Def: i NUMERI MACCHINA sono tutti i numeri rappresentabili esattamente sul calcolatore
Hanno sicuramente un numero finito di cifre: non sono rappresentabili i numeri irrazionali e quei numeri razionali che abbiano sviluppo decimale infinito (es. i numeri periodici)
Def: l'insieme dei numeri macchina è chiamato SISTEMA FLOATING POINT
Es: 31415 -0.00000123

$= 3.1415 \cdot 10^4$ $= -1.23 \cdot 10^{-6}$

$= 0.031415 \cdot 10^6$ $= -0.123 \cdot 10^{-5}$ * notazione scientifica

* $= 0.31415 \cdot 10^5$ $= 123 \cdot 10^{-8}$

Floating point perchè il "punto", la virgola si sposta.

$p \rightarrow$ 11 bit, esponente $\in [-1022, 1023]$

$$x = \underset{\text{segno}}{s} \cdot \underbrace{\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \dots + \frac{d_t}{\beta^t}\right)}_{\text{mantissa, 52 bit, } 0 \le d_i \le \beta - 1 \quad i = 1, \dots, t} \underset{\text{base}}{\beta}^p$$

$x = \pm 0 . d_1 d_2 \dots d_t \cdot \beta^p$

PICCOLA GUIDA MATHLAB:

- comando *help* richiede il significato di alcune funzioni o di alcune costanti già predefinite

\>> *help realmax*

// *spiegazione*

\>> *realmax*

ans= 1.797693134862316e+308

\>> help *realmin*

// *spiegazione*

\>> *realmin*

ans= 2.225073858507201e-308

//proviamo ora ad inserire un numero più grande del max

\>> 1.9 * 10^350

ans= Inf

\>> help *Inf*

// spiegazione

//proviamo ora ad inserire un numero più piccolo del min

\>> 1.2* 10^-8

ans= 1.200000000000000e-308

//ma quindi posso rappresentare numeri più piccoli? Rivediamo l'aiuto e aggiungiamo una specifica

\>> help *realmin*

//spiegazione

Se si rinuncia a qualcosa (tipo il bit del segno, qualche dettaglio della rappresentazione normalizzata floating point standard) allora si può tenere in memoria qualche numero più piccolo del realmin positivo.
Quindi il numero che Matlab ha tenuto in memoria, che abbiamo inserito noi, ha rinunciato alla

rappresentazione standard floating point.

>> 1/0
ans= Inf
//lo vede come limite
>>0/0
ans= NaN
//not a number
>> help Nan
//spiegazione

>> help *eps*
…
eps, with no arguments, is the distance from 1.0 to the next larger doubler precision number
that is eps with no arguments returns 2^(-52)
…
>> eps
ans= 2.220446049250313e-16

>> 2^(-52)
ans= 2.220446049250313e-16
// è il più piccolo numero che possiamo sommare a uno e che MathLab riesce a memorizzare
>> 1+eps >1
ans= logical
1
// vero
>> 1+1.11* 10^ (-16) > 1
ans= logical
0
// falso
// se aggiungo ad uno qualcosa di più piccolo di eps, per Mathlab quel numero non esiste!

>> 1+ realmin
ans= 1
//realmin non esiste se aggiunto ad 1, tra 1 e 1+esp il calcolatore non riesce a mettere nessun numero.

Infatti i calcoli che faremo, noi commetteremo un errore che sarà sempre maggiore o uguale a eps
Eps viene chiamata PRECISIONE DI MACCHINA.

OSS: il più grande numero positivo memorizzabile è REALMAX, al di sopra del quale si ha overflow.
Il minimo numero maggiore di 0 rappresentabile è REALMIN.
Altri valori più piccoli compresi tra 0 e REALMIN possono essere rappresentati rinunciando alla rappresentazione standard IEEE dei numeri floating point.
OSS: alcuni valori speciali richiedono una codifica speciale: NaN e Inf.

Def: l'EPSILON MACCHINA è il più piccolo numero macchina positivo x tale che

$(1+x) > 1$

E indica la sensibilità del sistema floating point adottato, indica di quanto possa variare al più l'errore relativo

Vediamo alcuni errori di rappresentazione di questi numeri:

```
//creiamo una successione
>> S(1)=0;
//per i che va da 1 a 10000
>> for i=1:1000
// vado a memorizzare in S(i+1) il valore di S(i) ci aggiungo 10^(-4)
>> S(i+1)=S(i)+0.0001;
>> end
//vado a vedere l'ultima cella della mia successione
>> S (end)
      ans= 9.999999999999062e-01
// è 1? No, c'è un errore, possiamo anche vedere quale errore commettiamo:
>> abs (1-S (end))
      ans= 9.381384558082573e-14
```

//0.0001 sommato tutte le volte, in realtà sono numeri che non sono rappresentati correttamente nella rappresentazione floating point, quindi si iniziano a introdurre degli errori, quando poi faccio delle somme via a via aumentano.

```
//se avessi introdotto la successione con una delle possibili sintassi di matlab
>> S= [0:0.0001:1];
//traduzione: da 0 a 1, con passo 0.0001
>>S (end)
      ans= 1
```

//a seconda dell' algoritmo che si usa per assegnare/calcolare dei valori si commettono più o meno errori

Es: ERRORI DI RAPPRESENTAZIONE

```
* S(1) = 0
for i = 1: 10000
      S(i+1) = S(i) + 0.0001
      end
S (end)= 9.999999999999062e-01
```

```
* S= [0: 0.0001:1]
S(end) = 1
```

Es: ERRORI DI CANCELLAZIONE NUMERICA

x= 77777777

dovrebbe esserlo

$y=\sqrt{x^2+1}-x \overset{?}{=} z=\dfrac{1}{\sqrt{x^2+1}+x} = 6.428571493 \cdot 10^{-9}$ → sia Matlab che calcolatrice

$= 0$

//verifico i singoli passaggi:
>> x= 77777777
>> x^2
    ans= 6.049382595061729e+15
>> x^2 +1
    ans= 6.049382595061730e+15
>> sqrt (x^2 +1)
    ans= 77777777
//non dovrebbe essere esattamente x, matlab non riesce a memorizzare il fatto che io abbia aggiunto 1
//ora verifico z
>> sqrt (x^2 +1) +x
    ans=15555554
>> 1/ (sqrt (x^2 +1) +x)
    ans= 6.428571492857143e-09

Esercizio: Verifico su Matlab queste espressioni analiticamente equivalenti:

$y_1 = (1 - x^6)$ $\qquad$ $y_2 = x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1$

In 100 punti equidistanti nell'intervallo $[1-\delta, 1+\delta]$ $\qquad$ $\delta = 0.1, 0.01, 0.005, 0.0025$

OSS: alcune proprietà che valgono in aritmetica esatta non valgono in aritmetica floating point.
Per le operazioni macchina

$\odot : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ $\qquad$ $\boxdot : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

data una qualsiasi operazione in aritmetica esatta $\qquad$ operazione in aritmetica floating point

Vale in generale la proprietà commutativa ma non le altre proprietà (es. associativa)

$(a \boxplus b) \boxplus c \neq a \boxplus (b \boxplus c)$

>> a = 0.1234567;
>> b = 6666.325;
>> c = -6666.325;
>> (a + b) + c
    ans= 1.234567000001334e-01
>> a + (b + c)
    ans= 1.234567000000000e-01
// vediamo perchè il primo è sbagliato
>> a + b
    ans= 6.666448456700000e+03
// se sommo c, in realtà dopo gli 0 che vediamo stampati, non è riuscito a tenere in memoria altri 0, non sa che ce ne sono altri; quando va a sottrarre c va a mettere altro e sporca il risultato.

Def: ERRORE ASSOLUTO=

$$E_a := \|\text{valore esatto} - \text{valore approssimato}\|$$

ERRORE RELATIVO=

$$E_r := \frac{E_a}{\|\text{valore esatto}\|}$$

Def: un modello f(x) si dice BEN CONDIZIONATO se vale una relazione del tipo

norma * VARIAZIONE di dato

$$\frac{\|f(x+\delta x) - f(x)\|}{\|f(x)\|} \le k \frac{\|\delta x\|}{\|x\|} \qquad f(x) \neq 0 \quad x \neq 0$$

* norma perché potrebbero essere anche dei vettori, ma pensiamo pure al modulo

Con k "piccolo".
k è definito NUMERO DI CONDIZIONAMENTO.

Condizionamento di un problema: ad una variazione dei dati corrisponde una variazione dei risultati di che entità? È dello stesso ordine di grandezza? Se commetto un errore dell'ordine di 10^(-2), il mio output ha un errore a sua volta dell'ordine di 10^(-2)?
In generale, se si verifica questa situazione, il problema è BEN condizionato (ordine di grandezza degli errori sui dati = ordine di grandezza degli errori sui risultati); se invece queste due grandezze sono molto diverse, il problema si dice MAL condizionato.

Studiamo il condizionamento della somma:

$x, y \in \mathbb{R}$

$$\bar{x} = fl(x) = x(1+\varepsilon_1) = x + x\varepsilon_1$$

errore relativo ad x

$$\bar{y} = fl(y) = y(1+\varepsilon_2) = y + y\varepsilon_2$$

disuguaglianza triangolare: il modulo della somma è minore o uguale della somma dei moduli

$$\frac{|\bar{x}+\bar{y}-(x+y)|}{|x+y|} = \frac{|\cancel{x}+x\varepsilon_1+\cancel{y}+y\varepsilon_2-\cancel{x}-\cancel{y}|}{|x+y|} = \frac{|x\varepsilon_1 + y\varepsilon_2|}{|x+y|} \le \frac{|x||\varepsilon_1|}{|x+y|} + \frac{|y||\varepsilon_2|}{|x+y|} = k_1|\varepsilon_1| + k_2|\varepsilon_2|$$

Quando k può essere molto grande?

Se x e y sono simili ma con segno opposto

Quindi, quando la somma è MAL condizionata?

Se $x \to -y$ allora $k_1, k_2 \to +\infty$ Cattivo condizionamento (CANCELLAZIONE NUMERICA)

Condizionamento del prodotto:

scomodo: $ab \le \frac{1}{2}(a^2+b^2)$

$$\frac{|\bar{x}\bar{y} - xy|}{|xy|} = \frac{|x(1+\varepsilon_1)y(1+\varepsilon_2) - xy|}{|xy|} = |\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2| \le |\varepsilon_1|+|\varepsilon_2|+|\varepsilon_1\varepsilon_2| \le |\varepsilon_1|+|\varepsilon_2|+\frac{1}{2}|\varepsilon_1^2+\varepsilon_2^2| \le |\varepsilon_1|+|\varepsilon_2|+\frac{1}{2}|\varepsilon_1|+\frac{1}{2}|\varepsilon_2|$$

Siccome questi ε dovrebbero essere molto piccoli, faccio quest'ipotesi

ossia: $|\varepsilon|^2 < |\varepsilon|$ se $|\varepsilon| \le 1$

$$= \underbrace{\frac{3}{2}}_{k_1}|\varepsilon_1| + \underbrace{\frac{3}{2}}_{k_2}|\varepsilon_2|$$

OSS: k= 3/2 quindi il prodotto è BEN condizionato.
Non dipende da x e y, il nostro k è sempre 3/2, l'errore che commetto sui risultati è dello stesso

ordine di grandezza dell'errore che commetto sui dati.
Da qui ne concludo che la moltiplicazione al calcolatore è BEN condizionata rispetto alla somma (= è meglio che faccia fare moltiplicazioni al calcolatore piuttosto che delle somme, dal punto di vista del condizionamento)

Eulero

Matlab: è case sensitive
Per il numero di Nepero: exp(1)
i= di default è la parte immaginaria dei complessi
sqrt()= radice quadrata
sin ()
cos ()
tan () $\longrightarrow$ argomento in radianti
asin ()
acos ()
atan ()
log() = [ha come base il numero di Nepero]
log2()= ha base 2
log10()= ha base 10
abs () =valore assoluto
Di default il risultato (ans) lo si vede con 15 cifre decimali e in notazione scientifica, altrimenti si può cambiare con:

- **format short**: il risultato avrà 4 cifre decimali
- **format short e**: 4 cifre decimali + notazione scientifica
- **format long**: per ripristinare le 15 cifre, non min formato scientifico
- **format long e**: 15 cifre + notazione scientifica

Se voglio cancellare una sola variabile: clear nomevariabile
clc = pulisco tutto il foglio

RICERCA DI RADICI DI EQUAZIONI NON LINEARI
Riconsideriamo il problema:

data $f:(a,b) \subseteq \mathbb{R} \to \mathbb{R}$, si cerca $\alpha \in (a,b)$ t.c. $f(\alpha)=0$

Sia $f \in C^m(a,b)$ $m \in \mathbb{N}^+$

Def: α si dice RADICE SEMPLICE

$f(\alpha) = 0$ e $f'(\alpha) = 0$

Def: α si dice RADICE DI ORDINE m se

$f^{(m-1)}(\alpha) = f^{(m-2)}(\alpha) = \cdots = f'(\alpha) = f(\alpha) = 0$ e $f^{(m)}(\alpha) \neq 0$. In tal caso $f(x) = (x-\alpha)^m h(x)$ con $h(\alpha) \neq 0$.

Es: $\sqrt{2}$ è radice della f.ne $f(x) = x^2 - 2$

$f'(x) = 2x$ $f'(\sqrt{2}) \neq 0$ $\to$ $\sqrt{2}$ è RADICE SEMPLICE di $f(x) = x^2 - 2$

Es: $f(x) = x^2 - 3x + 2 = (x-1)(x-2)$

ha due radici $x=1$ e $x=2$ semplici

Es: $x^2 - 2x + 1 = (x+1)^2$

ha una sola radice di ORDINE 2 $\quad x = 1 \quad f(1) = 0$

$f'(x) = 2(x-1) \quad f'(1) = 0$

$f''(x) = 2 \quad f'(1) = 2 \neq 0$

METODO DI BISEZIONE si basa sul teorema di esistenza degli zeri per funzioni continue.

Data una funzione: $f:[a,b] \to \mathbb{R}$, continua in $[a,b]$ e t.c. $f(a)f(b) < 0$, allora $\exists\, \alpha \in (a,b)$ t.c. $f(\alpha) = 0$.

Algoritmo:

$I_0 = [a,b] \quad a^0 = a \quad b^0 = b \quad x^0 = \frac{1}{2}(a+b)$

$f(x_0) = 0$ STOP x_0 è radice

$f(x^0)f(a^0) < 0 \quad a^1 = a^0 \quad b^1 = x^0$

$f(x^0)f(b^0) < 0 \quad a^1 = x^0 \quad b^1 = b^0$

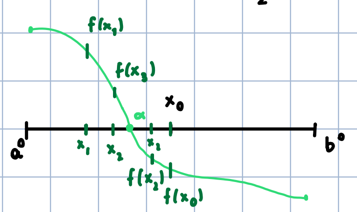
$x_1 = \frac{a^1 + b^1}{2}$

$f(x_1)f(a_1) < 0 \quad a_2 = a_1 \quad b^2 = x_1$

$f(x_1)f(b_1) < 0 \quad a_2 = x_1 \quad b_2 = b_1$

$x^2 = \frac{a_2 + b_2}{2}$

$I_k = [a_k, b_k] \qquad x_k = \frac{a_k + b_k}{2}$



Supponiamo di arrestare l'algoritmo al passo m > 0 $\quad m \in \mathbb{N}$

$$E_{a(m)} = |x_m - \alpha| \leq |b_m - a_m| = \frac{|b_0 - a_0|}{2^m} \xrightarrow[m \to +\infty]{} 0$$

PROPOSIZIONE: il metodo di bisezione è GLOBALMENTE CONVERGENTE

Oss: se fisso una tolleranza ε, posso conoscere a priori il numero di passi necessari per ottenere un'approssimazione della radice con la tolleranza richiesta.

$e_m \leq \varepsilon$

$|x_m - \alpha| \leq \varepsilon$

$e_m = |x_m - \alpha| < \frac{|b-a|}{2^m} \leq \varepsilon \qquad \frac{|b-a|}{\varepsilon} \leq 2^m \qquad \log_2\left[\frac{b-a}{\varepsilon}\right] \leq m \quad \leftarrow$ arrotondamento per eccesso all'intero sucessivo

Oss: il metodo di bisezione non è a convergenza monotona e converge "lentamente" rispetto ad altri metodi.

$\downarrow$

può esistere k t.c. $e_{k+1} > e_k$

Es: Calcolare con il metodo di bisezione

$$P_5(x) = \frac{x}{8}(63x^4 - 70x^2 + 15) \qquad a = 0.6 \qquad b = 1 \qquad \varepsilon = 10^{-10} \qquad \alpha \cong 0.9062$$

Def: si dice che la successione $\{x_k\}_{k=1,\dots}$ generata da un metodo numerico, converge ad α con ordine p se

$$\exists c > 0 : |x_{k+1} - \alpha| \leq c\,|x_k - \alpha|^p \qquad \forall k > k_0 \qquad k_0 \in \mathbb{N}$$

Oss: Il metodo di bisezione è globalmente convergente, ma non è neanche di ordine 1.
Oss: Nel caso p=1 per avere la convergenza ad α, necessariamente c < 1.

ALGORITMI CON ORDINE DI CONVERGENZA SUPERIORE MA SOLO LOCALMENTE CONVERGENTI
Supponiamo che α sia la radice della funzione non lineare f.
Supponiamo che f sia derivabile

$$0 = f(\alpha) = f(x_0) + f'(x_0)\cdot(\alpha - x_0) + f''(x_0)\cdot\left(\frac{\alpha - x_0}{2!}\right)^2 + f'''(x_0)\left(\frac{\alpha - x_0}{3!}\right)^3 + f^{IV}(x_0)\left(\frac{\alpha - x_0}{4!}\right)^4 + \dots + \theta(\alpha - x_0)^k$$

$\exists c \in [x_0, \alpha]$ $\qquad f(\alpha) = f(x_0) + \underbrace{f'(c)}_{\approx q}(\alpha - x_0)$

$0 = f(\alpha) \neq f(x_0) + q(\alpha - x_0)$ → non è più un uguaglianza

$$f(x_0) + q(x_1 - x_0) = 0 \qquad x_1 = \frac{-f(x_k)}{q} + x_0$$

Reitero il procedimento

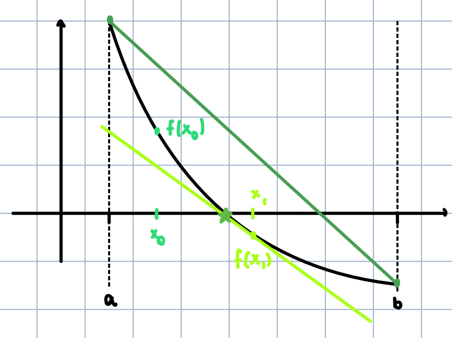
$$x_{k+1} = x_k - \frac{f(x_k)}{q_k}$$

METODO DELLE CORDE

uguale ad ogni passo

$$q_k = \text{costante} = \frac{f(b) - f(a)}{b - a}$$

$$\begin{cases} x_0 \text{ dato} \\ x_{k+1} = x_k - \dfrac{b-a}{f(b) - f(a)}\cdot f(x_k) \end{cases}$$



Considero la retta passante per $(a, f(a))$ e $(b, f(b))$

$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a) \qquad \text{CORDA}$$

Considero la retta parallela alla corda passante per $(x_k, f(x_k))$

$$y = \frac{f(b) - f(a)}{b - a}(x - x_k) + f(x_k)$$

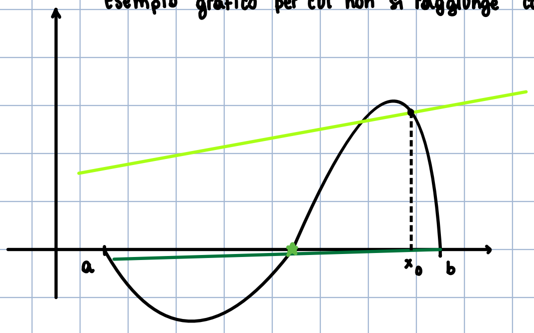
Trovo l'Intersezione tra la nuova retta e l'asse delle ascisse

$$\frac{f(b) - f(a)}{b - a}(x - x_k) + f(x_k) = 0 \;\rightarrow\; x = x_{k+1} = x_k - \underbrace{\boxed{\frac{b-a}{f(b) - f(a)}}}_{q_k} f(x_k) \qquad k = 0, 1, \dots$$

x_0 assegnato

Oss: Il metodo delle corde ha ordine di congruenza pari a 1.
Oss: il metodo delle corde è solo localmente convergente.

Esempio grafico per cui non si raggiunge convergenza:



TEOREMA: il metodo delle corde converge se

1. $\text{segno}\left[f'(\alpha)\right] = \text{segno}\left[q\right] = \text{segno}\left[\frac{f(b)-f(a)}{b-a}\right]$

2. $(b-a) < 2\,\frac{f(b)-f(a)}{f'(\alpha)}$

Comando fzero(FUN, X0): it tries to find a zero of the function FUN near X0, if X0 is scalar. It first finds an interval containing X0 where the function values of the interval endpoints differ in signs, then searches that interval for a zero. FUN is a function handle. FUN accepts real scalar input X and returns a real scalar function value F, evaluated at X. The value X returned by fzero is near a point where FUN changes sign (is FUN is continuous), or NaN if the search fails