Algoritmi per operazioni con le matrici

Sommario

- **Definizioni**
- **Alcune operazioni principali sulle matrici**
  - Somma di due matrici
  - Trasposta di una matrice
  - Prodotto di matrici: algoritmo classico
  - Prodotto di matrici: algoritmo di Strassen
  - Soluzione di sistemi di equazioni lineari: Algoritmo di eliminazione di Gauss

Definizioni

- Una matrice è un array rettangolare di numeri

$$A\ (3X4) = \begin{pmatrix} 1 & 1 & 1 & 3 \\ 2 & 1 & -1 & 0 \\ 1 & 2 & -1 & -1 \end{pmatrix}$$

- La trasposta di una matrice A è la matrice A^T ottenuta scambiando le righe con le colonne di A

$$A^T\ (4X3) = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \\ 1 & -1 & -1 \\ 3 & 0 & 1 \end{pmatrix}$$

- Una matrice simmetrica soddisfa la condizione $A = A^T$
- Una matrice triangolare superiore U è tale che $u_{ij}=0$ per $i>j$ (tutti gli elementi sotto la diagonale principale sono nulli).
- Rango di una matrice: massimo numero di righe linearmente indipendenti
- Matrice singolare: con determinate nullo
- Una matrice ha rango massimo se e solo se è non singolare

Operazioni sulle matrici

- **Somma di matrici: associativa e commutativa**
  - La dimensione delle matrici deve essere compatibile
- **Sottrazione di matrici: A-B= A+(-B)**
  - La dimensione delle matrici deve essere compatibile
- **Trasposizione di matrici: A -> A^T**
- **Prodotto di matrici: associativo ma NON commutativo**
  - A(BC)= (AB)C
  - AB != BA
  - Dimensioni compatibili: il numero di colonne di A deve essere uguale al numero di righe di B.
- **Determinante di una matrice quadrata**

Operazioni sulle matrici: somma

- Input: A(nxm), B(nxm)
- Output C(nxm)
- Algoritmo

```
for i = 1 to n
    for j = 1 to m do
        C[i,j] <- A[i,j] + B[i,j]
```

Operazioni sulle matrici: trasposta

- **Input: A(nxm)**
- **Output: B(mxn)**
- **Algoritmo:**

```
for i = 1 to n
    for j = 1 to m do
        B[j,i] <- A[i,j]
```

- **L'algoritmo va bene anche se vogliamo modificare la matrice A, senza crearne una nuova?**
- **Input: A(nxm)**
- **Output: A^T (mxn)**

```
for i = 1 to n
    for j = 1 to m do
        A[j,i] <- A[i,j]
```

- **Esercizio: modificare il secondo algoritmo in modo che sia corretto**

Operazioni sulle matrici: prodotto

- **Input: A(nxm), B(mxh)**
- **Output C(nxh)**
- **Algoritmo classico: moltiplicazione righe per colonne**

```
for i=1,n
    for j=1,h
            RIS =0;
            for k=1,m
                RIS = RIS+A[i,k]*B[k,j];
            C[i,j]=RIS;
```

- **Complessità asintotica n'=max(n,m,h) O(n'³)**
- **Esempio**

A (2X3) = $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix}$ **B (3X4) =** $\begin{pmatrix} 1 & 2 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 \end{pmatrix}$ **C (2X4) =** $\begin{pmatrix} 1+2+2 & 2+2+0 & 0+0+3 & 1+0+0 \\ 0+1+6 & 0+1+0 & 0+0+9 & 0+0+0 \end{pmatrix}$

Algoritmo di Strassen

- **Complessità asintotica O($n^{2,81}$);**
- **Basato sul paradigma divide et impera;**
- **Si applica a matrici *A,B,C* di dimensione *n*x*n* con $n=2^m$**
- **Idea: *A,B* e *C* vengono divise in *4* sottomatrici:**

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

- **Che corrispondono a:**

  $r = ae + bg$

  $s = af + bh$

  $t = ce + dg$

  $u = cf + dh$

- **Ogni equazione contiene 2 prodotti di matrici di dimensione *n/2 x n/2 e le relative 4 somme O(n^2)***
- ***Utilizzando il paradigma divide et impera si ottiene la ricorrenza***

$$T(n) = 8\,T(n/2) + O(n^2)$$

La soluzione della ricorrenza è O(n^3)

…Algoritmo di Strassen

- **L'approccio di Strassen richiede solo 7 moltiplicazioni ricorsive di matrici *n/2 x n/2* e 18 tra addizioni e sottrazioni: *O(n²)***

$$T(n) = 7\,T(n/2) + O(n^2)$$

la cui soluzione è $O(n^{\log_2 7}) \sim O(n^{2.81})$

- **Perché solo 7 prodotti?**
- **Gli 8 prodotti possono essere ricavati solo mediante somme e sottrazioni dei 7 seguenti sottoprodotti:**

  P1 = (b-d)(g+h)
  P2 = (a+d)(e+h)
  P3 = (a+c)(e+f)
  P4 = (a+b)h
  P5 = a(f-h)
  P6 = d(g-e)
  P7 = (c+d)e

- **Inoltre:**

  r = P1 + P2 – P4 +P6
  s = P4 + P5
  t = P6 + P7
  u = P2 – P3 + P5 – P7

- **Esempio:**

  s = P4 + P5 = (a+b)h + a(f-h) = ah + bh + af – ah = bh + af

Algoritmo di Strassen: considerazioni

- **I fattori costanti sono più alti dei fattori costanti dell'algoritmo classico con tempo *$O(n^3)$*.**
- **Non è efficieinte per la moltiplicazione di matrici sparse.**
- **L'algoritmo si usa quando le dimensioni delle matrici sono molto grandi.**
- **Quando la dimensione dei sottoproblemi è *relativamente piccola* si usa il metodo classico.**

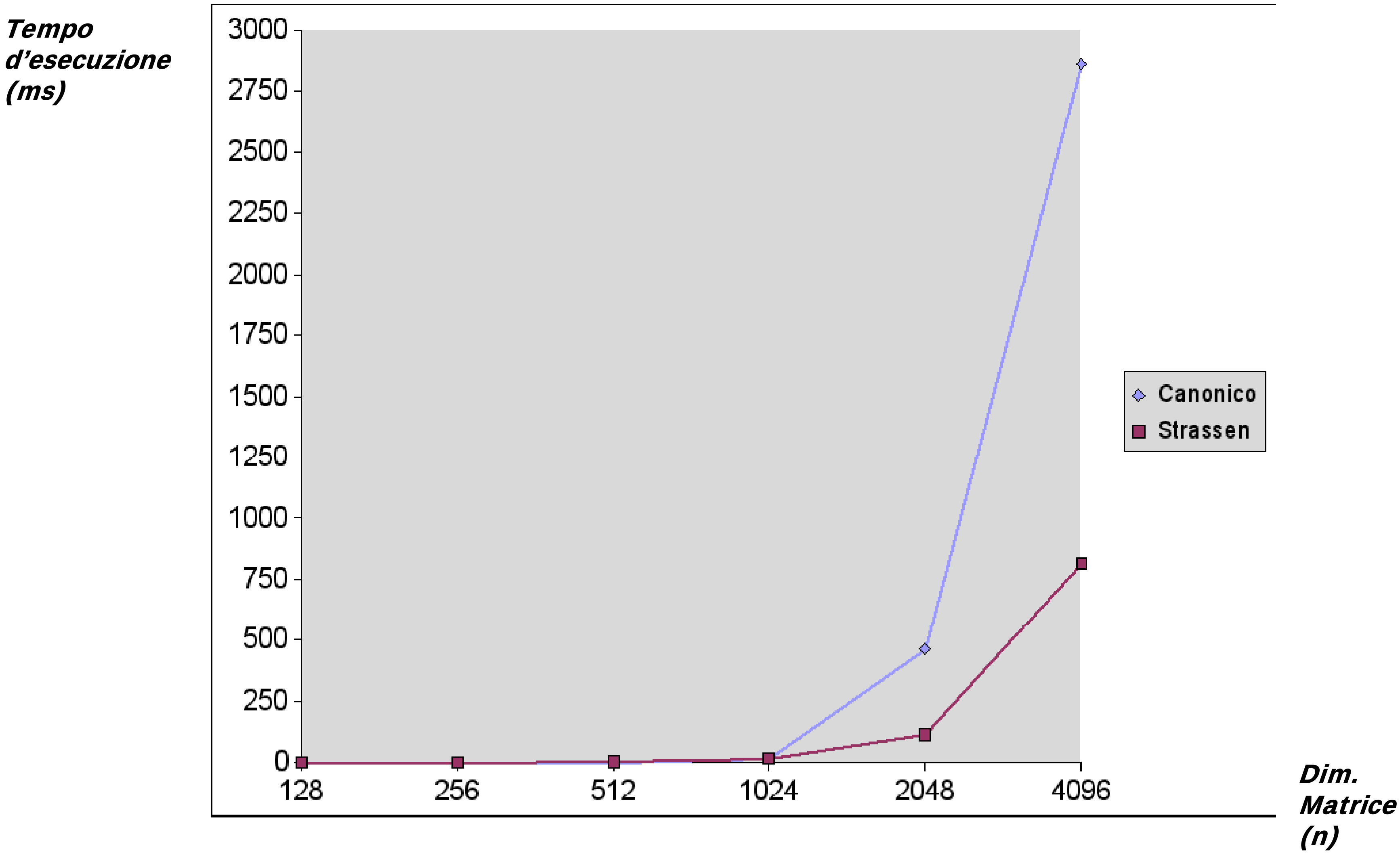
Ci sono algoritmi più efficienti

I limiti della complessità computazionale finora conosciuti sono $O(n^{2,376})$ e $\Omega(n^2)$

Non si conosce un limite asintotico esatto

per il problema della moltiplicazione

Algoritmo di Strassen: tempi d'esecuzione



Risoluzione di un sistema di equazioni lineari

- **Sistema: Ax = b**
- **A: matrice nxn**
- **B: array unidimensionale**
- **esempio:**

$$\begin{cases} x + y + z = 3 \\ 2x + y - z = 0 \\ x + 2y - z = -1 \end{cases}$$

A			B
1	1	1	3
2	1	-1	0
1	2	-1	-1

Metodo di eliminazione di Gauss per la triangolarizzazione di una matrice

Gaussian Elimination

- *Operando trasformazioni lineari il sistema non cambia*
  - **Scambio di due righe**
  - **Moltiplicazione di una riga per un coefficiente**
  - **Somma o sottrazione di una riga per un'altra**
- **Si operano opportune trasformazioni lineari della matrice e del relativo array dei termini noti in modo che la matrice diventi triangolare superiore**
- **L'ultima riga ha solo un'incognita, la penultima 2 etc…**
- **Si opera una sostituzione all'indietro trovando tutte le soluzioni**
- **Esempio: la seconda riga viene trasformata moltiplicando la prima riga per 2 e sottraendola alla seconda; la terza riga è ottenuta dalla sottrazione tra la prima e la terza.**

1	1	1	3	1	1	1	3
2	1	-1	0	0	1	3	6
1	2	-1	-1	0	0	1	2

Triangolarizzazione della matrice

- **Si trasforma la matrice in una matrice triangolare superiore**

$$\begin{array}{c} 2 \\ \\ \\ \end{array} \begin{array}{cccc} 1 & 1 & 1 & 3 \\ 2 & 1 & -1 & 0 \\ 1 & 2 & -1 & -1 \end{array}$$

moltiplico la prima riga per 2 e la sottraggo alla seconda

$$\begin{array}{cccc} 1 & 1 & 1 & 3 \\ 0 & 1 & 3 & 6 \\ 1 & 2 & -1 & -1 \end{array}$$

sottraggo la prima riga alla terza

$$\begin{array}{cccc} 1 & 1 & 1 & 3 \\ 0 & 1 & 3 & 6 \\ 0 & -1 & 2 & 4 \end{array}$$

moltiplico la seconda riga per -1

sottraggo la seconda riga alla terza

$$\begin{array}{cccc} 1 & 1 & 1 & 3 \\ 0 & 1 & 3 & 6 \\ 0 & 0 & -5 & -10 \end{array}$$

moltiplico la terza riga per -1/5

$$\begin{array}{c} \\ \\ -1/5 \end{array} \begin{array}{cccc} 1 & 1 & 1 & 3 \\ 0 & 1 & 3 & 6 \\ 0 & 0 & 1 & 2 \end{array}$$

x=1

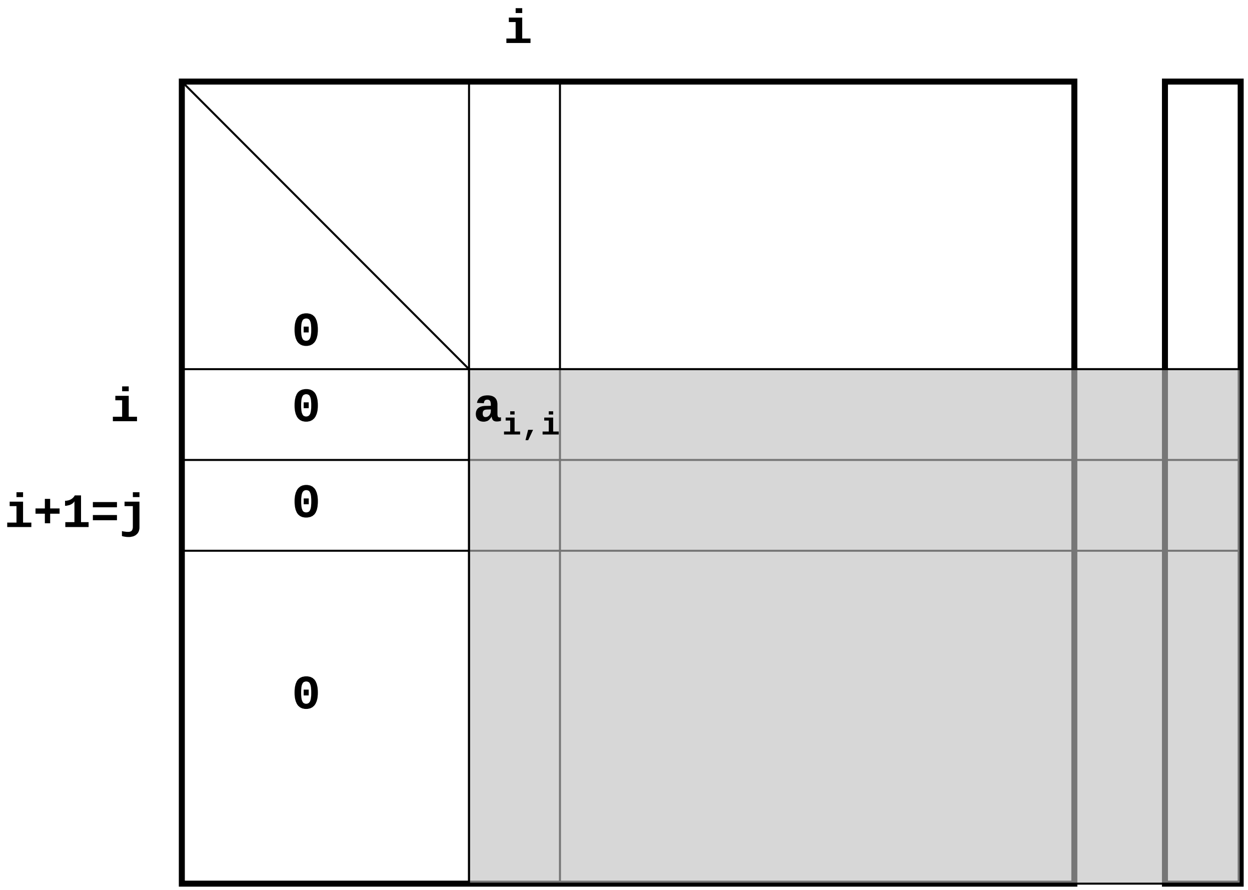
y=0

z = 2

Metodo di Gauss: pseudocodifica

```
A: matrice quadrata di dimensione NxN;
B: array di dimensione N;
;; eliminazione della i-esima colonna
for i <- 1,N-1
   for j <- i+1,N ;;j e' l'indice di riga successivo a i
        molt <- A[j,i]/A[i,i] ;; calcolo del coeff. moltiplicativo
        ;; semplificazione delle k colonne della j-esima riga
        for k <- i,N
               A[j,k] <- molt * A[i,k] - A[j,k]
        B[j] <- molt * B[i] – B[j]
```

Evoluzione dei calcoli



Problemi di stabilità numerica

- **Cosa succede quando il coefficiente moltiplicativo**
  molt = A[j,i]/A[i,i] è un numero "molto" grande?
  A[j,i]>>A[i,i]
- **Si possono verificare errori di approssimazione, visto che l'insieme dei numeri in virgola mobile non è uniformemente distribuito rispetto all'asse reale, ma i valori floating-point sono densi intorno allo zero e sempre più sparsi verso gli estremi dell'intervallo di definizione.**

Inoltre

Sistemi mal condizionati possono dar luogo a problemi di instabilità numerica

Problemi mal condizionati

- Un problema si dice **mal condizionato** quando, variando di poco i dati del problema, i risultati possono variare di molto.
- Un tipico problema mal condizionato e'

$$\begin{cases} x + y = 10 \\ 1.002x + y = 0 \end{cases} \quad \textit{le cui soluzioni sono} \quad \begin{cases} x = -5000 \\ y = 5010 \end{cases}$$

- Se il coefficiente della x diventa 1.001 le soluzioni diventano $x=-10000$ $y= 10010$
- Un esempio di problema ben condizionato:

$$\begin{cases} x + y = 10 \\ 1.002\,x - y = 0 \end{cases} \quad \textit{le cui soluzioni sono} \quad \begin{cases} x = -5000 \\ y = 5010 \end{cases}$$

Il malcondizionamento è intrinseco al problema

Associato al mal condizionamento c' è il problema della stabilità

Algoritmi stabili

- Un procedimento numerico o algoritmo si dice (numericamente) stabile quando, al crescere del numero dei passi, l'errore si mantiene limitato. Se questo non succede l'algoritmo si dirà (numericamente) instabile.
- L'instabilità numerica è una caratteristica dell'algoritmo,
- Possono esistere procedimenti stabili ed altri instabili per la soluzione dello stesso problema.

- Come rendere stabile l'algoritmo di eliminazione di Gauss?
- Facendo in modo che il coefficiente moltiplicativo sia sempre un valore minore di 1.
  - Si calcola il massimo sulla colonna i-esima per le righe dalla i-esima alla n-esima. Sia m la riga in cui si trova il massimo.
  - Si scambia la riga m con la riga i e si procede al calcolo.
- Essendo il denominatore il più grande di tutti i valori sulla colonna, il coefficiente moltiplicativo sarà sempre minore di 1.

Sostituzione all’indietro

Back substitution

```
X: array delle soluzioni
;;calcolo l’ultima soluzione
  X[N] <- B[N]/A[N,N]
  for i <- N-1,1,-1
  ;;termine noto
  QUOT=B[i]
      ;; porto tutti i valori noti “a destra” dell’uguaglianza
      for j <- i+1,N
            QUOT <- QUOT – A[i,j] * X[j]
    ;;divido per il coeff. dell’incognita da calcolare
     X[i] <- QUOT/A[i,i]
```

Come verificare che l'algoritmo sia corretto

- Il sistema originario deve essere conservato
- Le soluzioni vengono sostituite nel sistema originario
- esempio:

```
2 3 1 4 3 0
0 3 8 2 3 9
9 1 0 3 2 1
1 2 9 4 5 6
2 3 0 0 2 1
COEFFICIENTI DEL SISTEMA: dimensione 5
2.000000 3.000000 1.000000 4.000000 3.000000 0.0000000E+00
0.0000000E+00 3.000000 8.000000 2.000000 3.000000 9.000000
9.000000 1.000000 0.0000000E+00 3.000000 2.000000 1.000000
1.000000 2.000000 9.000000 4.000000 5.000000 6.000000
2.000000 3.000000 0.0000000E+00 0.0000000E+00 2.000000 1.000000
...............................................
MATRICE TRIANGOLARIZZATA
2.000000       3.000000       1.000000       4.000000       3.000000   0.0000000E+00
0.0000000E+00  -3.000000      -8.000000      -2.000000      -3.000000  -9.000000
0.0000000E+00  0.0000000E+00   28.83333      -6.666667      1.000000   8.50000
0.0000000E+00  0.0000000E+00   0.0000000E+00 -3.323699      -2.751445  5.069365
0.0000000E+00  0.0000000E+00   0.0000000E+00 0.0000000E+00  2.537391   -4.118261
```

```
SOLUZIONI
0.4050715 1.145305 1.349554 -0.1816313 -1.623030
VERIFICA SULLA RIGA 1 SBAGLIATA
VERIFICA SULLA RIGA 2 CORRETTA
VERIFICA SULLA RIGA 3 SBAGLIATA
VERIFICA SULLA RIGA 4 SBAGLIATA
VERIFICA SULLA RIGA 5 SBAGLIATA
...............................................

con DOUBLE PRECISION
...........
SOLUZIONI
0.405071967100754 1.14530500342700 1.34955448937629
-0.181631254283756 -1.62302947224126
VERIFICA SULLA RIGA 1 CORRETTA
VERIFICA SULLA RIGA 2 CORRETTA
VERIFICA SULLA RIGA 3 CORRETTA
VERIFICA SULLA RIGA 4 CORRETTA
VERIFICA SULLA RIGA 5 CORRETTA
```