systems of linear equations: special matrices

Dice che qua andrà veloce.
Sono solo richiami.

Qui ci elenca solo qualche proprietà dei sistemi di equazioni lineari.
Una matrice simmetrica definita positiva è un matr = alla sua trasposta. Definita positiva vuol dire che la matrice * prod scalare con un vettore non nullo otteniamo sempre un valore positivo.

✓ symmetric, positive definite matrices

$$A = A^T$$

$$O\left(\frac{n^3}{6}\right)$$

Cholesky's
algorithm

$$x^T A x > 0 \,, \quad \forall x \neq \mathbf{0}$$

L'algoritmo di Cholesky's è l'algoritmo di gauss ma ottimizzato per le matrice simmetrice e la sua complessità è la metà di quella di guass.

Per la risoluzione con matrici tridiagonali (cioè ha solo la diag pricnipale, sotto e sopra.). Usiamo l'algoritmo tipo di thomas che ha complessità lineare 5N.

✓ tridiagonal matrices

$$O(5n)$$

Thomas
algorithm

queste due definizione le ho scritte prime.

symmetric matrix

$$A = A^T$$

positive definite matrix

$$x \neq \mathbf{0} \;\Rightarrow\; x^T A x > 0$$

Aggiunge che in una matrice definita positiva gli autovalori sono positivi e i determinati delle sottomatrici principali di A sono positivi

eigenvalues of A are positive

determinants of the principal submatrices of A are positive

$$A = LL^T$$

Qui non vuole entrare nei dettagli dell'algirtomo di Chilesky, ma dice solo che mentre gaus ha a. che fare con la fattorizzazione LU, qui si usa la fattorizzazione L*L, trasposto, cioè a si fattorizza in un prodotto di una matrice per la trasposta della stessa matrice.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = LL^T$$

$$= \begin{pmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{pmatrix} \begin{pmatrix} \ell_{11} & \ell_{21} & \ell_{13} \\ 0 & \ell_{22} & \ell_{23} \\ 0 & 0 & \ell_{33} \end{pmatrix}$$

$$A = LL^T$$

Non lo spiega E NON LO CHIEDE COME SI FATTORIZZA. E QUINDI COME SI OTTIENE A.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = LL^T$$

$$= \begin{pmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{pmatrix} \begin{pmatrix} \ell_{11} & \ell_{21} & \ell_{13} \\ 0 & \ell_{22} & \ell_{23} \\ 0 & 0 & \ell_{33} \end{pmatrix}$$

Cholesky's factorization algorithm (column-wise)

NON LO CHIEDE

$$a_{11} = \ell_{11}^2 \quad \Rightarrow \ell_{11} = \sqrt{a_{11}}$$

$$a_{21} = \ell_{21}\ell_{11} \quad \Rightarrow \ell_{21} = a_{21}/\ell_{11}$$

$$a_{31} = \ell_{31}\ell_{11} \quad \Rightarrow \ell_{31} = a_{31}/\ell_{11}$$

$$a_{22} = \ell_{21}^2\ell_{22}^2 \quad \Rightarrow \ell_{22} = \sqrt{a_{22} - \ell_{21}^2}$$

$$a_{32} = \ell_{31}\ell_{21} - \ell_{32}\ell_{22} \quad \Rightarrow \ell_{32} = \left(a_{32} - \ell_{31}\ell_{21}\right)/\ell_{22}$$

....

$$A = LL^T$$

Ci ricorda solo che invece di a METTIAMO LLTx = b poniamo Ltx = p
otteniamo quindi Lp = b, lo risolviamo e noto p possiamo calcolare x risolvendo Ltx = p. Quelli in rosso sono le incognite.

$$Ax = b$$

$$L\boxed{L^T x} = b$$

posed $L^T x = p$

$$\Downarrow$$

$$Lp = b$$

$$L^T x = p$$

$$A = LL^T$$

Credo che sia autoesplicativa, non ne parla nella lezione.

note that if we set

$$R = L^T$$

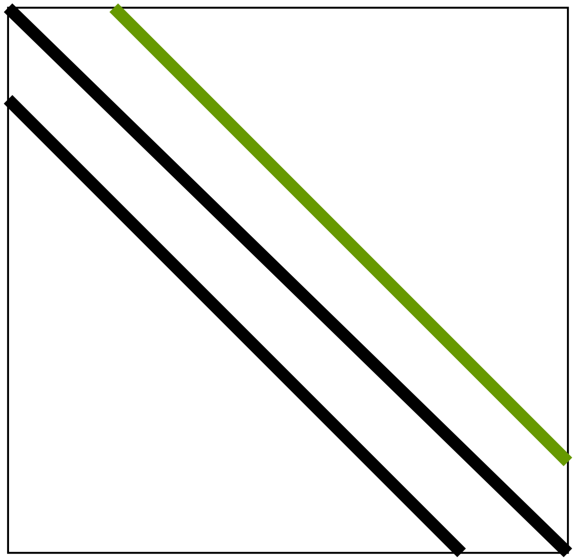
then we get the alternative Cholesky factorization

$$A = R^T R$$

with R **upper** triangular

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

tridiagonal matrix



Qui è matlab.. .cioè per generarle.

```
B = tridiag(7,ones(7,1),2*ones(7,1),3*ones(7,1))

B =

     1     2     0     0     0     0     0
     3     1     2     0     0     0     0
     0     3     1     2     0     0     0
     0     0     3     1     2     0     0
     0     0     0     3     1     2     0
     0     0     0     0     3     1     2
     0     0     0     0     0     3     1
```

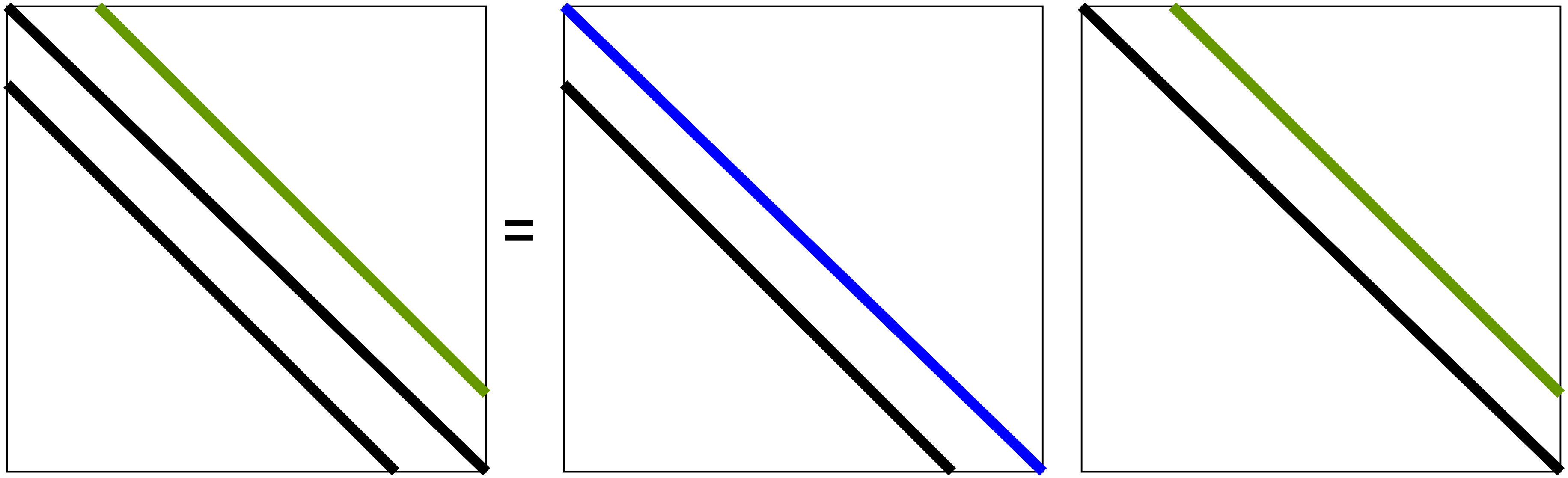
tridiagonal matrix

LU factorization

$$T = LU$$

Queste matrici tridiagonali possono essere fattorizzare nel prodotto di due matririci bidiagonali.
E infatti thoms è l'algoritmo che risolve il sistema di matrici trdiagonali.

NON MOSTRA L'ALGORITMO. Quindi devo saltare un po di slide.



tridiagonal matrix

$T = LU$

$$T = \begin{pmatrix} d_1 & f_1 & 0 & 0 \\ c_1 & d_2 & \ddots & \ddots \\ \ddots & \ddots & \ddots & f_{n-1} \\ 0 & 0 & c_{n-1} & d_n \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \ell_1 & 1 & \ddots & \ddots \\ \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ell_{n-1} & 1 \end{pmatrix} \qquad U = \begin{pmatrix} u_1 & f_1 & 0 & 0 \\ 0 & u_2 & \ddots & \ddots \\ \ddots & \ddots & \ddots & f_{n-1} \\ 0 & 0 & 0 & u_n \end{pmatrix}$$

tridiagonal matrix

$T = LU$

$$T = \begin{pmatrix} d_1 & f_1 & 0 & 0 \\ c_1 & d_2 & \ddots & \ddots \\ \ddots & \ddots & \ddots & f_{n-1} \\ 0 & 0 & c_{n-1} & d_n \end{pmatrix}$$

at step **`k`** compute

- ✓ the **`k`**-th component of the subdiagonal of L
- ✓ the **`k`**-th component of the diagonal of U

Thomas factorization algorithm

not in place

```
function [linf,udiag] = ...
          Fatt_LU_tridia(c,d,f)
linf=c; udiag=d; n=length(d);
for j=1:n-1
   linf(j)=c(j)/udiag(j);
   udiag(j+1)= udiag(j+1)-linf(j)*f(j)
end
```

$O(2n)$

solver of bidiagonal ststems

tridisolve(c,d,f,b)

BUT
a tridiagonal matrix is a **sparse structured** matrix

use Matlab’s **sparse matrices** tools

Matrices : structure

Esistono diversi tipi di matrici. Le matrici piene sono le classiche che conosciamo e hanno la maggior parte degli elementi diversi da 0. Le matrici sparse invece hanno la maggior parte degli elementi a 0, un tipico esempio è la matrice diagonale.

full matrices	sparse matrices
most entries are **non zero**	most entries are **zero**

sparsity indices

Possiamo definire due tipi di indici, di densità = al rapporto tra il numro di elementi non nulli e il numero degli elementi della matrice. L'indice di sparsità invece è il rapporto tra gli elementi nulli fratto gli elementi della matrice. Indice di psarsità = 1 vuol dire che tutti gli elementi sono 0.

density

$$d = \frac{nnz}{m \cdot n}$$

sparsity

$$s = 1 - d = \frac{m \cdot n - nnz}{m \cdot n}$$

mxn matrix

nnz = number of non zero elements

a seguire mostra un po di matrici sparse con una struttura.

structure

Si parla di struttura quando è possibile definire una regola per la quale sappiamo dove sono gli elementi nulli.

diagonal matrix

$$D = \begin{pmatrix} x & 0 & 0 & 0 \\ 0 & x & 0 & 0 \\ 0 & 0 & x & 0 \\ 0 & 0 & 0 & x \end{pmatrix}$$

Questa è una matrice diagonale e questa è la sua struttura.

$$a_{ij} = 0 \ \ , i \neq j$$

structure

$$T = \begin{pmatrix} x & x & 0 & 0 \\ x & x & x & 0 \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix}$$

tridiagonale matriz

Questa è tridiagonale.

$$a_{ij} = 0 \ , \left| i - j \right| > 1$$

structure

matrice a banda e quella è la regola.

$$
\begin{array}{cc}
 & \overbrace{\qquad\qquad\qquad}^{p} \\
B = & \left(\begin{array}{cccccc}
x & \cdots & x & 0 & \cdots & 0 \\
\vdots & \ddots & & \ddots & \ddots & \vdots \\
x & & \ddots & & \ddots & 0 \\
0 & \ddots & & \ddots & & x \\
\vdots & \ddots & \ddots & & \ddots & \vdots \\
0 & \cdots & 0 & x & \cdots & x
\end{array}\right)
\end{array}
\quad q \text{ (rows of the lower band)}
$$

band matrix

$$a_{ij} = 0 \quad , \; j - i > p \, , \; i - j > q$$

structure

Questa è una piena, però ha sulle diagonali gli stessi valori.

$$V = \begin{pmatrix} d & e & f & g \\ c & d & e & f \\ b & c & d & e \\ a & b & c & d \end{pmatrix}$$

Toeplitz matrix

equal entries along diagonals

structure

Questa è una matrice di toeplitz come vista vrima, ma è anche tridiagonale quindi tridiagonale di toeplitz.

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ 1 & 2 & -1 & 0 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

Toeplitz Tridiagonal matrix

quindi possiamo combinare più strutture.

combination of structures

structure

Qui abbiamo molti zero ma senza un regola in particolare quindi la chiamo sparsa non strutturata.

$$A = \begin{pmatrix} 6. & 0. & 0. & 3. & 0. \\ 4. & 1. & 0. & 0. & 2. \\ 7. & 0. & 6. & 0. & 8. \\ 0. & 0. & 9. & 5. & 0. \\ 0. & 0. & 0. & 0. & 3. \end{pmatrix}$$

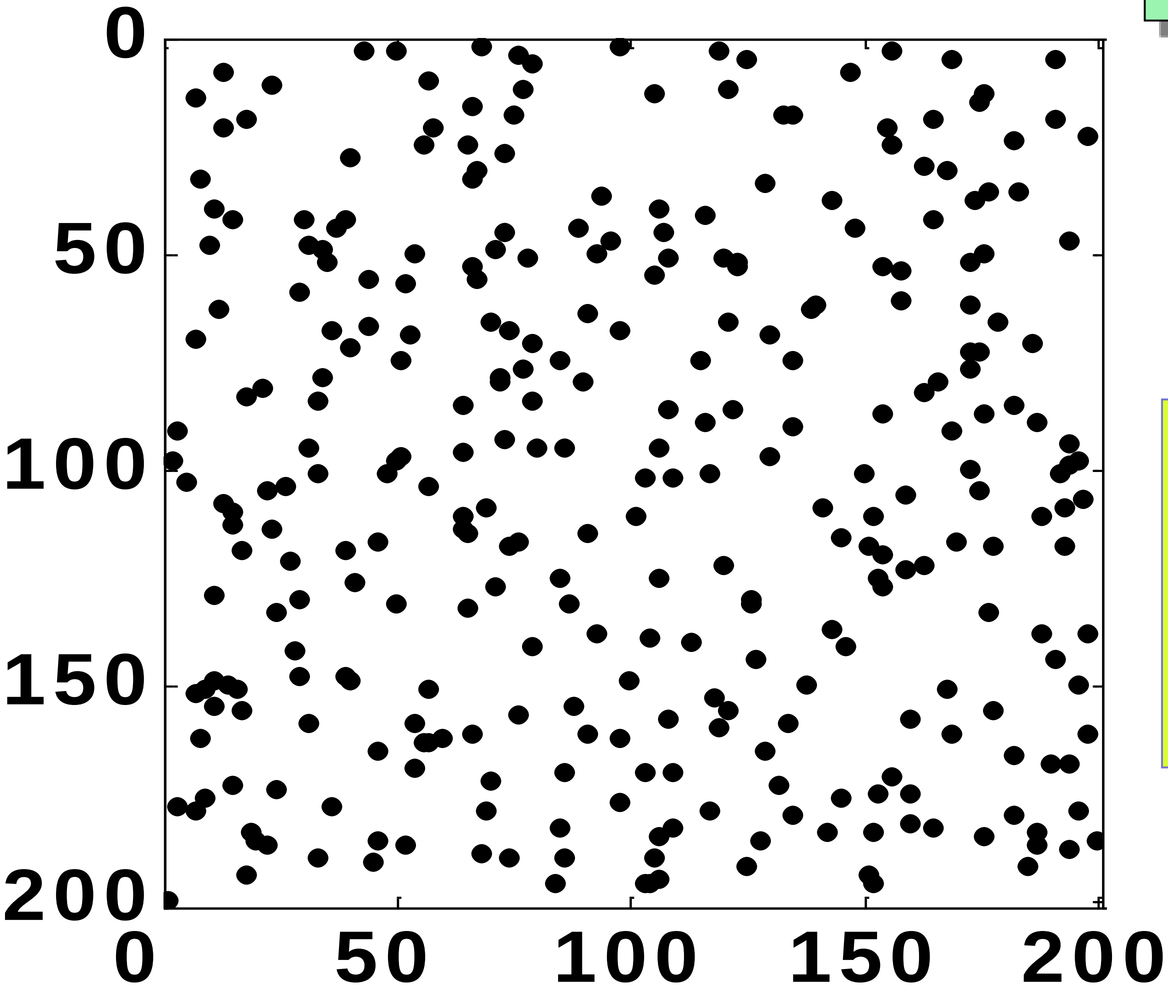
sparse un-structured matrix

Facciamo tutto questo perché il nostro obiettivo è memorizzare le matrici senza gli zeri, per risparmiare memoria e riscerivere gli algortimi in modo che ignorino le entries degli zeri e quindi più efficienti.

structure

Questo è un es. di matrice senza struttura.
Il bianco è 0 il nero è qualcosa di pieno.
Si nota come solo 1,6% è non nullo e l'indicie
di sparsità è s.

sparse un-structured matrix



s = 0.984

98.4% of entries are zero

1.6% non zeros

nnz = 40000*0.016

structure

Questa è una matrice diagonale a blocchi.
Quando vediamo una matrice sparsa dobbiamo trovare la sua struttura, che può essere anche più complessa delle strutture viste

$$\left(\begin{array}{ccc|cc|c} 2 & 4 & 9 & 0 & 0 & 0 \\ 3 & 0 & 6 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 9 & 7 & 0 \\ 0 & 0 & 0 & 7 & 5 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 6 \end{array}\right)$$

block-diagonal matrix

$$\begin{pmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \end{pmatrix}$$

Come memorizzaimo le matrici sparse strutturata o non strutturata?

data structure for storing matrices

full and **structured**, **sparse struttured** or **sparse un-structured**

per farlo possiamo usare una piena di toepliz che contiene l'array di tutti gli elementi che stanno sulle diagonali.

non standard storing (different from 2D arrays)

Exemple:

full Toeplitz → 1D array of size $m+n-1$ (number of diagonals)

Example:

Toeplitz (4x4)

$$T = \begin{pmatrix} 7 & -8 & 0 & 2 \\ 1 & 7 & -8 & 0 \\ 4 & 1 & 7 & -8 \\ 9 & 4 & 1 & 7 \end{pmatrix}$$

one 1D array of size **7**

```
T   [9,4,1,7,-8,0,2]
```

Example:

$$T = \begin{pmatrix} 7 & 5 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 0 & 2 & 4 & 8 \\ 0 & 0 & 6 & 1 \end{pmatrix}$$

tridiagonal

questi sono diversi modi di memorizzare.

three 1D arrays	2D array 3x4	2D array 4x3

```
c  [1,2,6]
d  [7,3,4,1]
f  [5,2,8]
```

$$\begin{bmatrix} 0 & 5 & 2 & 8 \\ 7 & 3 & 4 & 1 \\ 1 & 2 & 6 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 7 & 5 \\ 1 & 3 & 2 \\ 2 & 4 & 8 \\ 6 & 1 & 0 \end{bmatrix}$$

gli 0 per avere le stesse dimensioni

Example:

$$A = \begin{pmatrix} 2 & 0 & 3 & 0 & 0 \\ 1 & 5 & 0 & 4 & 0 \\ 0 & 2 & 6 & 0 & 5 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 3 & 4 \end{pmatrix}$$

5x5 band

storing **diagonal with offset**

questa è una matrice a banda

2D array 5x3

$$\begin{bmatrix} 0 & 2 & 3 \\ 1 & 5 & 4 \\ 2 & 6 & 5 \\ 1 & 2 & 0 \\ 3 & 4 & 0 \end{bmatrix}$$

OFFSET

-1	0	2

0 è la diagonale princ, sotto è -1 -2 etc.
Quindi possiamo creare un vettore degli offset, oppure direttamente una matrice Con le bande

sparse un-structured matrix

$$A = \begin{pmatrix} \boxed{6.} & 0. & 0. & 3. & 0. \\ 4. & 1. & 0. & 0. & 2. \\ 7. & 0. & 6. & 0. & 8. \\ 0. & 0. & 9. & 5. & 0. \\ 0. & 0. & 0. & 0. & 3. \end{pmatrix}$$

co-ordinate format

Questo è il modo con cui matlab lavora, cioè formato a coordinate. Considera il vettore formato da tutti gli elementi non nulli della matrice, si sceglono tipicamente muovendosi per riga, ma anche per colonna. Nelle colonne sotto al numero troviamo l'indice di riga e di colonna. Quindi prima riga tutti i valor inon nulli, 2 riga i rispettivi dinci di col e 3 riga risp indici di riga.

row-wise

```
A_nonzero [6. 3. 4. 1. 2. 7. 6. 8. 9. 5. 3.]
ind_col   [1  4  1  2  5  1  3  5  3  4  5 ]
ind_riga  [1  1  2  2  2  3  3  3  4  4  5 ]
```

$\longleftarrow$ **nnz** $\longrightarrow$

La lunghezza di questi vettori è nnz cioè il numero di elementi non nulli.

Lavorare con queste matrici ci permette di ridurre la complessità di spazio e ridure il tempo degli algoritmi che lavorano solo sugli elementi non nulli.

sparse un-structured matrix

store **only non-zero entries**

reduce space complexity

algorithms that act **only on non-zero entries**

reduce time complexity

Matlab ridefinisce poi gli operatori per lavorare su matrici sparse ed ha function per creare matrici sparse, visualizzare e estrarre informazioni.

sparse matrices in Matlab

co-ordinate format

elementary operators (`+`,`-`,`*`,`\`) e **`function`** di ALN acting on sparse matrices

`function` for building sparse matrices

`function` for extracting information and visualize sparse matrices

RIPRENDERE LA PROSSIMA LEZIONE DALLA REGISTRAZIONE SUCCESSIVA ALLA 01/10