi:

1. Riconoscere se una matrice è riducibile a blocchi
2. Ricavare la matrice di partizione P

Diversi metodi sono stati sviluppati in passato per risolvere questi problemi. Uno molto diffuso è il metodo del grafo

Grafo della matrice

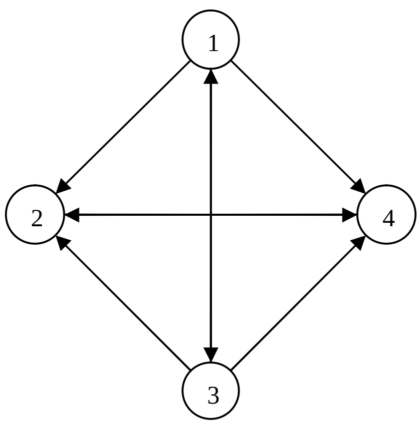
Secondo tale metodo si costruisce il grafo della matrice nel seguente modo

- Il numero dei nodi del grafo è uguale alla dimensione della matrice
- I nodi del grafo, numerati da 1 a n, rappresentano gli elementi della diagonale principale a_{ii}
- Per ogni elemento $a_{ij} \neq 0$ si traccia un cammino orientato dal nodo-i al nodo-j

Ad esempio, data la seguente matrice **A** (4 x 4):

4	2	3	1
0	-1	0	1
3	1	-1	2
0	1	0	1

Il grafo G(**A**) associato alla matrice sarà:



Il nodo 1 è collegato ai nodi 2, 3, 4.
Il nodo 2 è collegato al nodo 4
Il nodo 3 è collegato ai nodi 1, 2, 4
Il nodo 4 è collegato al nodo 2

Si osserva che dal nodo 2 non esiste nessun percorso che porti ai nodi 1 o 3
Similmente avviene se si parte dal nodo 4
Quindi il grafo non è fortemente connesso

Se partendo da ogni nodo è possibile trovare un qualunque percorso che attraversa tutti gli altri nodi allora si dice che il grafo è **fortemente connesso**.
Un importante teorema della teoria dei grafi afferma se il grafo G(**A**) è fortemente connesso la matrice associata **A** è irriducibile alla forma a blocchi. Viceversa la matrice è riducibile ed esiste senz'altro una matrice di partizione **P** che trasforma **A** alla forma a blocchi.
In sintesi

G(**A**) fortemente connesso $\Leftrightarrow$ matrice **A** irriducibile

G(**A**) non fortemente connesso $\Leftrightarrow$ matrice **A** riducibile a blocchi

Il metodo è elegante e molto importante dal punto di vista della teoria. Ma dal punto di vista pratico presenta diversi svantaggi:
- Diventa laborioso già per matrici dimensioni > 5
- Complicata traduzione in un programma
- Non fornisce la matrice di permutazione per la riduzione effettiva.

Nel nostro caso osserviamo che:

$\mathbf{B} = \mathbf{P}^T \mathbf{A} \mathbf{P}$, dove la matrice di permutazione $\mathbf{P} = [\, e_2, e_4, e_1, e_3 \,]$

A

4	2	3	1
0	-1	0	1
3	1	-1	2
0	1	0	1

P

0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0

$\mathbf{P}^T \mathbf{A} \mathbf{P}$

-1	1	0	0
1	1	0	0
2	1	4	3
1	2	3	-1

Ma per matrici di dimensioni superiori la ricerca delle permutazioni per tentativi diventa laboriosa. Ad esempio l'applicazione del metodo del grafo già alla seguente matrice (6 x 6) diventa molto complicato. E' praticamente inutilizzabile per matrici (10 x 10).

1	0	0	1	2	0
1	-1	1	2	-3	-2
-6	1	1	3	5	2
1	0	0	3	-1	0
2	0	0	5	1	0
-9	2	1	1	7	1

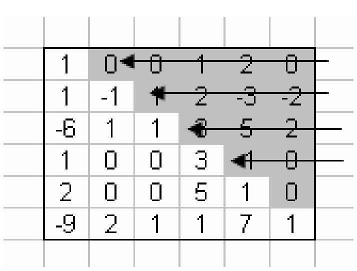
1	2	0	5	0	5	0	0
1	2	0	4	0	3	0	0
5	2	1	1	7	8	3	3
3	12	0	3	0	6	0	0
4	17	6	2	1	1	2	2
2	7	0	6	0	3	0	0
0	2	1	1	0	8	5	5
4	2	0	1	7	8	4	4

L'algoritmo a punteggio

(Ovvero un curioso algoritmo)
Presentiamo qui un metodo straordinariamente semplice ma molto efficace per la riduzione a blocchi delle matrici. Il metodo è applicabile anche a matrici di medie-grandi dimensioni. Si tratta di un algoritmo iterativo il cui scopo è quello di addensare gli zeri della matrice verso l'angolo alto-destro, mediante permutazioni righe-colonne.
Sebbene nato per essere implementato su calcolatore, questo metodo si presta ad essere eseguito anche a mano, almeno con matrici di piccole dimensioni.
Vediamo come lavora.

Prendiamo ad esempio la matrice 6 x 6 riportata sopra
Scriviamo a parte il vettore delle permutazioni iniziale

1	2	3	4	5	6
e_1	e_2	e_3	e_4	e_5	e_6



Lo scopo dell'algoritmo è annullare più elementi possibile nell'area grigia. Iniziamo a cercare tutti gli elementi sopra la diagonale principale diversi da zero (area grigia). La ricerca deve partire dalla prima riga e dall'ultima colonna: cioè dall'elemento a_{16} se zero si passa all'elemento a_{15} e cosi via fino all'elemento a_{12}. Poi si passa alla riga successiva, dall'elemento a_{26} all'elemento a_{23}. Poi alla riga successiva ecc..

	2			5	
1	0	0	1	2	0
1	-1	1	2	-3	-2
-6	1	1	3	5	2
1	0	0	3	-1	0
2	0	0	5	1	0
-9	2	1	1	7	1

Nel nostra caso, il primo elemento diverso da zero è a_{15}; cerchiamo, se esiste, il primo 0 sulla stessa riga a partire da sinistra. L'elemento è a_{12}. Dovremmo pertanto permutare le colonne 2 e 5 e consecutivamente le righe 2 e 5

Dopo la permutazione 2, 5 la matrice è la seguente

A

1	0	0	1	2	0
1	-1	1	2	-3	-2
-6	1	1	3	5	2
1	0	0	3	-1	0
2	0	0	5	1	0
-9	2	1	1	7	1

P

1	5	3	4	2	6
1	0	0	0	0	0
0	0	0	0	1	0
0	0	1	0	0	0
0	0	0	1	0	0
0	1	0	0	0	0
0	0	0	0	0	1

P^TA P

1	2	0	1	0	0
2	1	0	5	0	0
-6	5	1	3	1	2
1	-1	0	3	0	0
1	-3	1	2	-1	-2
-9	7	1	1	2	1

Osserviamo già l'addensamento degli zeri nella parte alta a destra della matrice.

		3	4		
1	2	0	1	0	0
2	1	0	5	0	0
-6	5	1	3	1	2
1	-1	0	3	0	0
1	-3	1	2	-1	-2
-9	7	1	1	2	1

Ora osserviamo che il primo elemento diverso da zero a partire da destra è a_{14}. Il primo 0 a partire da sinistra è a_{13}. quindi permutiamo 3 e 4

Dopo la permutazione 3, 4 la situazione è la seguente

A

1	2	0	1	0	0
2	1	0	5	0	0
-6	5	1	3	1	2
1	-1	0	3	0	0
1	-3	1	2	-1	-2
-9	7	1	1	2	1

P

1	2	4	3	5	6
1	0	0	0	0	0
0	1	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1

P^TA P

1	2	1	0	0	0
2	1	5	0	0	0
1	-1	3	0	0	0
-6	5	3	1	1	2
1	-3	2	1	-1	-2
-9	7	1	1	2	1

Tutti gli zeri sono stati riposizionati. La matrice è partizionata e l'algoritmo si ferma.

La matrice di partizione finale è pertanto

1	2	3	4	5	6
e1	e5	e4	e3	e2	e6

Come si vede con soli 2 scambi si è riusciti a partizionare agevolmente e velocemente una matrice (6 x 6). Il metodo conserva una buona efficenza anche per matrici molto più grandi
Vediamo un altro esempio

Ridurre a blocchi (se possibile) la seguente matrice

⇓			⇓		
3	1	-1	1	-5	2
0	-1	0	1	0	0
5	1	1	2	-3	4
0	0	0	1	0	0
1	1	7	-9	13	1
0	1	0	-6	0	1

Il primo elemento $\neq 0$, da destra, è a_{16}
il primo elemento $= 0$, da sinistra è a_{21}.
Quindi le colonne pivot di permutazione sono 1 e 6.

		⇓	⇓		
1	1	0	-6	0	0
0	-1	0	1	0	0
4	1	1	2	-3	5
0	0	0	1	0	0
1	1	7	-9	13	1
2	1	-1	1	-5	3

Il primo elemento $\neq 0$, da destra, è a_{14}
il primo elemento $= 0$, da sinistra è a_{13}.
Quindi le colonne pivot di permutazione sono 3 e 4.

(		(			
1	1	-6	0	0	0
0	-1	1	0	0	0
0	0	1	0	0	0
4	1	2	1	-3	5
1	1	-9	7	13	1
2	1	1	-1	-5	3

Il primo elemento (0, da destra, è a13
il primo elemento (0, da sinistra è a21.
Quindi le colonne pivot di permutazione sono 1 e 3.

Finalmente abbiamo la matrice finale

1	0	0	0	0	0
1	-1	0	0	0	0
-6	1	1	0	0	0
2	1	4	1	-3	5
-9	1	1	7	13	1
1	1	2	-1	-5	3

la matrice è stata partizionata a blocchi: 3 blocchi da 1x1 e un blocco da (3 x 3)

Si deve osservare che l'algoritmo non fornisce informazioni sul successo o meno del processo. Semplicemente si arresta quando non ci sono più elementi da permutare. Se la matrice finale risulta partizionata allora era significa che è riducibile, altrimenti la matrice è irriducibile e il suo grafo è fortemente connesso.

Funzione punteggio (Score-Function)

Le matrici finora usate come test avevano tutti gli zeri usati per la partizione. Vediamo cosa succede se la matrice ha altri zeri al di fuori di quelli strettamente necessari per la partizione (zeri spuri) . In tal caso non tutte le permutazioni saranno utili per l'addensamento degli zeri. Alcune di queste permutazioni avranno persino la tendenza opposta .
Quindi è necessario disporre di un metro di valutazione delle permutazioni . Visivamente è facile individuare le permutazioni "utili" da quelle "inutili". Dal punto di vista del calcolo numerico, invece, è necessario trovare una funzione per valutare numericamente l'efficacia delle permutazioni: la funzione "punteggio" è quella che viene adottata nel presente algoritmo

Tale funzione conta gli zeri del triangolo superiore prima (matrice A) e dopo la permutazione (matrice B) ritornando la differenza.

$$score = \sum_B w(i,j) - \sum_A w(i,j)$$

Il punteggio sarà positivo se la permutazione è vantaggiosa altrimenti sarà negativo o nullo

x					
x	x				
x	x	x			
x	x	x	x		
x	x	x	x	x	
x	x	x	x	x	x

Gli zeri non hanno tutti lo stesso peso: gli zeri più vicini all'angolo in alto a destra hanno peso maggiore. Quindi le configurazioni con zeri addensati verso l'angolo sono migliori di quelle con zeri verso la diagonale.

x	x	0	x	0	0
x	x	x	x	0	0
x	x	x	x	x	x
x	x	x	x	x	x
x	x	x	x	x	x
x	x	x	x	x	x

migliore

x	x	x	x	x	x
x	x	0	x	x	x
x	x	x	0	x	x
x	x	x	x	0	x
x	x	x	x	x	0
x	x	x	x	x	x

peggiore

La funzione "peso" è arbitrara. Una funzione che ha dato buoni risultati è la seguente.

$$w(i,j) = \begin{cases} 0 \quad \Leftrightarrow \ \mathrm{a}_{ij} \neq 0 \\ (n-i+1)^2 \cdot j^2 \quad \Leftrightarrow \ \mathrm{a}_{ij} = 0 \end{cases}$$

Funzione peso per matrice (n x n)
.

Per ogni permutazione individuata, l'algoritmo effettua la misura del punteggio; se positivo la permutazione viene eseguita, altrimenti viene scartata e passa alla ricerca di una nuova permuatazione. Dopo vari cicli di permutazioni la configurazione degli zeri raggiunge il massimo punteggio possibile; ogni nuovo tentativo produrrà solo punteggi negativi o nulli per cui l'algoritmo si ferma.

Esempi

Vediamo ora l'algoritmo al lavoro in casi pratici.

A

1	2	0	2	0	0
0	1	2	0	-3	0
0	0	1	0	5	3
0	3	1	1	0	0
0	0	0	0	1	3
0	0	0	0	1	3

$P^T A P$

1	3	0	0	0	0
1	3	0	0	0	0
5	3	1	0	0	0
-3	0	2	1	0	0
0	0	1	3	1	0
0	0	0	2	2	1

P = [e5, e6, e3, e2, e4, e1]
Permutazioni accettate = 6
Permutazioni scartate = 4

A

3	0	0	0	0	0	2	3	0	4
6	1	6	3	0	2	5	1	0	2
0	0	1	0	0	0	1	0	0	0
8	1	8	1	0	0	7	1	0	0
10	1	10	5	0	0	9	1	5	0
0	1	7	4	0	1	6	1	0	3
0	0	2	0	0	0	1	0	0	0
4	0	0	0	0	0	3	1	0	6
9	1	9	4	-1	3	0	1	1	5
5	0	5	0	0	0	0	0	0	1

$P^T A P$

1	2	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
0	5	1	5	0	0	0	0	0	0
2	0	4	3	3	0	0	0	0	0
3	0	6	4	1	0	0	0	0	0
5	6	2	6	1	1	3	2	0	0
7	8	0	8	1	1	1	0	0	0
6	7	3	0	1	1	4	1	0	0
0	9	5	9	1	1	4	3	1	-1
9	10	0	10	1	1	5	0	5	0

P = [e7, e3, e10, e1, e8, e2, e4, e6, e9, e5]
Permutazioni accettate = 9
Permutazioni scartate = 10

A

1	0	1	0	1	0	1	6	0	1
1	1	0	1	1	1	1	1	1	0
0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	4	1	1	0	1
0	0	1	0	1	0	5	0	0	0
1	0	1	0	0	1	1	1	0	0
0	0	1	0	0	0	1	0	0	0
0	0	1	0	4	0	1	3	0	0
1	0	1	3	0	4	1	1	1	1
0	0	0	0	0	0	1	4	0	1

$P^T A P$

1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	5	1	0	0	0	0	0	0	0
1	1	4	3	0	0	0	0	0	0
0	1	0	4	1	0	0	0	0	0
1	1	1	6	1	1	0	0	0	0
1	1	0	1	0	1	1	0	0	0
0	1	1	1	1	1	4	1	0	0
1	1	0	1	1	1	4	3	1	0
0	1	1	1	0	1	1	1	1	1

P = [e3, e7, e5, e8, e10, e1, e6, e4, e9, e2]
Permutazioni accettate = 7
Permutazioni scartate = 1

A

3	0	8	0	0	3	0	3	0	0	0	6	0	0	0	14	8	0	7	0
4	4	0	0	0	6	0	6	0	0	3	9	0	0	0	20	0	0	10	4
0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	2	0
0	0	17	10	10	0	10	0	10	0	0	15	0	10	10	0	17	10	16	0
4	9	16	9	9	11	9	11	9	9	8	14	9	9	9	30	0	9	15	9
0	0	0	0	0	1	0	0	0	0	0	4	0	0	0	10	0	0	20	0
0	0	20	20	0	0	13	20	0	20	12	0	0	13	13	38	20	13	0	13
0	0	0	0	0	2	0	2	0	0	0	20	0	0	0	0	7	0	6	0
4	11	18	0	20	13	11	13	11	11	10	16	11	11	11	34	18	0	17	11
20	5	0	0	0	7	0	0	0	5	0	0	0	0	0	0	1	0	11	5
4	0	9	0	0	0	0	4	0	0	1	0	0	0	0	20	0	0	8	0
0	0	4	0	0	0	0	0	0	0	0	2	0	0	0	0	4	0	3	0
4	6	13	0	0	8	0	8	0	6	5	11	6	0	0	0	0	0	12	6
0	7	14	0	0	9	0	9	0	7	20	12	7	7	0	0	0	0	13	0
4	0	19	12	12	14	12	14	12	0	0	17	0	12	12	36	19	12	18	0
0	0	5	0	0	0	0	0	0	0	0	3	0	0	0	8	5	0	4	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	8	15	0	0	10	0	10	0	8	7	13	8	8	0	0	0	8	14	8
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0
4	0	10	0	0	5	0	5	0	0	2	8	0	0	0	18	0	0	0	3

$P^T A P$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4	5	3	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	20	0	4	10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	6	0	20	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0
8	7	8	6	14	3	3	3	0	0	0	0	0	0	0	0	0	0	0	0
0	8	9	0	20	0	4	4	1	0	0	0	0	0	0	0	0	0	0	0
0	0	10	8	18	5	5	4	2	3	0	0	0	0	0	0	0	0	0	0
0	10	0	9	20	6	6	4	3	4	4	0	0	0	0	0	0	0	0	0
1	11	0	0	0	7	0	20	0	5	5	5	0	0	0	0	0	0	0	0
0	12	13	11	0	8	8	4	5	6	6	6	6	0	0	0	0	0	0	0
0	13	14	12	0	9	9	0	20	0	7	7	7	7	0	0	0	0	0	0
0	14	15	13	0	10	10	4	7	8	8	8	8	8	8	0	0	0	0	0
19	18	19	17	36	14	14	4	0	0	0	0	0	12	12	12	12	12	12	12
17	16	17	15	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10
20	0	20	0	38	0	20	0	12	13	0	20	0	13	13	13	20	13	0	0
18	17	18	16	34	13	13	4	10	11	11	11	11	11	0	11	0	11	11	20
0	15	16	14	30	11	11	4	8	9	9	9	9	9	9	9	9	9	9	9

P = [e17, e19, e3, e12, e16, e6, e8, e1, e11, e20, e2, e10, e13, e14, e18, e15, e4, e7, e9, e5]
Permutazioni accettate = 18
Permutazioni scartate = 237

Come si vede, anche per matrici di medie dimesioni il numero delle permutazioni rimane sempre molto contenuto. Tenendo conto di questo e del fatto che l'operazione di scambio è assai più veloce di ogni operazione aritmetica in virgola mobile, si intuisce l'elevata rapidità di fattorizzazione di questo algoritmo.
Tutte le matrici illustrate sono state ottenute per mezzo di Excel con l'addin Matrix.xla v 1.6

Leonardo Volpi