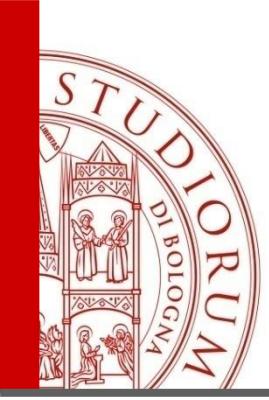


Condizionamento di un problema matematico

Stabilita' dell'algoritmo risolutivo

Problemi ed algoritmi

Problema numerico:

descrizione chiara e non ambigua di una connessione funzionale tra i dati x (input) del problema e i risultati desiderati (output) y.

Algoritmo:

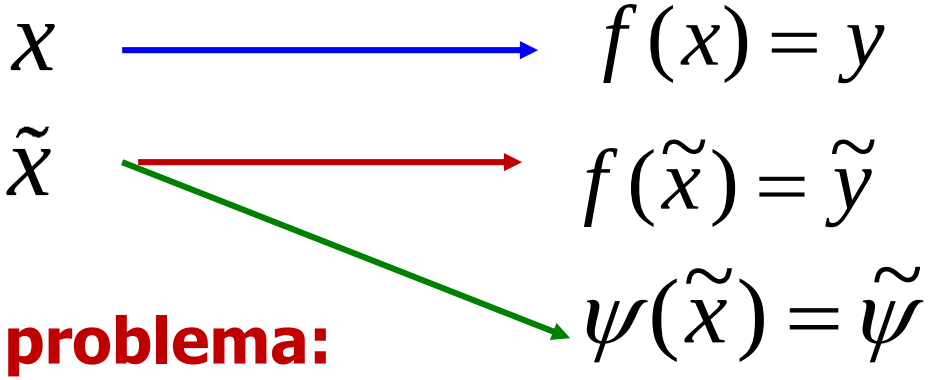
sequenza di di operazioni che devono essere eseguite al fine di ottenere, in un numero finito di passi, da un vettore dati x il corrispondente output $\tilde{\psi}$ non necessariamente uguale ad y.

AD OGNI PROBLEMA NUMERICO POSSIAMO ASSOCIARE PIU' ALGORITMI

Condizionamento di un problema

Perturbazioni nei dati: $\tilde{x} = x + \delta x$

Come vengono propagati dal problema (f) gli errori presenti nei dati in assenza di errori di calcolo e indipendentemente dal procedimento di calcolo?



Condizionamento di un problema:
Il confronto tra la risposta analitica f(x) e la risposta $f(\tilde{x}) = \tilde{y}$ ottenuta a partire da dati perturbati.

Problema ben/mal-condizionato

Quando a **piccole perturbazioni** (relative) nei **dati x** corrispondono perturbazioni (relative) sul risultato **f(x)** dello stesso ordine di grandezza, il **problema y=f(x)** è definito **ben condizionato**, altrimenti è detto **mal condizionato.**

Uno stesso problema può essere mal condizionato per certi dati ma non per altri.

Indice di condizionamento

$$\frac{\|f(x)-f(\tilde{x})\|}{\|f(x)\|} \leq K \frac{\|x-\tilde{x}\|}{\|x\|}$$

K è detto INDICE DI CONDIZIONAMENTO

Il condizionamento è legato al problema numerico e non ha alcun legame con gli errori di arrotondamento delle operazioni di macchina nè con il particolare algoritmo utilizzato.

Esempio 1

- Problema matematico:

Valutazione di una funzione f in un punto x (f differenziabile)

$$\tilde{x} = x + \delta x$$

x esatto, perturbazione:

$$f(x+\delta x) - f(x) \approx f'(x)\delta x$$

$$\frac{f(x+\delta x) - f(x)}{f(x)} \approx \frac{f'(x)\delta x}{f(x)}$$

$$\frac{f(x+\delta x) - f(x)}{f(x)} \approx \left(\frac{f'(x)x}{f(x)}\right)\frac{\delta x}{x}$$

K

Esempio 1: f(x)=tan(x)

$$x = 1.57079$$

$$tan(1.57079) = 1.58058x10^5$$

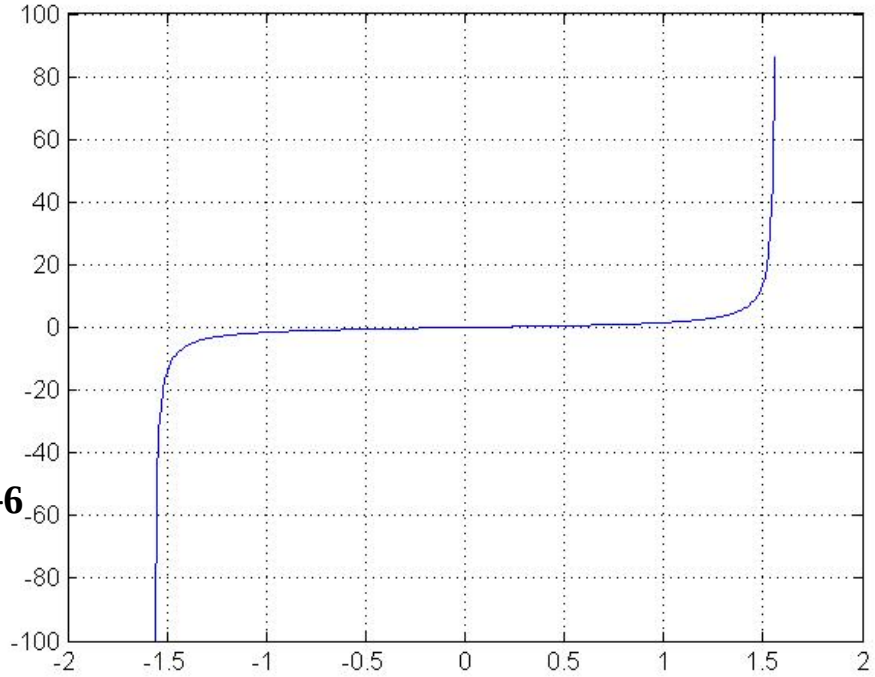
$$f'(x) = 1 + tan^2(x)$$

$$K = |\, x(1 + tan^2(x)) / tan(x) \,|$$

$$K = 2.48275x10^5$$

$$\delta x = |\, 1.57079 - 1.57078 \,| \approx 6.37x10^{-6}$$

$$\frac{f(x+\delta x) - f(x)}{f(x)} \approx 1.58$$



Per valori di x vicini a multipli di pi/2 (1.5707**9**6326794897) la funzione f(x) amplifica gli errori sul dato x, il problema è mal-condizionato

Esempio 2: problema mal condizionato **(Wilkinson 1963)**

Calcolare le radici del seguente polinomio:

$p(x) = (x-1)(x-2)\ldots(x-19)(x-20)$

$= x^{20}-210\,x^{19}+\ldots$

Calcolatore **B=2 t=30**

per il calcolo degli zeri **1, 2, 3, …,19, 20**

Per memorizzare i coefficienti del polinomio è necessario effettuare un arrotondamento alla 30 cifra significativa binaria

Perturbiamo ora il coefficiente di x^{19}

$-210 \longrightarrow -210 + 2^{-23}$

Esempio 2: problema mal condizionato **(Wilkinson 1963)**

Il polinomio diventa: $p(x)+2^{-23}\,x^{19}=0$

come sono variate di conseguenza le radici?

1.00000 0000	10.09526 6145 ± 0.64350 0904i
2.00000 0000	11.79363 3881 ± 1.65232 9728i
3.00000 0000	13.99235 8137 ± 2.51883 0070i
4.00000 0000	16.73073 7466 ± 2.81262 4894i
4.99999 9928	19.50243 9400 ± 1.94033 0347i
6.00000 6944	
6.99969 7234	
8.00726 7603	
8.91725 0249	
20.84690 8101	

Esempio 3:
problema mal condizionato

Risolvere il sistema lineare:

$$\begin{cases} x + y = 2 \\ 1001x + 1000y = 2001 \end{cases}$$

$$x = 1, \quad y = 1$$

Perturbiamo il coefficiente della x dell'1%:

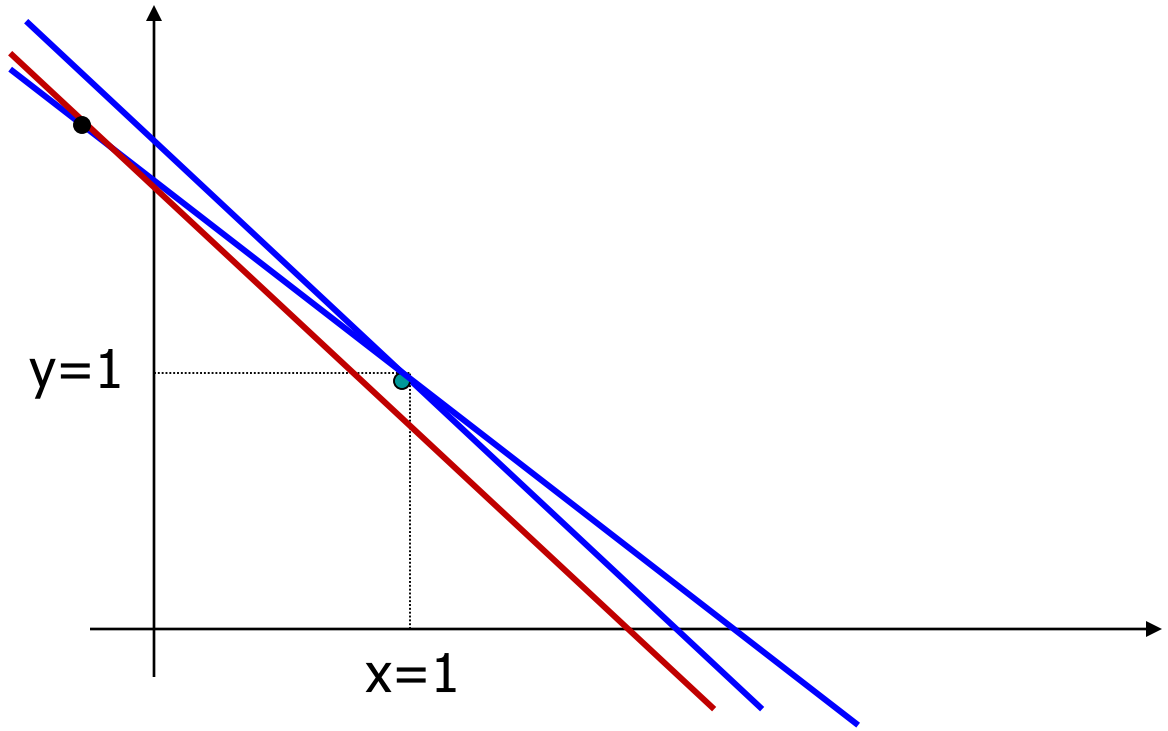
$$\begin{cases} (1+\frac{1}{100})x + y = 2 \\ 1001x + 1000y = 2001 \end{cases}$$

$$x = -1/9 = -0.1111, \quad y = 1901/900 = 2.1122$$

Errore del 110%

Esempio 3:
problema mal condizionato

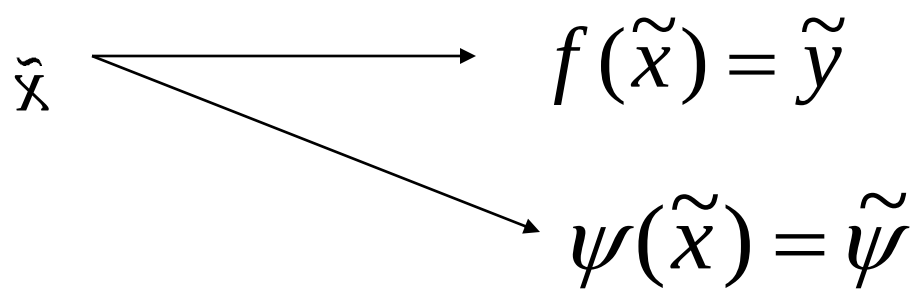
- Interpretazione geometrica



Algoritmo e stabilità

Stabilità di un algoritmo :

Confronto tra la risposta fornita dall'algoritmo $\psi(\tilde{x})$ con $f(\tilde{x}) = y$



E' generato dal calcolo della funzione $\tilde{\psi}$ come composizione di un numero finito di operazioni di macchina

La stabilità di un algoritmo valuta quindi la reazione fornita dall'algoritmo all'introduzione di perturbazioni nei dati iniziali.

Stabilità algoritmica

- Comportamento dell'algoritmo rispetto alla propagazione degli errori
- Se un algoritmo è **stabile** la successione con cui esegue le operazioni non amplifica più dell'inevitabile gli errori iniziali.
- Se un algoritmo è **instabile**, il risultato che produce non è accettabile in quanto la successione delle operazioni che esegue in aritmetica finita amplifica in modo esagerato gli inevitabili errori dovuti alla rappresentazione finita dei numeri.

Errore inerente ed algoritmico

Nel calcolo di una f, il valore effettivamente calcolato in corrispondenza dei dati x può essere affetto da errori di due tipi:

- **ERRORE INERENTE**:
  generato dalla rappresentazione dei dati come numeri finiti

$$E_{IN} = \frac{f(\tilde{x}) - f(x)}{f(x)}$$

- **ERRORE ALGORITMICO**:
  generato dal calcolo della funzione $\psi(\tilde{x})$ dovuto alle operazioni in aritmetica finita

$$E_{ALG} = \frac{\psi(\tilde{x}) - f(\tilde{x})}{f(\tilde{x})}$$

ERRORE TOTALE

L'accuratezza della soluzione (di quanto si discosta la soluzione calcolata da quella esatta) dipende sia dal condizionamento del problema che dalla stabilità algoritmica.

$$E_{TOT} = \frac{\psi(\tilde{x}) - f(x)}{f(x)}$$

$$E_{TOT} = E_{ALG}(1 + E_{IN}) + E_{IN} \approx E_{ALG} + E_{IN}$$

Infatti:

$$E_{TOT} = \frac{\psi(\bar{x}) - f(x)}{f(x)} = \frac{\psi(\bar{x})}{f(x)} - 1 = \frac{\psi(\bar{x})}{f(\bar{x})}\frac{f(\bar{x})}{f(x)} - 1 = (1 + E_{ALG})(1 + E_{IN}) - 1$$

Condizionamento di un problema, algoritmo e stabilità

La bassa accuratezza dei risultati prodotti da un processo numerico può essere imputabile a

alto mal condizionamento intrinseco del problema

oppure

all' instabilità dell'algoritmo utilizzato per produrlo.

Condizionamento di un problema, algoritmo e stabilità

La stabilità dell'algoritmo non garantisce che il risultato calcolato sia accurato.

Per un problema mal condizionato la distinzione tra algoritmo stabile e instabile non è molto significativa in quanto l'errore totale risulta dominato dall'errore inerente.

Quindi per un problema mal condizionato è opportuna, in generale, una sua riformulazione.

Algoritmo Stabile

Si parla di Stabilità o di Instabilità numerica intendendo che gli errori sui dati non sono (o sono) amplificati durante lo sviluppo dell'algoritmo

$$\left| E_{ALG} \right| \approx g(n) \cdot eps$$

n= numero di operazioni effettuate

g(n)=c*n ,c>0, crescita dell'errore lineare

$g(n)=c^n$,c>1 crescita dell'errore esponenziale

Un **algoritmo** numerico è considerato **stabile** se g(n) è lineare, cioè l'errore algoritmico è dell'ordine di grandezza della precisione di macchina, **instabile** altrimenti.

Esempio 1: algoritmo instabile

Valutare la funzione

$$y = \frac{(1+x)-1}{x}$$

Mediante l'algoritmo:

```
x=1e-15;
y=((1+x)-1)/x;
```

Valutiamo l'espressione con MATLAB otteniamo:

y=1.11022302462516 invece di y=1

Il problema è ben condizionato

$$K = \left| \frac{f'(x)x}{f(x)} \right| = 0 \quad poichè \quad f'(x) = 0 \; \forall x \neq 0$$

L'algoritmo è stabile?

$$fl(1+x) = (1+ fl(x))(1+\varepsilon)$$

$$fl((1+x)-1) = (fl(1+x)-1)(1+\varepsilon) \approx x(1+2\varepsilon)+\varepsilon$$

$$\frac{\frac{fl((1+x)-1)}{fl(x)}(1+\varepsilon)-1}{1} = \frac{x(1+2\varepsilon)+\varepsilon}{x(1+\varepsilon)}(1+\varepsilon)-1 \approx 2\varepsilon+\frac{\varepsilon}{x}$$

Valore f(x) esatto

Se x è piccolo, l'errore su y potrebbe essere grande.

Esempio 2: algoritmo instabile

La successione

$$1, \frac{1}{3}, \frac{1}{9}, \ldots, \frac{1}{3^n}, \ldots$$

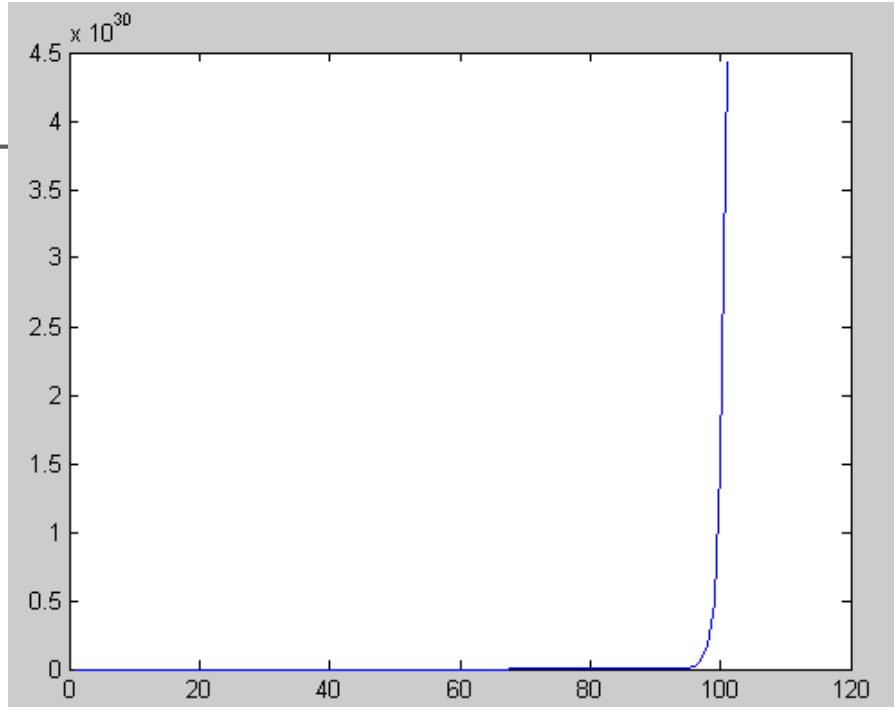
può essere generata con le seguenti relazioni ricorrenti:

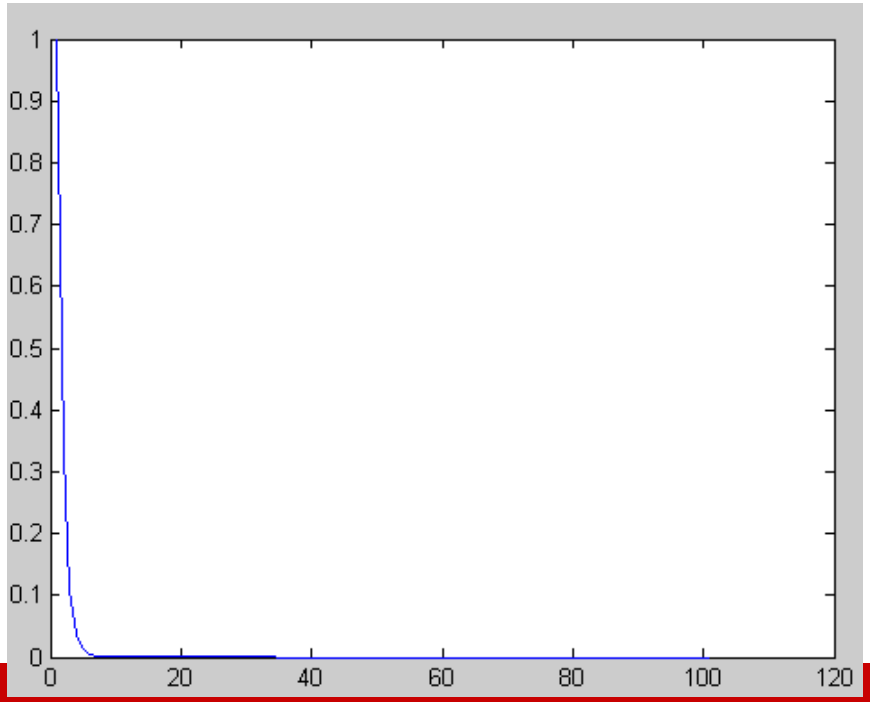
$$(1)\begin{cases} p_n = \frac{10}{3} p_{n-1} - p_{n-2} \\ p_0 = 1; p_1 = \frac{1}{3} \end{cases} \qquad (2)\begin{cases} p_n = \frac{1}{3} p_{n-1} \\ p_0 = 1 \end{cases}$$

Generare i primi 100 termini della successione

Relazione (1)

```
p1(1)=1;p1(2)=1/3;
for i=2:100
 p1(i+1)=10/3*p1(i)-p1(i-1)
end
plot(1:101,p1)
```





Relazione (2)

```
p2(1)=1;
for i=1:100
 p2(i+1)=1/3*p2(i);
end
plot(1:101,p2)
```

Analisi della propagazione dell'errore

$$\tilde{p}_0 = p_0 + \varepsilon, \quad \tilde{p}_1 = p_1 + \varepsilon,$$

$$\tilde{p}_2 = \frac{10}{3}\tilde{p}_1 - \tilde{p}_0 = \frac{10}{3}(p_1 + \varepsilon) - (p_0 + \varepsilon) = p_2 + \frac{7}{3}\varepsilon$$

$$\tilde{p}_3 = \frac{10}{3}\tilde{p}_2 - \tilde{p}_1 = \frac{10}{3}(p_2 + \frac{7}{3}\varepsilon) - (p_1 + \varepsilon) = p_3 + \frac{61}{9}\varepsilon$$

$$\tilde{p}_4 = \frac{10}{3}\tilde{p}_3 - \tilde{p}_2 = \frac{10}{3}(p_3 + \frac{61}{9}\varepsilon) - (p_2 + \varepsilon) = p_4 + \frac{583}{27}\varepsilon$$

.....

$$\tilde{p}_0 = p_0 + \varepsilon,$$

$$\tilde{p}_1 = \frac{1}{3}\tilde{p}_0 = \frac{1}{3}(p_0 + \varepsilon) = p_1 + \frac{1}{3}\varepsilon$$

$$\tilde{p}_2 = \frac{1}{3}\tilde{p}_1 = \frac{1}{3}(p_1 + \frac{1}{3}\varepsilon) = p_2 + \frac{1}{9}\varepsilon$$

$$\tilde{p}_3 = \frac{1}{3}\tilde{p}_2 = \frac{1}{3}(p_2 + \frac{1}{9}\varepsilon) = p_3 + \frac{1}{27}\varepsilon$$

.....

Bontà di un algoritmo

1. Generale e robusto: applicabile ad un qualsiasi insieme di dati di un certo dominio
2. Semplicità di verifica delle ipotesi di applicazione
3. Stabilità numerica
4. Richiesta di risorse
   - Numero di operazioni
   - Quantità di memoria richiesta

Costo computazionale

Complessità computazionale di un algoritmo= **numero di operazioni aritmetiche floating point richieste per la sua esecuzione**

Unità di misura
FLOP (floating point operation)

1 flop = una operazione elementare (+,-,*,/)

Esempio 1: sistemi lineari

Sistemi lineari: **Ax=b** n numero incognite

Ipotesi:

Tempo di esecuzione di 1 moltiplicazione circa 10^{-7} secondi

Regola di Cramer **O((n+1)!)**

(soluzione come quozienti di determinanti di ordine n)

Metodo di Gauss **$O(n^3)$**

n	Metodo di CRAMER	Metodo di GAUSS
10	4 secondi	3.3×10^{-5}sec
14	36 ore	9.1×10^{-5}sec
18	386 anni	1.9×10^{-4}sec

Esempio 2

Calcolo matrice A per vettore x

$A=[a_{ij}]$, i=1,..,m, j=1,..,n $x=[x_j]$,j=1,..,n

$$y = A \cdot x \qquad y_i = \sum_{j=1}^{n} a_{ij} x_j \quad i = 1,..,m$$

Algoritmo

```
For i=1:m
        y(i)=A(i,:)*x';
end
```

Ogni prodotto scalare costa:
n moltiplicazioni
n addizioni

Costo computazionale 2n*m. Se n=m allora $O(n^2)$

Esempio 3: calcolo di un polinomio in un punto

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

Algoritmo 1

$$p_0 = a_0; \quad r_0 = 1$$

$$\left.\begin{array}{c} r_i = r_{i-1} x \\ p_i = a_i r_i + p_{i-1} \\ p(x) = p_n \end{array}\right\} \quad \text{i=1,2,...,n}$$

2n moltiplicazioni
n addizioni

Esempio 3: calcolo di un polinomio in un punto

Metodo di Ruffini-Horner

$$p(x) = a_0 + x(a_1 + x(a_2 + ... + x(a_{n-1} + a_n x)....))$$

Esempio

$$p(x) = 1 + x + 3x^2 - 6x^3 \quad \rightarrow \quad p(x) = 1 + x(1 + x(3 - 6x))$$

Algoritmo 2

$$p_0 = a_n$$

$$p_i = p_{i-1}x + a_{n-i} \qquad i=1,2,\ldots,n$$

$$p(x) = p_n$$

n moltiplicazioni
n addizioni

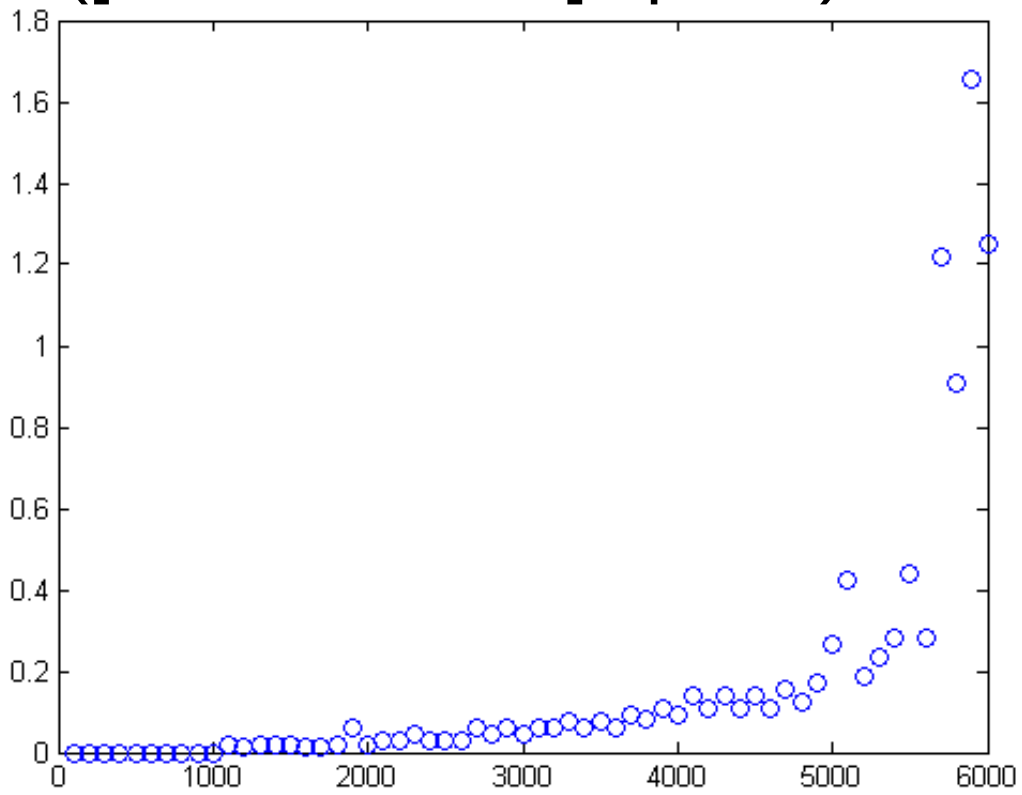
Performance di un programma

Tempo di CPU: tempo impiegato dall'unità centrale per eseguire un determinato programma

ESEMPIO: prodotto matrice per vettore: **y=Av** con **A** di dimensione n. L'algoritmo richiede $O(n^2)$ operazioni.

```
for n=100:100:6000
      A=rand(n,n);v=rand(n,1);
      t=cputime;
      A*v;
      cpu(n/100)=cputime-t;
end
```

plot([100:100:6000],cpu,'o')



Tempo di CPU (in sec.) necessario per eseguire A*v su PC a 1GHz.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Serena Morigi

Dipartimento di Matematica

serena.morigi@unibo.it

http://www.dm.unibo.it/~morigi