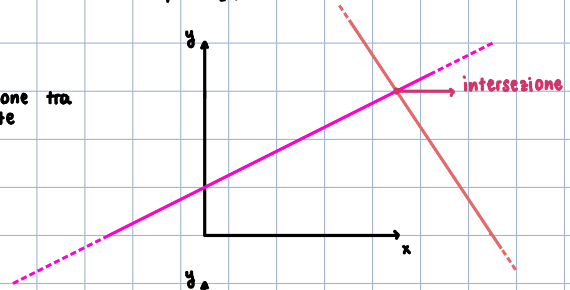
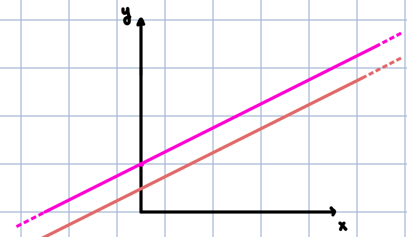
Partiamo da sistemi lineari molto semplici

La docente usa x_1 e x_2, io uso x, y

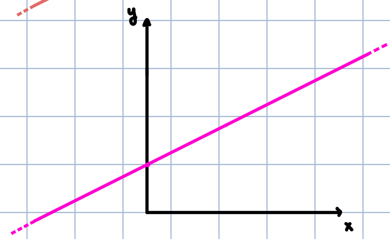
1) $\begin{cases} 3x + 2y = 18 \\ -x + 2y = 2 \end{cases}$ $\quad \begin{cases} x = 4 \\ y = 3 \end{cases}$ $\longrightarrow$ intersezione tra due rette



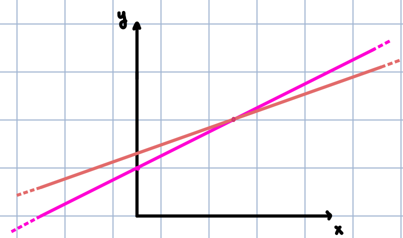
2) $\begin{cases} -\frac{1}{2}x + y = 1 \\ -\frac{1}{2}x + y = \frac{1}{2} \end{cases}$ non c'è soluzione: sono parallele



3) $\begin{cases} -\frac{1}{2}x + y = 1 \\ -x + 2y = 2 \end{cases}$ infinite soluzioni



4) $\begin{cases} -\frac{2.3}{5}x + y = 1.1 \\ -\frac{1}{2}x + y = 1 \end{cases}$ $\quad \begin{cases} x = 2.5 \\ y = 2.25 \end{cases}$



Considereremo sistemi quadrati di ordine n (= di n equazioni e n incognite)

Ogni singola equazione è rappresentata come una sommatoria $\downarrow x_1, \dots, x_n$

$$\boxed{\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i = 1, \dots, n}$$

$\det(a_{ij}) \neq 0$

$\downarrow$ così siamo sicuri che ha un'unica soluzione

$A\underline{x} = \underline{b}$ $\qquad A \in \mathbb{R}^{n \times n}$ $\qquad x \in \mathbb{R}^n$ $\qquad b \in \mathbb{R}^n$

$$\boxed{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}$$

$\det(A) \neq 0$

Calcoliamo i determinanti dei sistemi lineari:

1) $\det\begin{pmatrix} 3 & 2 \\ -1 & 2 \end{pmatrix} = 8$

2) $\det\begin{pmatrix} -\frac{1}{2} & 1 \\ -\frac{1}{2} & 1 \end{pmatrix} = 0$ matrice singolare

3) $\det\begin{pmatrix} -\frac{1}{2} & 1 \\ -1 & 2 \end{pmatrix} = 0$ matrice singolare

4) $\det\begin{pmatrix} -\frac{2\cdot 3}{5} & 1 \\ -\frac{1}{2} & 1 \end{pmatrix} = -0.04$

Esempio di sistema lineare, con dimensioni discrete, che può dare problemi:

H = hilb(15);

Voglio vedere come si comporta questa matrice lineare, faccio un test: assegno un termine noto per cui conosco la soluzione.

Voglio analizzare cosa succede a

$H\underline{x} = \underline{b}$

considero una soluzione che mi è nota: volessi $\underline{x} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$

Il termine noto B lo otterrei facendo il prodotto riga-colonna della matrice di Hilbert per il vettore colonna 1

b = H * ones (15, 1);

Con questo termine noto, al mia soluzione analiticamente è uguale a 1 (= un vettore di termini 1).
Per verificare se il mio algortimo reagisce bene, trovo le soluzioni della matrice:
x= H \ b

Per visualizzare meglio la matrice di Hilbert, il formato più adatto è rat.
hilb(4)

ans= $\begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{pmatrix}$

L'elemento nelle varie posizioni è 1/ (i+j -1), dove i riga, j colonna .
La matrice di Hilbert è un esempio di una mal condizionata matrice.
Verifichiamo:
H= hilb(4);
x_ex= ones(4, 1);
b= H* x_ex;
Torno in format long e
x= H\b // mi aspetto come soluzione un vettore da 4 righe, con tutti 1, ma non è ciò che ottengo

```
x =

   9.999999999999835e-01
   1.000000000000186e+00
   9.999999999995481e-01
   1.000000000000295e+00
```

perchè siamo in aritmetica di macchina, è andata abbastanza bene.

Facciamo anche vedere l'errore:
Err= norm(x- x_ex, inf) // prendo le componenti di x esatte, tolgo le componenti di x approssimate

e faccio la norma infinito (= ne seleziono il massimo)
Aumentiamo le dimensioni del sistema
H= hilb(15);
x_ex= ones(15,1);
b= H* x_ex;
Le cose vanno malino: il primo elemento non è male, poi precipita tutto.
Questo algoritmo non riesce a risolvere un sistema lineare con la matrice di Hilbert.
Matlab però avvisa che non riesce a risolvere a pieno questa matrice.
Il determinante di H non è 0, non è quindi una matrice singolare, ma è mooolto vicino allo 0.

Es: $A = \begin{pmatrix} \frac{1}{10} & 0 & \cdots & 0 \\ 0 & \frac{1}{10} & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \frac{1}{10} \end{pmatrix}$ $\qquad \det(A) = \frac{1}{10^n}$

⭑ Una matrice diagonale ha come determinante il prodotto degli elementi della diagonale

e se n è molto grande, il determinante sarà molto piccolo, tende a 0

Dopotutto però questa matrice non si comporta malissimo, ci sono due cose che ci preavvisano di quanto si comporterà male questa matrice:

- il determinante;
- Il COND nel warning di Matlab

Se io vado a risolvere un sistema lineare al calcolatore, a causa di arrotondamenti, di errori di calcolo, di errori dell'algoritmo, introduco degli errori che non mi fanno ottenere la soluzione esatta, ma mi fanno ottenere una sua approssimazione
Def: Vettore residuo

$$\underline{r} := \underline{b} - A\underline{\tilde{x}}$$

$\underline{\tilde{x}}$: non è la mia soluzione esatta, ma è la soluzione approssimata : $\underline{\tilde{x}} = \underline{x} + \delta\underline{x}$

Oss: $\overbrace{\frac{\|\delta \underline{x}\|}{\|x\|}}^{\text{Errore relativo}} \leq \mathrm{cond}(A) \cdot \frac{\|\underline{r}\|}{\|\underline{b}\|}$

Def: Condizionamento di una matrice

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

= numero di condizionamento

Se il numero di condizionamento è molto grande, questa relazione ci dice che l'errore relativo potrebbe essere molto grande [è una disuguaglianza, non è detto che si verifichi il caso peggiore]
Il numero di condizionamento è un proprietà intrinseca della matrice e non dipende dall'algoritmo.
In generale, il cond(A) è sempre un numero maggiore di 1

$$\mathrm{cond}(A) = \|A\| \, \|A^{-1}\| \geq \|A \cdot A^{-1}\| = \underbrace{\|Id\|}_{1}$$

Matlab da in output RCOND = è una stima del reciproco del condizionamento della matrice in norma 1.
Se la matrice è ben condizionata, rcond sarà vicino ad 1,. Se è mal condizionata sarà vicino ad eps.
PROP: se $\|\delta A\| < \frac{1}{\|A^{-1}\|}$ allora

potrebbe avere un err. rel molto grande

$$\boxed{\frac{\|\delta x\|}{\|\underline{x}\|}} \leq \frac{\mathrm{cond}(A)}{1 - \mathrm{cond}(A) \frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta A\|}{\|A\|} \cdot \frac{\|\delta \underline{b}\|}{\|\underline{b}\|} \right)$$

se io commetto un errore piccolo relativo nell'inserire gli elementi della matrice

⭑ potrebbe, è una disuguaglianza

un errore piccolo nell'inserire gli elementi del termine noto

se il condizionamento è molto elevato, potrebbe compensare il piccolo errore commesso nei dati

Es: Consideriamo questo sistema lineare

$A\underline{x} = \underline{b}$ con $A = \begin{pmatrix} \varepsilon & 0 \\ 0 & \frac{1}{3} \end{pmatrix}$ $0 < \varepsilon << 1$ $\det(A) = 1 \rightarrow$ non singolare

Calcoliamo il condizionamento di norma 1 di questa matrice

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}| = \max\left(\varepsilon, \frac{1}{\varepsilon}\right) = \frac{1}{\varepsilon}$$

Abbiamo bisogno della norma dell'inversa $A^{-1} = \begin{pmatrix} \frac{1}{\varepsilon} & 0 \\ 0 & \varepsilon \end{pmatrix}$

$$\|A^{-1}\|_1 = \frac{1}{\varepsilon}$$

$\rightarrow \text{cond}_1(A) = \frac{1}{\varepsilon^2}$, tanto più piccolo ε, maggiore sarà il condizionamento

$\downarrow$ mal condizionata

Si può migliorare il condizionamento di questo sistema moltiplicando per la matrice, entrambi i membri obv,

$C = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon^2 \end{pmatrix}$ PRECONDIZIONATORE

Risolviamo il sistema $\tilde{A}\underline{x} = \tilde{\underline{b}}$

$\tilde{A} = CA = \begin{pmatrix} \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix}$ $\tilde{\underline{b}} = C\underline{b}$ $A^{-1} = \begin{pmatrix} \frac{1}{\varepsilon} & 0 \\ 0 & \frac{1}{\varepsilon} \end{pmatrix}$

$$\text{cond}(\tilde{A}) = \|\tilde{A}\|_1 \cdot \|\tilde{A}^{-1}\|_1 = \varepsilon \cdot \frac{1}{\varepsilon} = 1$$

Oss: non è sempre possibile trovare un buon precondizionatore.
Il calcolo di un precondizionatore ha un costo computazionale.

Vediamo degli algoritmi poter risolvere sistemi lineari

$A\underline{x} = \underline{b}$ $\underline{x}, \underline{b} \in \mathbb{R}^n$ $A \in \mathbb{R}^{n \times n}$

$\det(A) \neq 0$

METODI DIRETTI: in aritmetica esatta, forniscono la soluzione esatta in un numero finito di passi;
METODI ITERATIVI: producono una successione di approssimanti della soluzione.

- A matrice diagonale (metodo diretto)

$A = \begin{pmatrix} a_{11} & & 0 \\ & a_{22} & \\ & & \ddots & \\ 0 & & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ $\underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$ $\det(A) \neq 0$

$\downarrow$ vale se sulla diagonale non c'è nessuno zero

$$A\underline{x} = \underline{b} \quad \begin{cases} a_{11}x_1 = b_1 \\ a_{22}x_2 = b_2 \\ \vdots \\ a_{nn}x_n = b_n \end{cases} \Rightarrow \begin{cases} x_1 = \frac{b_1}{a_{11}} \\ x_2 = \frac{b_2}{a_{22}} \\ \vdots \\ x_n = \frac{b_n}{a_{nn}} \end{cases}$$

Costo computazionale di questo algortimo: n divisioni

- A matrice triangolare inferiore

$$A = \begin{pmatrix} a_{11} & & & 0 \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & \dots & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$\det(A) \neq 0$
↓
vale se sulla diagonale non c'è nessuno zero

$$\begin{cases} a_{11}x_1 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + a_{n4}x_4 + \dots + a_{nn}x_n = b_n \end{cases} \Rightarrow \begin{cases} x_1 = \frac{b_1}{a_{11}} \\ x_2 = \frac{b_2 - a_{21}x_1}{a_{22}} \\ x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}} \\ x_4 = \frac{b_4 - a_{41} - a_{42}x_2 - a_{43}x_3}{a_{44}} \\ \vdots \\ x_n = \frac{b_n - a_{n1}x_1 - a_{n2}x_2 - \dots}{a_{nn}} \end{cases}$$

$$= \frac{b_n - \sum_{i=1}^{n-1} a_{ni}x_i}{a_{nn}}$$

Algoritmo: METODO di SOSTITUZIONE in avanti

$i = 1, \dots, n$

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}}$$

Costo computazionali di questo algoritmo :

1° RIGA → 1 divisione

2° RIGA → 1 divisione + 1 prodotto + 1 somma

3° RIGA → 1 divisione + 2 prodotto + 2 somma

⋮

N° RIGA → 1 divisione + n-1 prodotto + n-1 somma

TOTALE n divisione + $\frac{(n-1)n}{2}$ prodotto + $\frac{(n-1)n}{2}$ somma $\simeq O(n^2)$

Caso del tutto analogo:

- A matrice triangolare superiore

$$A = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{1n} \\ & a_{22} & & a_{2n} \\ & & \ddots & \vdots \\ 0 & & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$\det(A) \neq 0$
↓
vale se sulla diagonale non c'è nessuno zero

$$\begin{cases} \sum_{i=1}^{n} a_{1i}x_i = b_1 \\ \sum_{i=2}^{n} a_{2i}x_i = b_2 \\ \sum_{i=3}^{n} a_{3i}x_i = b_3 \\ \vdots \\ a_{n-1\,n-1}x_{n-1} + a_{n-1\,n}x_n = b_{n-1} \\ a_{nn}x_n = b_n \end{cases} \rightarrow \begin{cases} x_1 = \frac{b_1 - \sum_{i=2}^{n} a_{1i}x_i}{a_{11}} \\ \vdots \\ x_{n-1} = \frac{b_{n-1} - a_{n-1\,n}x_n}{a_{n-1\,n-1}} \\ x_n = \frac{b_n}{a_{nn}} \end{cases}$$

faccio anche $x_{n-2} = \dfrac{b_{n-2} - a_{n-2\,n-1}x_{n-1} - a_{n-2\,n}x_n}{a_{n-2\,n-2}}$

Algoritmo: METODO di SOSTITUZIONE all'indietro

$i = n, \cdots, 1 \mid n:-1:1$

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}$$

Costo computazionale: $O(n^2)$

A matrice qualsiasi densa o piena

$\underbrace{M \cdot A}_{\tilde{A} \text{ triangolare}} \underline{x} = \underline{b} \cdot M$ M non singolare

Lezione 8, min 50