



Consegna 5 relazioni (una per coppia) :

Possibilmente (verrà premiata la puntualità; comunque prima dell'esame orale) :

- le *prime 2* entro il 9 novembre 2009;
- le *seconde 3* entro il 14 dicembre 2009 .

N.B. : l'esame finale (colloquio orale sulle relazioni) è comunque individuale !

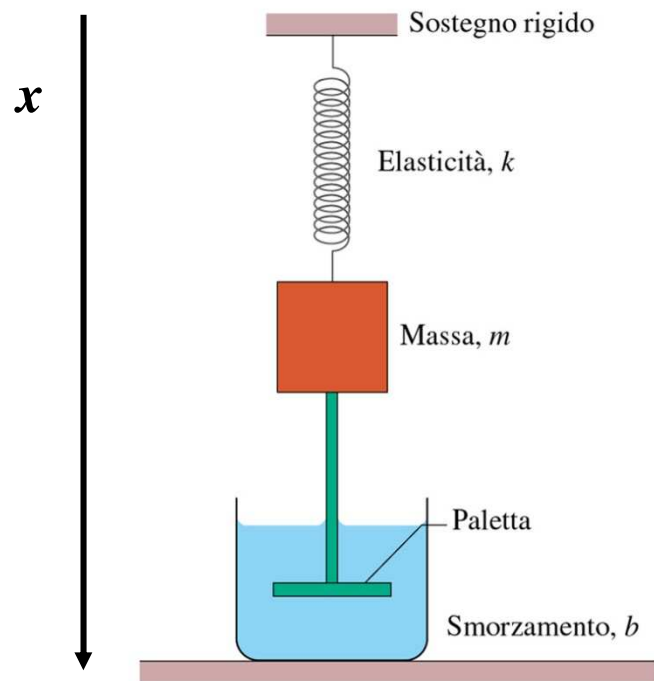
Metodi Computazionali della Fisica



Equazioni differenziali ordinarie
(Ordinary Differential Equations, ODE)

Problema da risolvere

Molte leggi fisiche sono formulate in termini di **equazioni differenziali**, es. **oscillatore armonico smorzato**:



$$ma = -kx - bv, \quad v = \frac{dx}{dt}, \quad a = \frac{d^2x}{dt^2} \Rightarrow$$
$$\Rightarrow m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = 0$$

Problema da risolvere

- Risolvere **numericamente** equazioni differenziali è una delle operazioni più frequenti quando si vuol descrivere sistemi fisici mediante **modelli**
- La forma più generale di un'equazione differenziale ordinaria (**ODE**) è un'insieme di M equazioni **al prim'ordine**, accoppiate:

$$\frac{d\vec{y}}{dx} = \vec{f}(x, \vec{y})$$

Problema da risolvere

Nell'equazione
$$\frac{d\vec{y}}{dx} = \vec{f}(x, \vec{y}) \quad (1)$$

x è la variabile **indipendente**, \vec{y} è un'insieme di M variabili **dipendenti** e \vec{f} è in generale un vettore di M componenti

N.B. equazioni differenziali di ordine **superiore** possono essere espresse nella forma (1) introducendo opportune **funzioni ausiliarie**

Problema da risolvere

Esempio: moto in **1D** di una particella di massa m , sottoposta ad un campo di forza $F(x)$ (equazione differenziale al **second'ordine**):

$$ma = m \frac{d^2 x}{dt^2} = F(x) \quad (2)$$

definendo il **momento** (o **quantità di moto**, funzione ausiliaria):

$$p(t) = mv = m \frac{dx}{dt}$$

Problema da risolvere

allora la (2) è equivalente all'insieme delle 2 equazioni differenziali al **prim'**ordine (**ODE**) (equazioni di Hamilton) accoppiate:

$$\frac{dx}{dt} = \frac{p}{m} \qquad \frac{dp}{dt} = F(x)$$

che sono proprio nella forma (1); allora è **sufficiente** considerare in dettaglio solo i metodi che si applicano alle equazioni differenziali al **primo** ordine (**ODE**)

Problema da risolvere

Per semplicità (la generalizzazione non è difficile)
consideriamo solo il caso particolare in cui ci sia una
singola variabile dipendente $y(x)$:

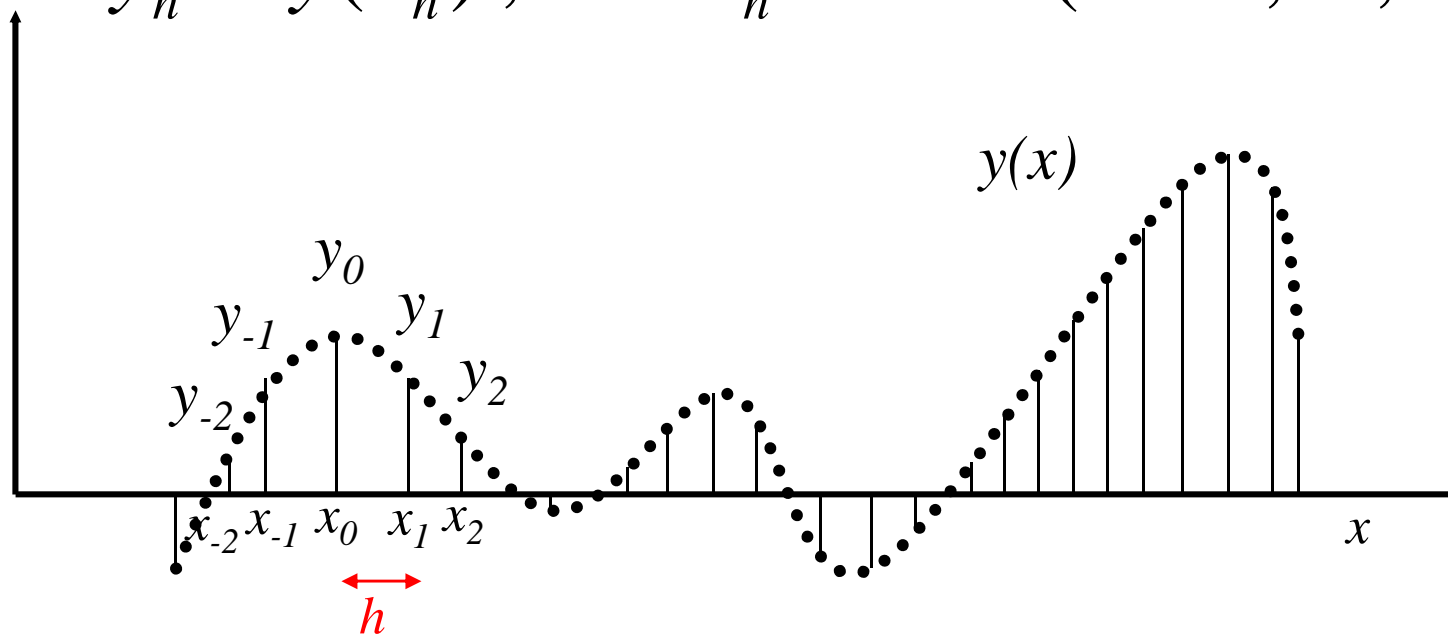
$$\frac{dy}{dx} = f(x, y) \quad (3)$$

Obiettivo: trovare $y(x)$, che soddisfa la (3), dato il valore di y in qualche punto iniziale, ad esempio $y(x=0)=y_0$;
questo è, ad esempio, il caso quando sono dati la **posizione** ed il **momento iniziali** di una particella e si vuole trovare il moto negli istanti successivi usando le equazioni di Hamilton scritte precedentemente.

Derivazione numerica

Supponiamo di voler calcolare la **derivata prima** di una data funzione $y(x)$ per $x=x_0=0$ (la generalizzazione ad un punto qualsiasi è banale), $y'(x)$, supponendo di conoscere y su di una **griglia equispaziata** di valori di x :

$$y_n = y(x_n), \quad x_n = nh \quad (n = 0, \pm 1, \pm 2, \dots)$$



Derivazione numerica

L'**obiettivo** è quello di calcolare un valore **approssimato** di $y'(0)$ in termini dei valori $\{y_n\}$; cominciamo con usare la **serie di Taylor** per espandere y vicino a $x=0$:

$$y(x) = y_0 + x y' + \frac{x^2}{2!} y'' + \frac{x^3}{3!} y''' + \dots$$

dove tutte le derivate sono calcolate per $x=0$; è facile verificare che:

$$y_{\pm 1} \equiv y(x = \pm h) = y_0 \pm h y' + \frac{h^2}{2} y'' \pm \frac{h^3}{6} y''' + \mathcal{O}(h^4) \quad (1)$$

$$y_{\pm 2} \equiv y(x = \pm 2h) = y_0 \pm 2h y' + 2h^2 y'' \pm \frac{4h^3}{3} y''' + \mathcal{O}(h^4) \quad (2)$$

Derivazione numerica

sottraendo y_{-1} da y_1 si ha:

$$y' = \frac{y_1 - y_{-1}}{2h} - \frac{h^2}{6} y''' + \mathcal{O}(h^4)$$

e quindi si ottiene l'approssimazione “a 3 punti”:

$$y' \approx \frac{y_1 - y_{-1}}{2h} \quad (3)$$

la (3) sarebbe **esatta** se y fosse un polinomio di **secondo grado** nell'intervallo $[-h, h]$, poichè allora la derivata terza e quelle di ordine più alto sarebbero nulle, quindi la (3) assume che sia **valida** un'interpolazione polinomiale **quadratica** nei 3 punti $x = -h, 0, h$ (ovviamente l'accuratezza dell'approssimazione (3) aumenta col **diminuire** del “passo” h).

Derivazione numerica

N.B. la (3) (“**simmetrica**” o “**centrale**”) è **più accurata**, di un ordine in h , rispetto alle formule alternative “**forward difference**” (differenza “in avanti”) o “**backward difference**” (differenza “all’indietro”):

$$y' = \frac{y_1 - y_0}{h} + \mathcal{O}(h)$$

$$y' = \frac{y_0 - y_{-1}}{h} + \mathcal{O}(h)$$

queste formule “**a 2 punti**” sono basate sull’assunzione che y sia ben approssimata da una funzione **lineare** negli intervalli tra $x=0$ e $x=h$ e tra $x=-h$ e $x=0$, rispettivamente.

Derivazione numerica

Esempio: consideriamo il problema di calcolare **numericamente** $y'(x=1)$, con $y(x)=\sin(x)$ e x in radianti, usando le formule precedenti; ovviamente la soluzione **esatta** è $\cos(1)=0.540302$

Derivazione numerica

La seguente **tabella** riporta i risultati prodotti da un tale programma, per vari valori di h (usando variabili in **singola precisione** per mettere in evidenza l'errore numerico), confrontati con quelli ottenuti con le formule “a 2 punti”:

h	symmetric	forward	backward
	3-point	2-point	2-point
0.50000	0.022233	0.228254	-0.183789
0.20000	0.003595	0.087461	-0.080272
0.10000	0.000899	0.042938	-0.041139
0.05000	0.000225	0.021258	-0.020808
0.02000	0.000037	0.008453	-0.008380
0.01000	0.000010	0.004224	-0.004204
0.00500	0.000010	0.002108	-0.002088
0.00200	-0.000014	0.000820	-0.000848
0.00100	-0.000014	0.000403	-0.000431
0.00050	0.000105	0.000403	-0.000193
0.00020	-0.000163	-0.000014	-0.000312
0.00010	-0.000312	-0.000312	-0.000312
0.00005	0.000284	0.001476	-0.000908

Derivazione numerica

Come si può osservare il risultato **migliora** al **diminuire** di h , ma solo **fino ad un certo punto**, dopodichè la situazione peggiora; questo è dovuto alla precisione limitata dell'aritmetica su computer (in **singola precisione** 6-7 cifre decimali) e al fatto che, quando si calcolano le differenze nei numeratori delle formule precedenti, queste sono soggette ad **errori di arrotondamento** grandi se h è piccolo e quindi y_1 ed y_{-1} sono **quasi uguali** ;

ad **esempio** (con 6 cifre significative), se $h=10^{-6}$ allora:

$$y_1 = \sin(1.000001) = 0.841472$$

$$y_{-1} = \sin(0.999999) = 0.841470$$

$$\Rightarrow y_1 - y_{-1} = 0.000002 \Rightarrow y' \approx 1.000000$$

che è una stima **pessima** ($y'(\text{esatto})=0.540302$) !

Derivazione numerica



Invece con **10** cifre significative (ad esempio in **doppia precisione**):

$$y_1 = 0.8414715251$$

$$y_{-1} = 0.8414704445$$

$$\Rightarrow y' \approx 0.540300$$

che è un'**ottima** stima ($y'(\text{esatto})=0.540302$) !

N.B. la derivazione numerica è un processo intrinsecamente **instabile** (non esiste un limite ben definito per $h \rightarrow 0$) e perciò deve essere usata con **attenzione** !

Derivazione numerica

Formule per calcolare derivate di **ordine superiore** possono essere costruite utilizzando sempre opportune combinazioni della (1) e (2); ad esempio è facile vedere che:

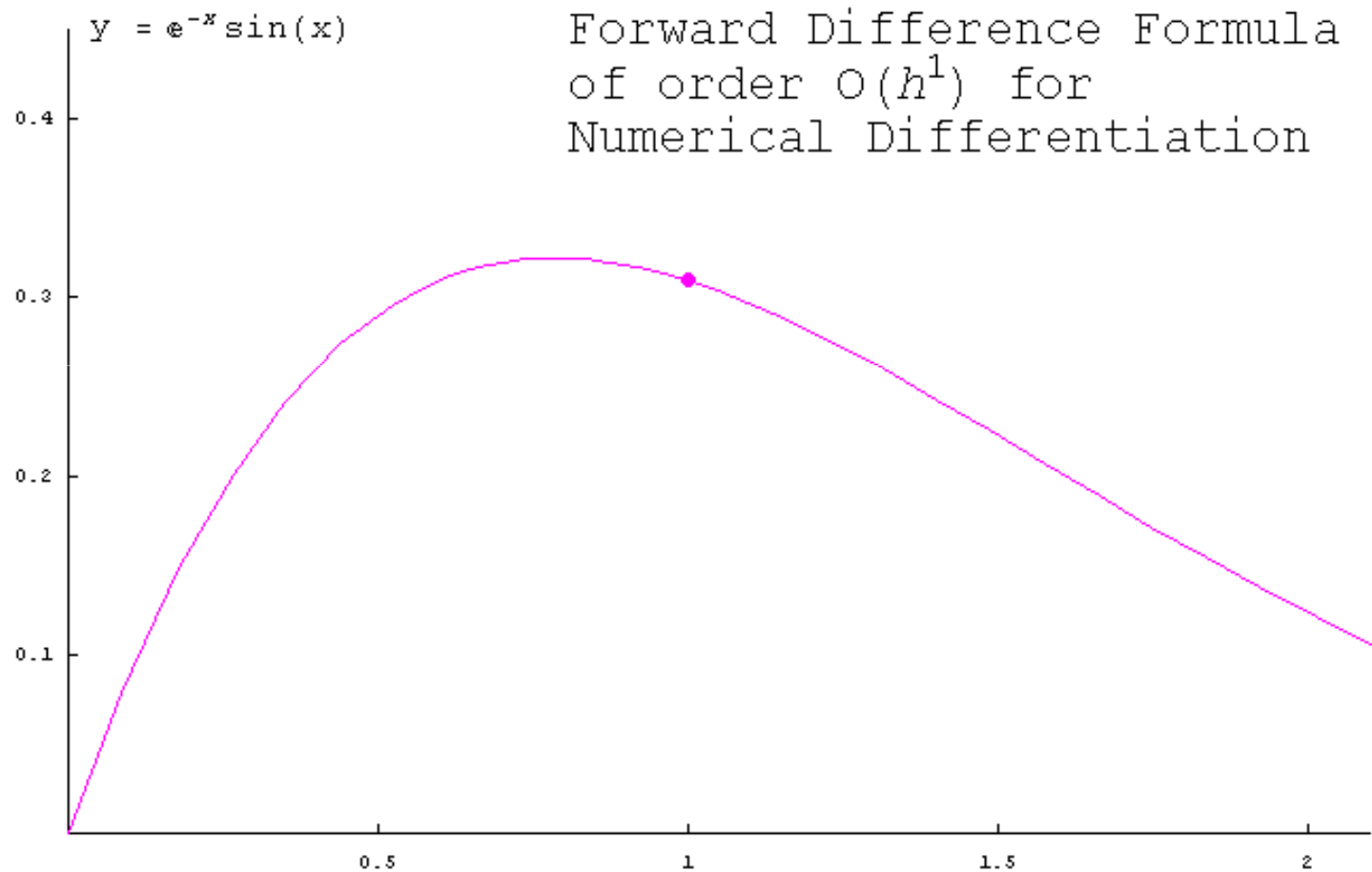
$$y_1 - 2y_0 + y_{-1} = h^2 y'' + \mathcal{O}(h^4)$$

allora un'approssimazione di ordine $\mathcal{O}(h^2)$ per la derivata **seconda** è:

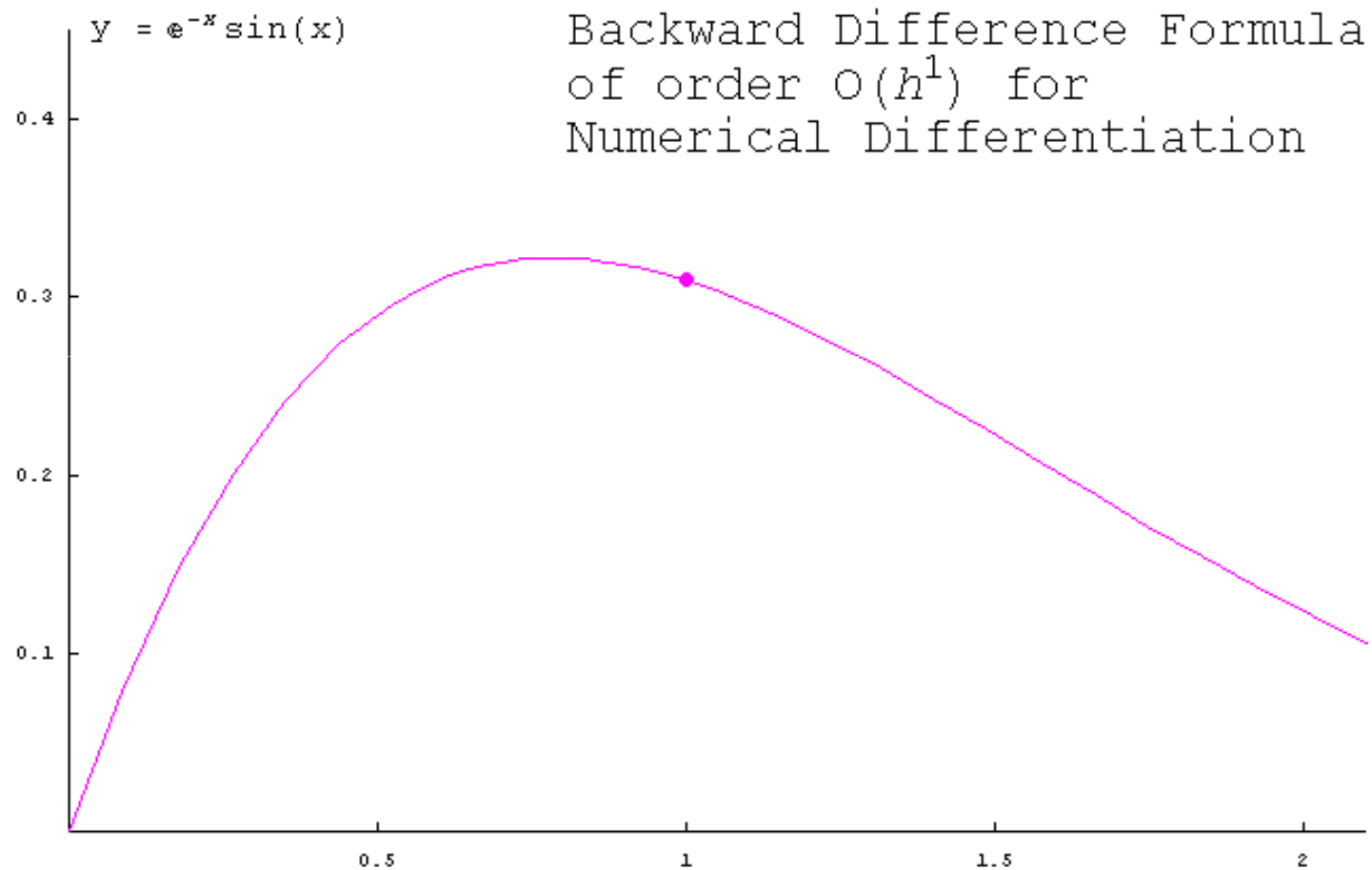
$$y'' \approx \frac{y_1 - 2y_0 + y_{-1}}{h^2}$$

e analogamente per derivate di ordine **superiore**.

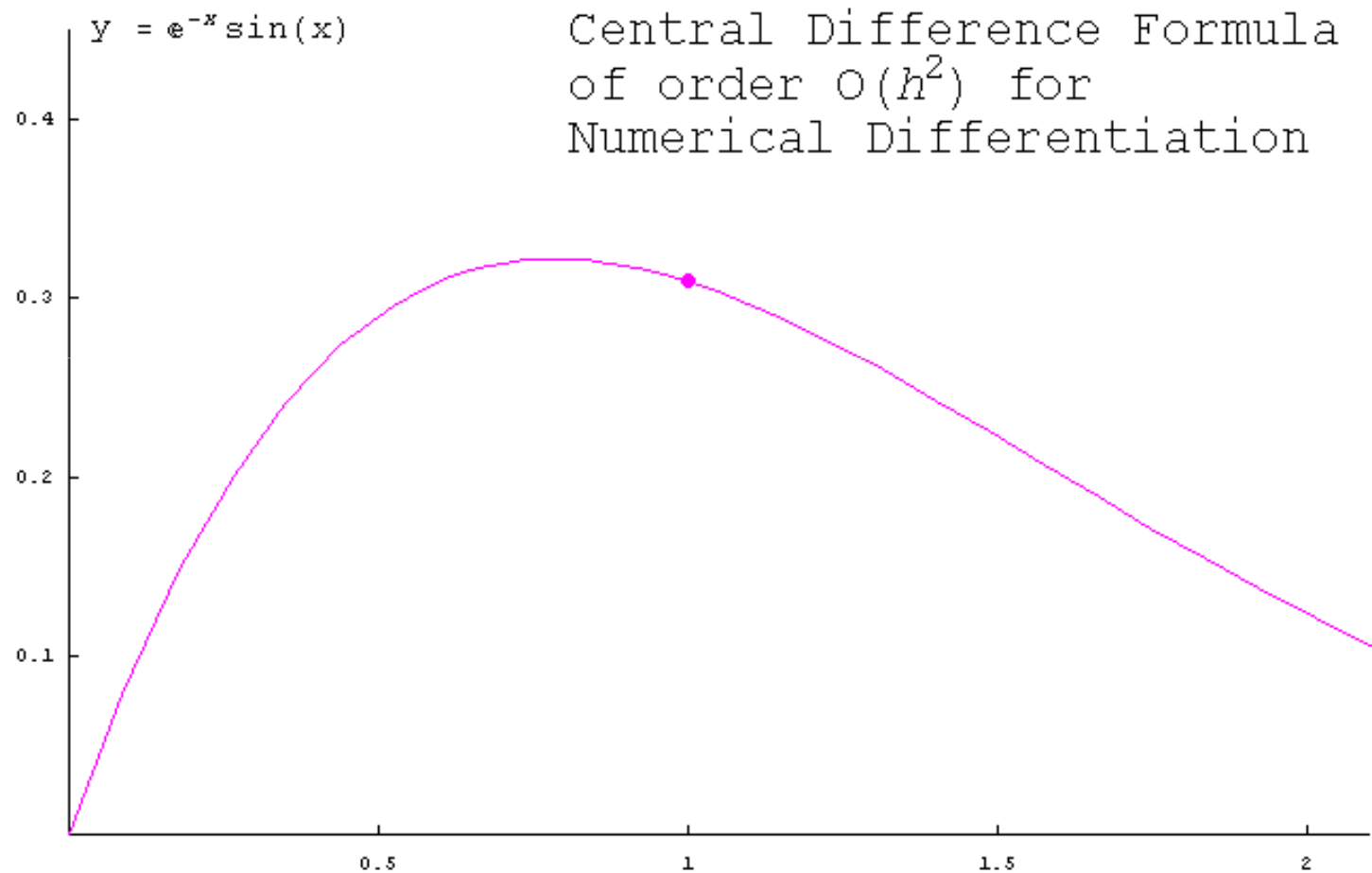
Derivazione numerica



Derivazione numerica



Derivazione numerica



Problema da risolvere

Torniamo al nostro problema originale in cui, per semplicità (la generalizzazione non è difficile), consideriamo solo il caso particolare in cui ci sia una **singola** variabile dipendente $y(x)$:

$$\frac{dy}{dx} = f(x, y) \quad (3)$$

Obiettivo: trovare $y(x)$, che soddisfa la (3), dato il valore di y in qualche punto iniziale, ad esempio $y(x=0)=y_0$;

Metodo di Eulero

rappresenta uno degli algoritmi **più semplici** per trovare la soluzione dell'ODE:

$$\frac{dy}{dx} = f(x, y)$$

con la condizione iniziale $y(x=0)=y_0$

Se vogliamo trovare il valore di y per un particolare valore di x , ad esempio $x=1$, allora la **strategia generale** è la seguente:

Metodo di Eulero


- **suddividere** l'intervallo $[0, 1]$ in un numero (grande) N di **sottointervalli** equispaziati di lunghezza $h=1/N$
- sviluppare una **formula ricorsiva** che stabilisca una relazione tra y_n e $\{y_{n-1}, y_{n-2}, \dots\}$, essendo y_n l'approssimazione per $y(x_n=nh)$

La formula ricorsiva consentirà allora **un'integrazione** “**passo-dopo-passo**” dell'ODE da $x=0$ a $x=1$; in particolare si considera l'ODE:

$$\frac{dy}{dx} = f(x, y) \quad (3)$$

Metodo di Eulero

nel generico punto x_n , sostituendo la derivata dy/dx con l'approssimazione alle differenze finite “**in avanti**” (“forward difference approximation”) $(y_{n+1}-y_n)/h$, allora la (3) diventa:

$$\frac{y_{n+1} - y_n}{h} + \vartheta(h) = f(x_n, y_n)$$
 **errore**

dalla quale si ricava subito la **formula ricorsiva**:

$$y_{n+1} = y_n + h f(x_n, y_n) + \vartheta(h^2)$$

Metodo di Eulero

N.B. l'errore locale (nel singolo passo da y_n a y_{n+1}) è $\theta(h^2)$, ma **l'errore globale** (per ottenere $y(1)$ compiendo N passi per integrare da $x=0$ a $x=1$) è $N\theta(h^2)=\theta(h)$, cioè l'errore diminuisce solo **linearmente** al diminuire di h : per dimezzare l'errore nel risultato finale $y(1)$ è necessario usare $h'=h/2$ e $N'=2N$; notare che ad ogni passo il **costo numerico** è essenzialmente un **singolo calcolo** di f .

Esempio:
$$\frac{dy}{dx} = -xy \quad y(0) = 1 \quad (4)$$

in questo caso esiste la soluzione **analitica**: $y = e^{-x^2/2}$

Metodo di Eulero

La seguente **tabella** riporta gli **errori** relativi a

$y(1)=e^{-1/2}=0.606531$ e $y(3)=e^{-9/2}=0.011109$, per vari valori di h :

h	y(1)	y(3)
0.500	-0.143469	0.011109
0.200	-0.046330	0.006519
0.100	-0.021625	0.003318
0.050	-0.010453	0.001665
0.020	-0.004098	0.000666
0.010	-0.002035	0.000333
0.005	-0.001014	0.000167

come previsto l'**errore** diminuisce **linearmente** diminuendo h , tuttavia **l'errore relativo** (=errore diviso per il valore di y) **aumenta** con x , poiché, a parità di h , **aumenta** il numero di passi N e, inoltre, y diventa **più piccolo**

N.B. una “**misura**” dell'errore relativo dell'algoritmo si può ottenere usando il valore **finale** di y come condizione **iniziale** e integrando “all'indietro” (“backward”) dal valore finale di x al punto iniziale: la **discrepanza** tra il valore risultante di y e quello iniziale originale dà una stima dell'errore.

Metodo di Eulero

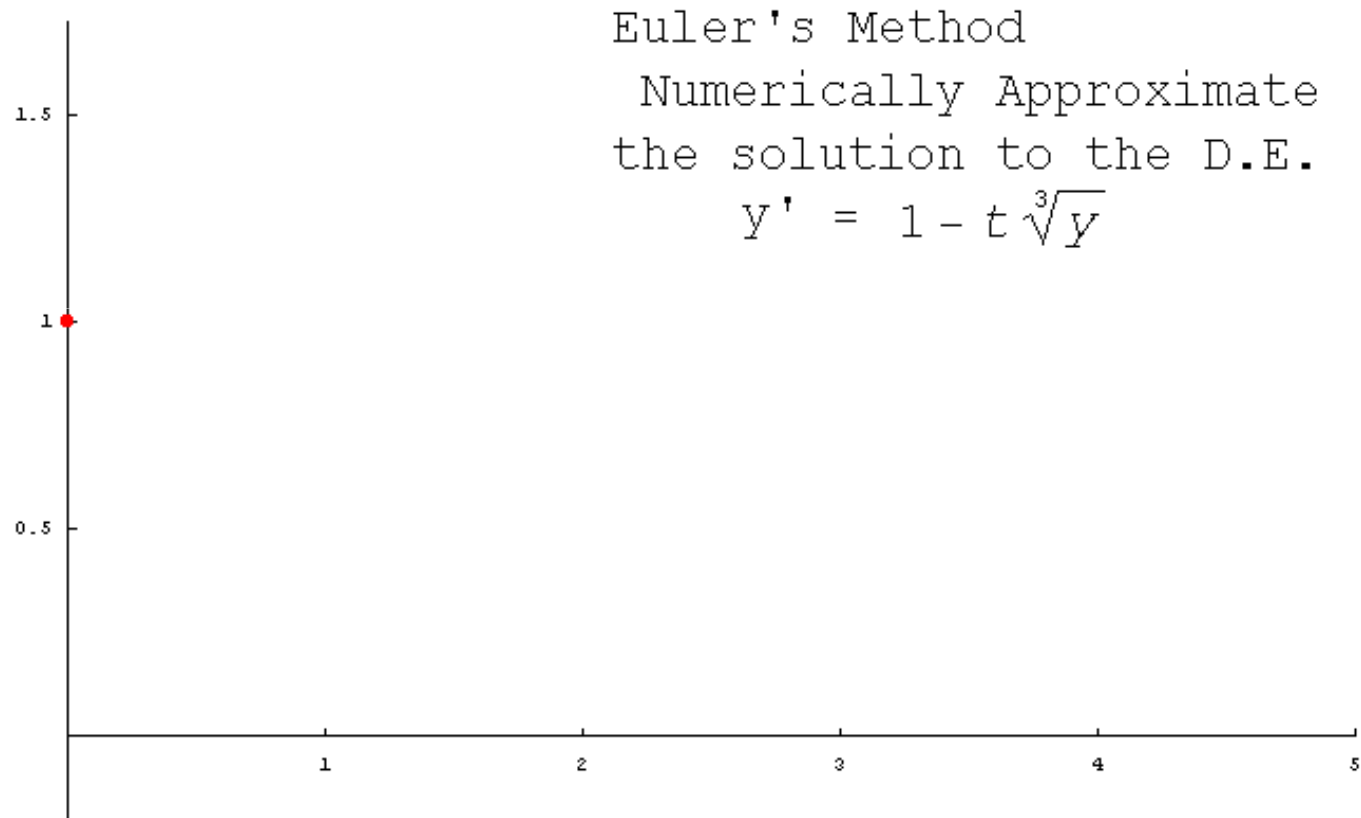
Il metodo di Eulero in generale non è soddisfacente a causa della sua **bassa accuratezza**.

In linea di principio si può ridurre h se x aumenta, però in questo modo si **perde** rapidamente in efficienza.

Esempio:
$$\frac{dy}{dt} = 1 - t \sqrt[3]{y} \quad y(0) = 1$$

per t tra 0 e 5, cambiando il numero N di sottointervalli, cioè il valore di h

Metodo di Eulero



Metodo di Taylor

Una classe di metodi **più efficienti** e **più accurati** del metodo di Eulero possono essere ricavati considerando l'**espansione** in serie di Taylor:

$$y_{n+1} = y(x_n + h) = y_n + h y'_n + \frac{1}{2} h^2 y''_n + \mathcal{O}(h^3) \quad (5)$$

dove (3) implica che: $y'_n = f(x_n, y_n) \quad (6)$

$$\text{e } y''_n = \frac{df(x_n, y_n)}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \frac{dy}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f \quad (7)$$



derivata parziale

Metodo di Taylor

Sostituendo (6) e (7) in (5) si trova che :

$$y_{n+1} = y_n + hf + \frac{1}{2}h^2 \left[\frac{\partial f}{\partial x} + f \frac{\partial f}{\partial y} \right] + \mathcal{O}(h^3) \quad (8)$$

dove f e le sue derivate vanno calcolate per (x_n, y_n)

N.B. la relazione **ricorsiva** (8) ha un **errore locale** $\theta(h^3)$ e quindi un **errore globale** $\theta(h^2)$, cioè è un “ordine” più accurata del metodo di Eulero; il metodo di Taylor è utile soprattutto quando f è **noto analiticamente** ed è abbastanza **semplice** per fare le **derivate**; in linea di principio si possono ottenere algoritmi ancora più accurati considerando **ulteriori termini** nello sviluppo di Taylor, però la **complessità** dell'algoritmo diventa rapidamente proibitiva !

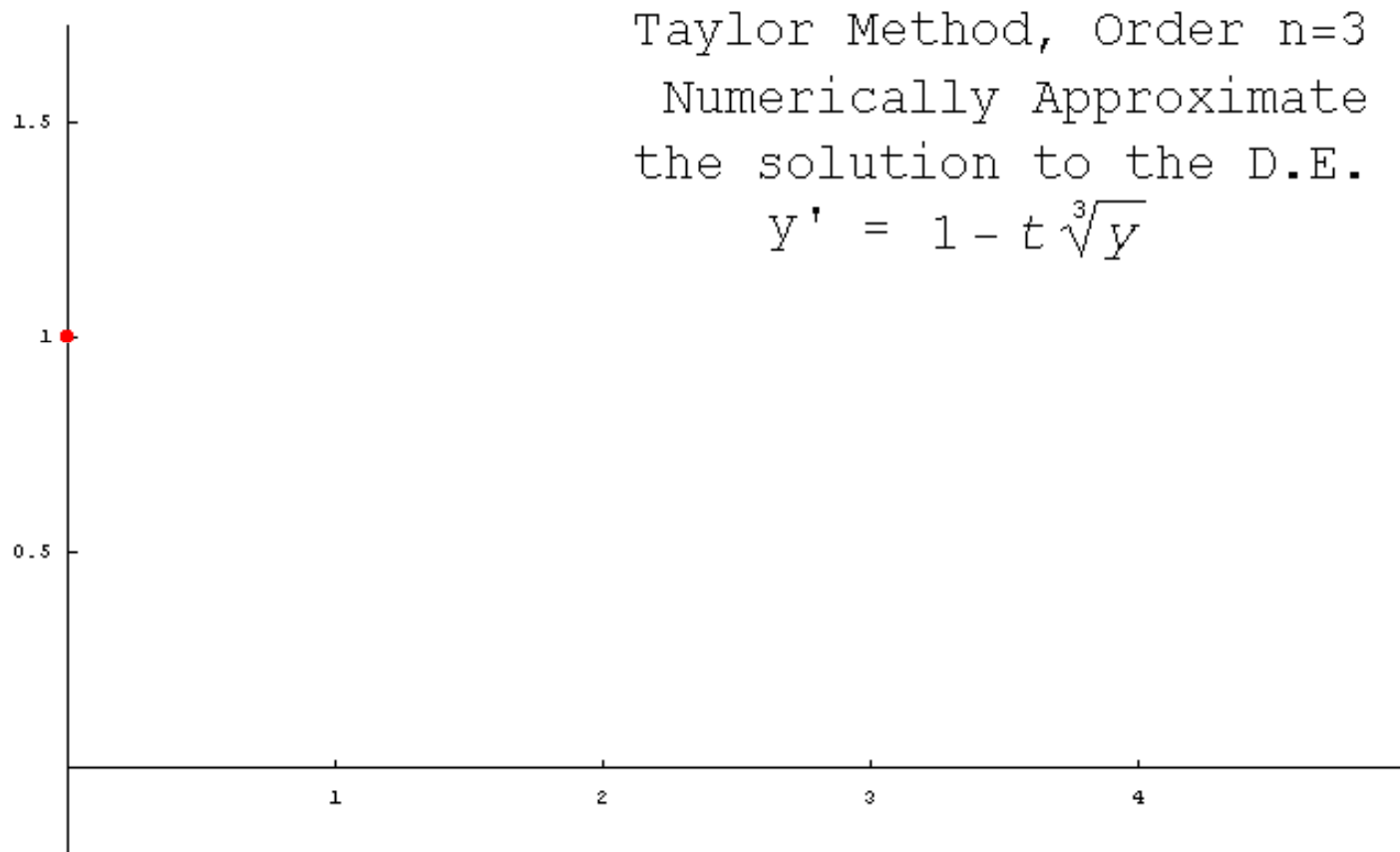
Metodo di Taylor

La seguente **tabella** riporta gli **errori** relativi a

$y(1)=e^{-1/2}=0.606531$ e $y(3)=e^{-9/2}=0.011109$, per vari valori di h :

Eulero			Taylor	
h	y(1)	y(3)	y(1)	y(3)
0.500	-0.143469	0.011109	0.032312	-0.006660
0.200	-0.046330	0.006519	0.005126	-0.000712
0.100	-0.021625	0.003318	0.001273	-0.000149
0.050	-0.010453	0.001665	0.000317	-0.000034
0.020	-0.004098	0.000666	0.000051	-0.000005
0.010	-0.002035	0.000333	0.000013	-0.000001
0.005	-0.001014	0.000167	0.000003	0.000000

Metodo di Taylor



Metodi “impliciti”

I metodi discussi finora sono tutti “**espliciti**”, nel senso che y_{n+1} è dato direttamente in termini del **già noto** valore di y_n ; un altro modo per ottenere un’elevata accuratezza è rappresentato dai **metodi “impliciti”** nei quali si deve risolvere un’equazione per determinare y_{n+1} :

Consideriamo sempre l’ODE: $\frac{dy}{dx} = f(x, y)$ nel punto $x_{n+1/2} \equiv (n + \frac{1}{2})h$

$$\left. \frac{dy}{dx} \right|_{x_{n+1/2}} = f(x_{n+1/2}, y_{n+1/2}) \text{ , allora:}$$

$$\frac{y_{n+1} - y_n}{2 \cdot (\frac{1}{2}h)} + \mathcal{O}(h^2) = \frac{1}{2} [f_n + f_{n+1}] + \mathcal{O}(h^2)$$

Metodi “impliciti”

che implica la relazione **ricorsiva implicita**:

$$y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] + \mathcal{O}(h^3) \quad (9)$$

allora bisogna risolvere un'equazione (in genere non banale) ad **ogni** step (può essere molto **costoso** !);

si ha una notevole **semplificazione** se f è **lineare** in y , cioè si può scrivere $f(x, y) = g(x) \cdot y$, allora la (9) si risolve facilmente:

$$y_{n+1} = \left[\frac{1 + \frac{1}{2} g(x_n) h}{1 - \frac{1}{2} g(x_{n+1}) h} \right] y_n$$

Metodi “impliciti”

ad esempio, applicando questo metodo al nostro caso:

$$\frac{dy}{dx} = -xy$$

abbiamo:

$$g(x) = -x$$

e si ottiene la seguente tabella , dalla quale si evince chiaramente un andamento dell'**errore quadratico** in h :

Metodi “impliciti”

La seguente **tabella** riporta gli **errori** relativi a

$y(1)=e^{-1/2}=0.606531$ e $y(3)=e^{-9/2}=0.011109$, per vari valori di h :

Eulero			Taylor		implicit	
h	y(1)	y(3)	y(1)	y(3)	y(1)	y(3)
0.500	-0.143469	0.011109	0.032312	-0.006660	-0.015691	0.001785
0.200	-0.046330	0.006519	0.005126	-0.000712	-0.002525	0.000255
0.100	-0.021625	0.003318	0.001273	-0.000149	-0.000631	0.000063
0.050	-0.010453	0.001665	0.000317	-0.000034	-0.000157	0.000016
0.020	-0.004098	0.000666	0.000051	-0.000005	-0.000025	0.000003
0.010	-0.002035	0.000333	0.000013	-0.000001	-0.000006	0.000001
0.005	-0.001014	0.000167	0.000003	0.000000	-0.000001	0.000000

Metodi “Runge-Kutta”

Rappresentano una classe di metodi particolarmente **convenienti** e largamente **usati**.

Deriviamo **esplicitamente** la versione “**al second’ordine**” per illustrare lo **spirito** dell’approccio, e diamo solo i risultati per le versioni “**al terzo ordine**” e “**quarto ordine**”, che sono quelle più comunemente **usate**; consideriamo l’equazione:

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(x, y) dx$$

e approssimiamo f nell’integrale con la sua espansione in serie di Taylor attorno al **punto di mezzo** dell’intervallo di integrazione:

$$y_{n+1} = y_n + h f(x_{n+1/2}, y_{n+1/2}) + \mathcal{O}(h^3) \quad (11)$$

Metodi “Runge-Kutta”

In linea di principio bisognerebbe conoscere $y_{n+1/2}$, tuttavia, poiché l'errore nella (11) è già $\theta(h^3)$, si può scrivere:

$$y_{n+1} = y_n + h f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k\right) + \mathcal{O}(h^3) \quad (12)$$

dove: $k = hf(x_n, y_n)$, infatti, usando ad esempio la formula di

Eulero: $y_{n+1/2} = y_n + \frac{h}{2} f(x_n, y_n + \frac{k}{2}) + \mathcal{O}(h^2)$ ←

e l'idea è quella di sostituire **approssimazioni** per i valori di y nella **parte destra** di espressioni **implicite** che coinvolgono f .

La (12) ha la **stessa accuratezza** del metodo di **Taylor** o dei metodi “**impliciti**” $\theta(h^3)$, ma ha il **vantaggio** di **non** richiedere **proprietà speciali** per la funzione f , come la facile **differenziabilità** o la **linearità** in y , tuttavia la (12) richiede il calcolo di f **2 volte** per ogni step.

Metodi “Runge-Kutta”

Schemi Runge-Kutta di ordine **superiore** possono essere derivati analogamente; ad esempio, un algoritmo “al **terzo** ordine”, con un errore locale $\theta(h^4)$, si ottiene approssimando l'integrale:

$\int_{x_n}^{x_{n+1}} dx f(x, y)$ con una somma finita di valori di f , usando la regola di Simpson (richiede il calcolo di f **3 volte** per ogni step):

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 4k_2 + k_3) + \mathcal{O}(h^4)$$

$$k_1 = h f(x_n, y_n)$$

$$k_2 = h f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right)$$

$$k_3 = h f(x_n + h, y_n - k_1 + 2k_2)$$

Metodi “Runge-Kutta”

L'esperienza ha dimostrato che l'algoritmo di Runge-Kutta che offre il **miglior compromesso** tra **accuratezza** e **sforzo computazionale** è un algoritmo “al **quarto** ordine”, che richiede di valutare f **4 volte** per ogni step:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5)$$

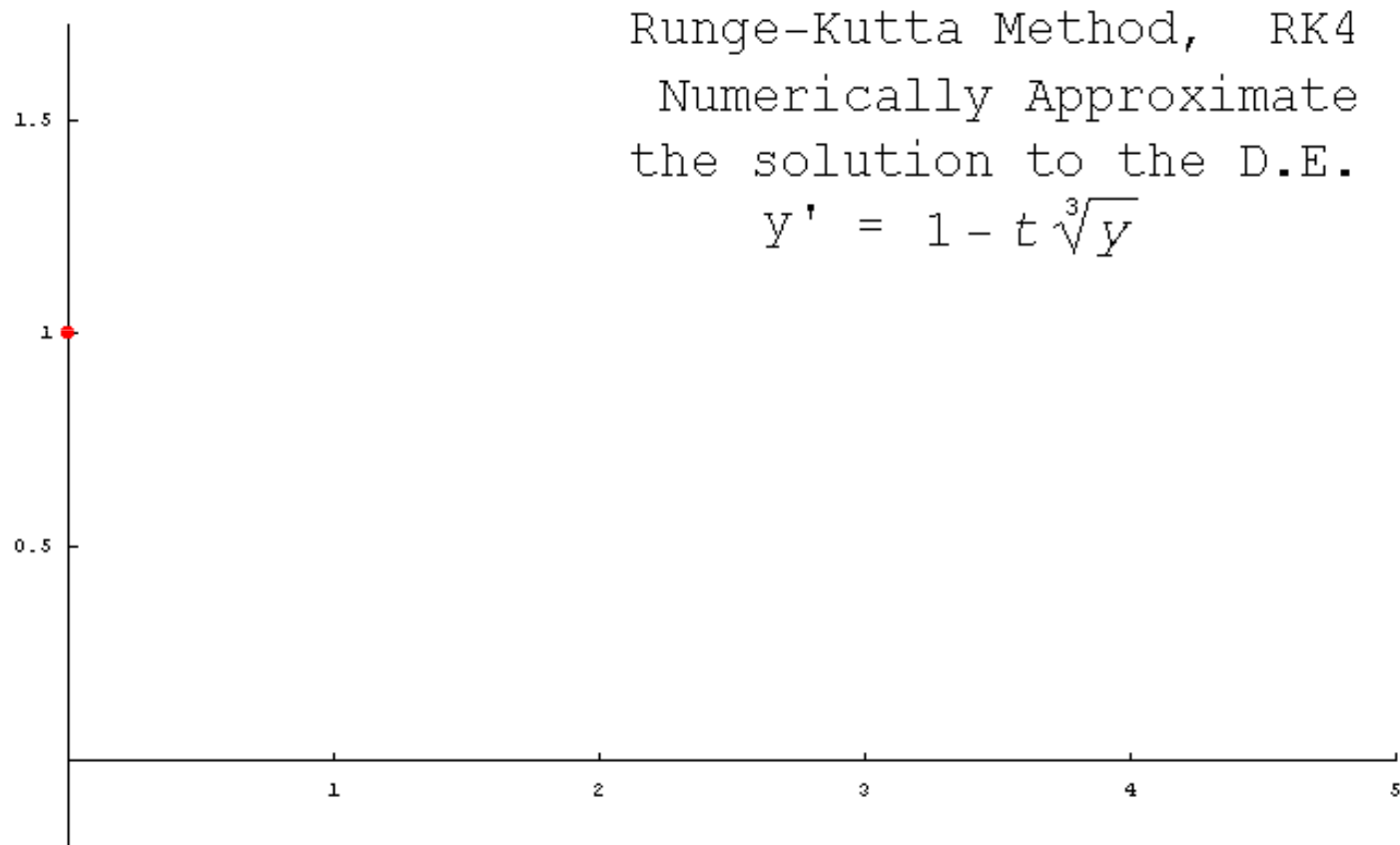
$$k_1 = h f(x_n, y_n)$$

$$k_2 = h f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right)$$

$$k_3 = h f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right)$$

$$k_4 = h f(x_n + h, y_n + k_3)$$

Metodi “Runge-Kutta”



Stabilità

Quando si integrano equazioni differenziali un aspetto **fondamentale** è rappresentato dalla **stabilità numerica** dell'algoritmo usato, cioè la misura in cui gli errori di **arrotondamento** o altri errori nel calcolo numerico possono essere **amplificati** fino a crescere talmente che alla fine il “**rumore**” diventi più grande del risultato.

Per illustrare questo problema cerchiamo di migliorare l'accuratezza del metodo di **Eulero**:

$$y_{n+1} = y_n + h f(x_n, y_n) + \mathcal{O}(h^2)$$

a tal fine approssimiamo la derivata in: $\frac{dy}{dx} = f(x, y)$

usando l'approssimazione **simmetrica**:

$$y'_n = \frac{y_{n+1} - y_{n-1}}{2h} + \mathcal{O}(h^2)$$

Stabilità

allora otteniamo la “**relazione ricorsiva a 3 termini**”:

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n) + \mathcal{O}(h^3) \quad (13)$$

che, **a prima vista**, sembrerebbe un algoritmo (“multistep”) altrettanto **valido** di altri caratterizzati da $\mathcal{O}(h^3)$, come il metodo di Taylor o qualche algoritmo “implicito”; tuttavia vediamo cosa succede quando applichiamo questo algoritmo al semplice caso:

$$\frac{dy}{dx} = -y \quad y(x=0) = 1$$

la cui **soluzione** analitica è: $y = e^{-x}$

per “iniziare” la relazione ricorsiva (13) abbiamo bisogno, oltre che di $y_0=1$, anche di y_1 ; y_1 può essere ottenuto ad esempio usando il metodo di Taylor:

Stabilità

$$y_{n+1} = y_n + hf + \frac{1}{2}h^2 \left[\frac{\partial f}{\partial x} + f \frac{\partial f}{\partial y} \right] + \mathcal{O}(h^3)$$

allora:
$$y_1 = y_0 - h + \frac{1}{2}h^2 + \mathcal{O}(h^3)$$

che non è nient'altro che la serie di Taylor per e^{-h}

e possiamo scrivere un **programma** numerico per utilizzare la (13); una parte dell'output del programma è riportato nella **tabella** successiva (prendendo $h=0.1$):

Stabilità

La seguente **tabella** riporta il valore esatto (analitico) e l'**errore** per l'integrazione di $dy/dx = -y$, con $y(0)=1$ (e $h=0.1$):

x	esatto	errore	x	esatto	errore	x	esatto	errore
0.2	0.818731	-0.000269	3.3	0.036883	-0.000369	5.5	0.004087	-0.001533
0.3	0.740818	-0.000382	3.4	0.033373	-0.000005	5.6	0.003698	0.001618
0.4	0.670320	-0.000440	3.5	0.030197	-0.000380	5.7	0.003346	-0.001858
0.5	0.606531	-0.000517	3.6	0.027324	0.000061	5.8	0.003028	0.001989
0.6	0.548812	-0.000538	3.7	0.024724	-0.000400	5.9	0.002739	-0.002257
			3.8	0.022371	0.000133	6.0	0.002479	0.002439

si può notare come, per **piccoli** valori di x , la soluzione numerica è solo leggermente più grande del valore esatto, l'errore essendo consistente con la stima $\theta(h^3)$; tuttavia, attorno a $x=3.5$, comincia a svilupparsi un'**oscillazione** nella soluzione numerica, che diventa, alternativamente, **maggiore** e **minore** del valore esatto; tale oscillazione diventa **sempre più grande** all'aumentare di x finchè, attorno a $x=6$, arriva a “**nascondere**” completamente l'andamento esponenzialmente decrescente !

Stabilità



Un tale comportamento è un sintomo di **instabilità** nell'algoritmo (13)

Spiegazione: nel nostro caso specifico la (13) assume la forma:

$$y_{n+1} = y_{n-1} - 2h y_n \quad (14)$$

cerchiamo di risolvere l'equazione (14) assumendo che la **soluzione** sia del tipo: $y_n = A r^n$, dove A e r sono costanti;

allora, sostituendo in (14), si arriva ad un'equazione per r (A non è importante perché la relazione ricorsiva è lineare):

$$r^2 + 2hr - 1 = 0 \quad (15)$$

Stabilità

Le **soluzioni** dell'equazione (15) sono (per $h \ll 1$):

$$r_+ = -h + \sqrt{1 + h^2} \approx 1 - h$$

$$r_- = -h - \sqrt{1 + h^2} \approx -(1 + h)$$

la radice **positiva** è leggermente minore di 1 e corrisponde alla soluzione decrescente esponenzialmente (soluzione **vera**), tuttavia la radice **negativa** è leggermente minore di -1 e corrisponde alla soluzione **spuria**:

$$y_n \approx (-1)^n (1 + h)^n$$

la cui **grandezza** (valore assoluto) aumenta con n e che **oscilla** da un punto ad un altro.

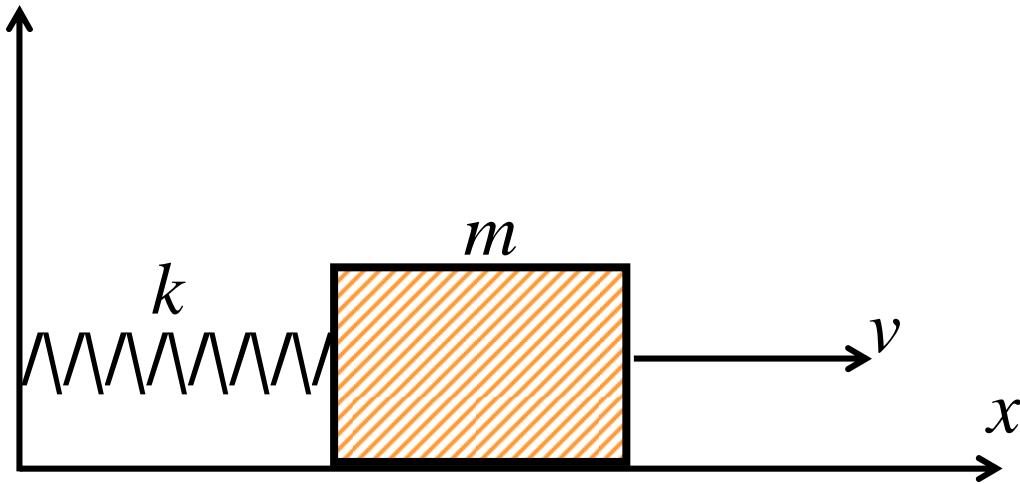
Stabilità

D'altra parte la **soluzione generale** dell'equazione (13) è una **combinazione lineare** delle due soluzioni esponenziali; anche se si possono scegliere i valori iniziali y_0 e y_1 in maniera tale che sia presente, per x piccoli, **solo** la soluzione decrescente, gli effetti degli **arrotondamenti numerici** (dalla (14) si vede che si **sottraggono** due quantità positive per ottenerne una più piccola) introdurranno una piccola **componente** della soluzione **spuria**, la quale crescerà fino a diventare **dominante** !

N.B. in questo caso l'**instabilità** è chiaramente associata alla natura “**a 3 termini**” della relazione ricorsiva (14).

Regola generale: bisogna prestare **attenzione** agli effetti legati a **instabilità** e problemi di arrotondamento ogni volta che si integra una soluzione che **decresce rapidamente** al procedere dell'iterazione.

Esempio Fisico: oscillatore armonico ideale classico



Consideriamo un blocco di massa m , che scivola senza attrito su di una superficie orizzontale ed è legato ad una parete fissa mediante una molla di costante elastica k ; se la molla non è troppo compressa o allungata si può assumere di essere in condizioni **armoniche** e quindi la **forza** (di richiamo) agente sul blocco nella posizione x è:

$$F = -kx$$

Oscillatore armonico ideale classico

allora l'equazione di **Newton** è : $m \frac{d^2 x}{dt^2} = -kx$

che si può anche riscrivere come: $\frac{d^2 x}{dt^2} = -\frac{k}{m} x = -\omega_0^2 x \quad (16)$

avendo introdotto la **frequenza** : $\omega_0 = \sqrt{\frac{k}{m}}$

l'equazione (16) può essere risolta **analiticamente** ed ha la ben nota soluzione: $x(t) = A \cos(\omega_0 t + \phi)$ dove A è l'ampiezza e ϕ una fase costante.

Come già discusso, l'equazione (16), differenziale al **second**'ordine, può essere riscritta come 2 equazioni differenziali al **prim**'ordine (**ODE**) accoppiate, introducendo la **velocità**:

$$v(t) = \frac{dx}{dt}$$

Oscillatore armonico ideale classico

$$\frac{dx}{dt} = v(t)$$

$$\frac{dv(t)}{dt} = -\omega_0^2 x(t)$$

una volta realizzato un algoritmo che risolva questo problema è opportuno effettuare i seguenti **test**:

- confrontare la soluzione **numerica** con quella **analitica**;
- verificare che la soluzione $x(t)$ sia **periodica**: $x(t+T)=x(t)$, col periodo $T=2\pi/\omega_0$ che dipende solo da k/m e non da A o φ ;
- verificare che l'energia meccanica totale si **conservi** (la forza elastica è **conservativa**): supponiamo di scegliere le condizioni iniziali $x(t=0)=1$ m, $v(t=0)=0$ m/s, allora il blocco è “a riposo” a $t=0$ ma possiede l'energia potenziale $1/2kx^2(t=0)=1/2k$ che rimane costante per ogni t :

$$E_{TOT} = \frac{1}{2} kx^2(t) + \frac{1}{2} mv^2(t) = COST. = E_0 = \frac{k}{2}$$

Oscillatore armonico ideale classico

Nel seguito si trova un **programma** che risolve il problema usando il metodo di **Runge-Kutta** al **quarto ordine**;

poiché vogliamo effettuare il test di **periodicità** è conveniente scegliere il tempo finale $t_f \geq 2\pi$ ($t_i=0$), prendendo $k=m=1$;

l'intervallo temporale $[t_i, t_f]$ viene suddiviso in una **griglia** con “step size” $h=(t_f-t_i)/N$ (N =numero di punti della griglia);

sostanzialmente il metodo di Runge-Kutta è usato per ottenere i valori x_{i+1} , v_{i+1} partendo dai valori precedenti x_i, v_i ;

ovviamente la **stabilità** della soluzione numerica deve essere studiata in funzione del numero N di punti della griglia (o equivalentemente lo step size h).

Oscillatore armonico ideale classico

```
/*
*****
*   rk4.c: 4th order Runge-Kutta solution for harmonic oscillator   *
*                                                                 *
*****
*/
#include <iostream>
#include <cmath>
#include <fstream>

#define N 2                      /* number of equations */
#define MIN 0.0                 /* minimum x */
#define MAX 10.0                /* maximum x */

int main() {
    using namespace std;
    double dist, e0, etot, detot, x, y[N];
    int j;

    void runge4(double x, double y[], double step);
    double f(double x, double y[], int i);

    ofstream out;                /* save data in rk4.out */
    out.open("rk4.out");

    cout << " stepsize ?" << endl;          /* read stepsize */
    cin >> dist;

    e0 = 0.5;                    /* initial exact energy */
```

RK4.cpp

Oscillatore armonico ideale classico

```
y[0] = 1.0;          /* initial position */
y[1] = 0.0;          /* initial velocity */

detot = 0.0;

for(x = MIN; x <= MAX ; x += dist)
{
    runge4(x, y, dist);
    etot=0.5*(y[0]*y[0]+y[1]*y[1]);          /* total energy */
    detot=etot-e0;
    out << x+dist <<" "<< y[0] <<" "<< cos(x+dist) <<" "<< detot << endl;
}
cout << "data stored in rk4.out"<< endl;
}
/*-----end of main program-----*/
```

Oscillatore armonico ideale classico

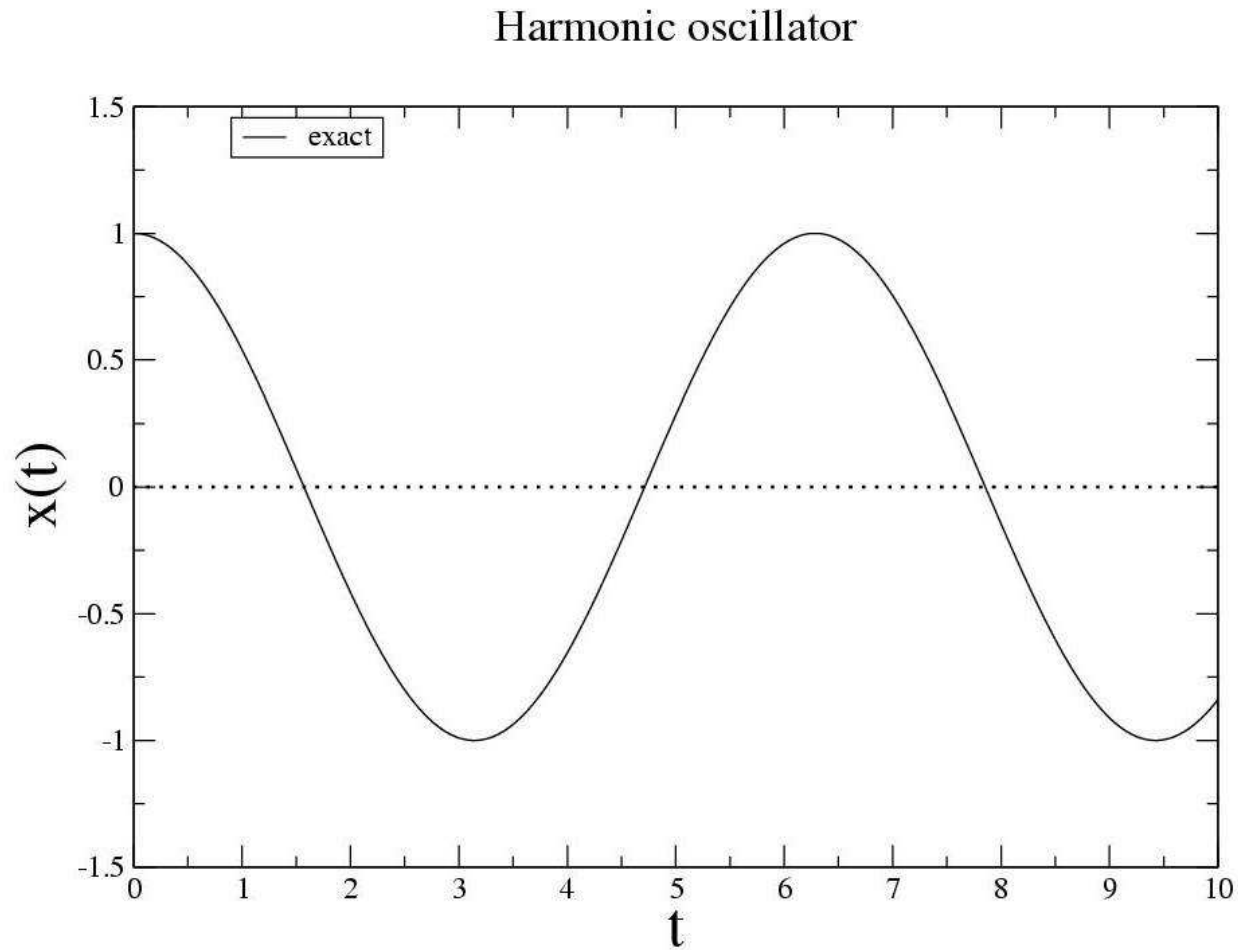
```
/* Runge-Kutta subroutine */
void runge4(double x, double y[], double step)
{
    double h=step/2.0,                /* the midpoint */
           t1[N], t2[N], t3[N],      /* temporary storage */
           k1[N], k2[N], k3[N], k4[N]; /* for Runge-Kutta */
    int i;

    for (i=0; i<N; i++) t1[i] = y[i]+0.5*(k1[i]=step*f(x, y, i));
    for (i=0; i<N; i++) t2[i] = y[i]+0.5*(k2[i]=step*f(x+h, t1, i));
    for (i=0; i<N; i++) t3[i] = y[i]+    (k3[i]=step*f(x+h, t2, i));
    for (i=0; i<N; i++) k4[i] =          step*f(x + step, t3, i);

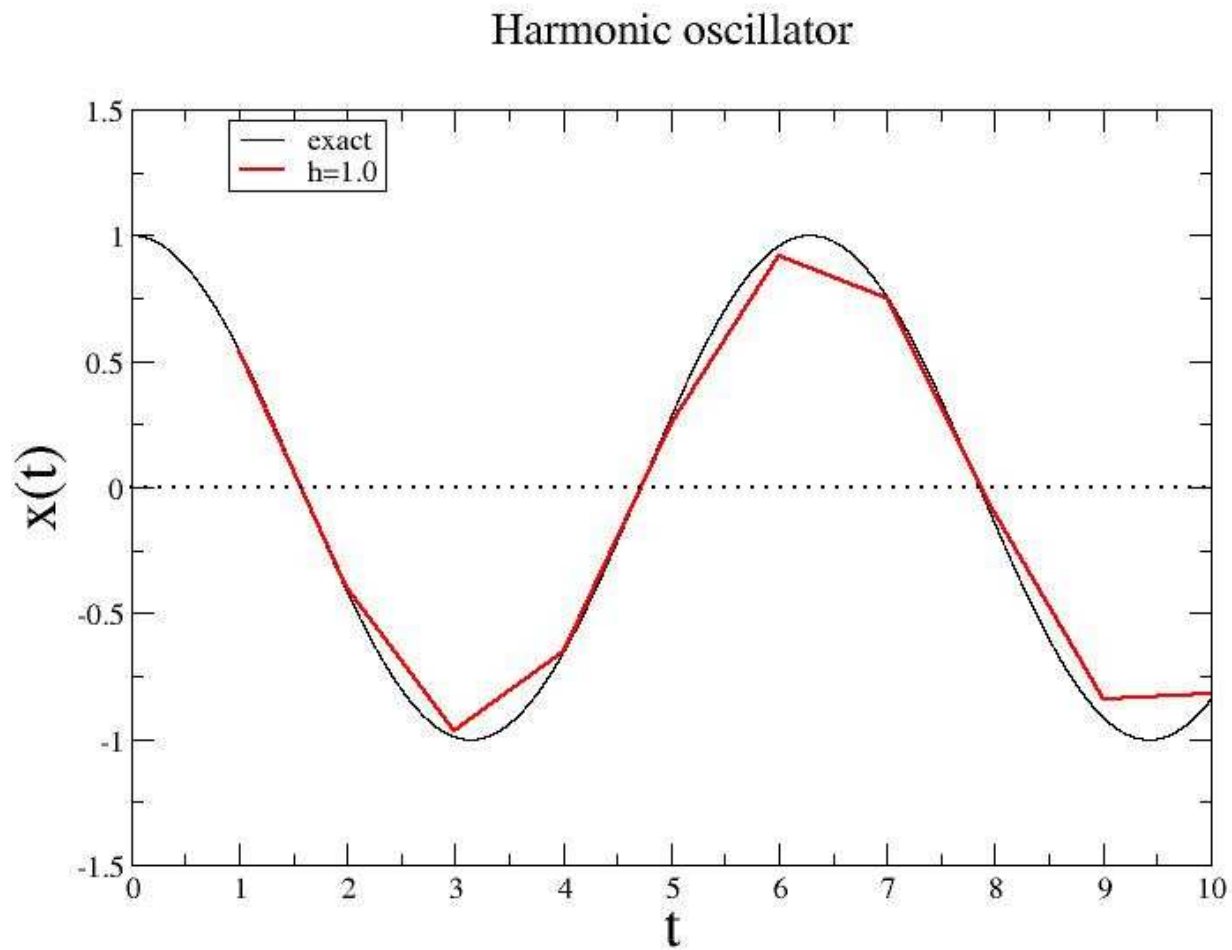
    for (i=0; i<N; i++) y[i] += (k1[i]+2*k2[i]+2*k3[i]+k4[i])/6.0;
}
/*-----*/

/* definition of equations - this is the harmonic oscillator */
double f(double x, double y[], int i)
{
    if (i == 0) return(y[1]);          /* RHS of first equation */
    if (i == 1) return(-y[0]);         /* RHS of second equation */
}
```

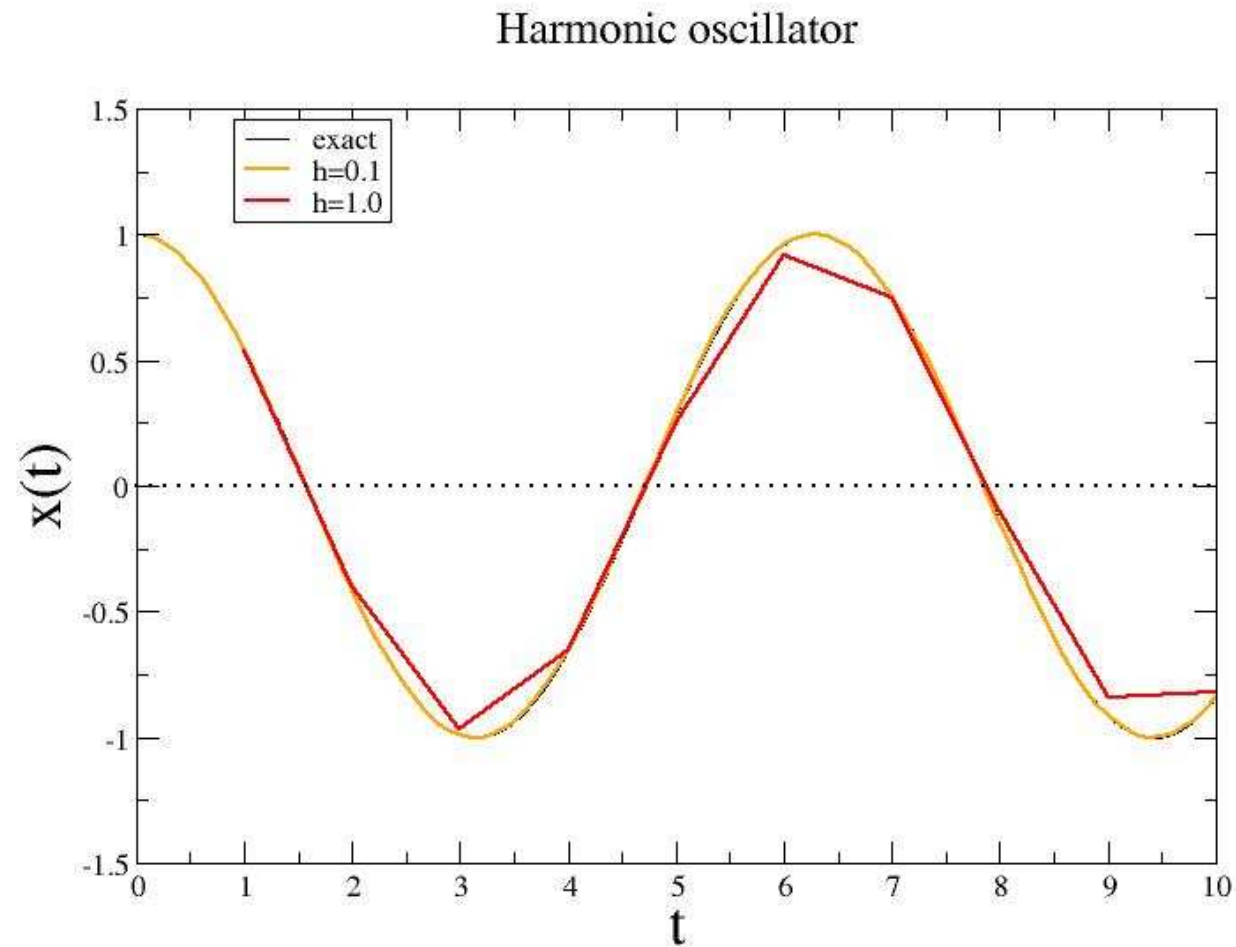
Oscillatore armonico ideale classico



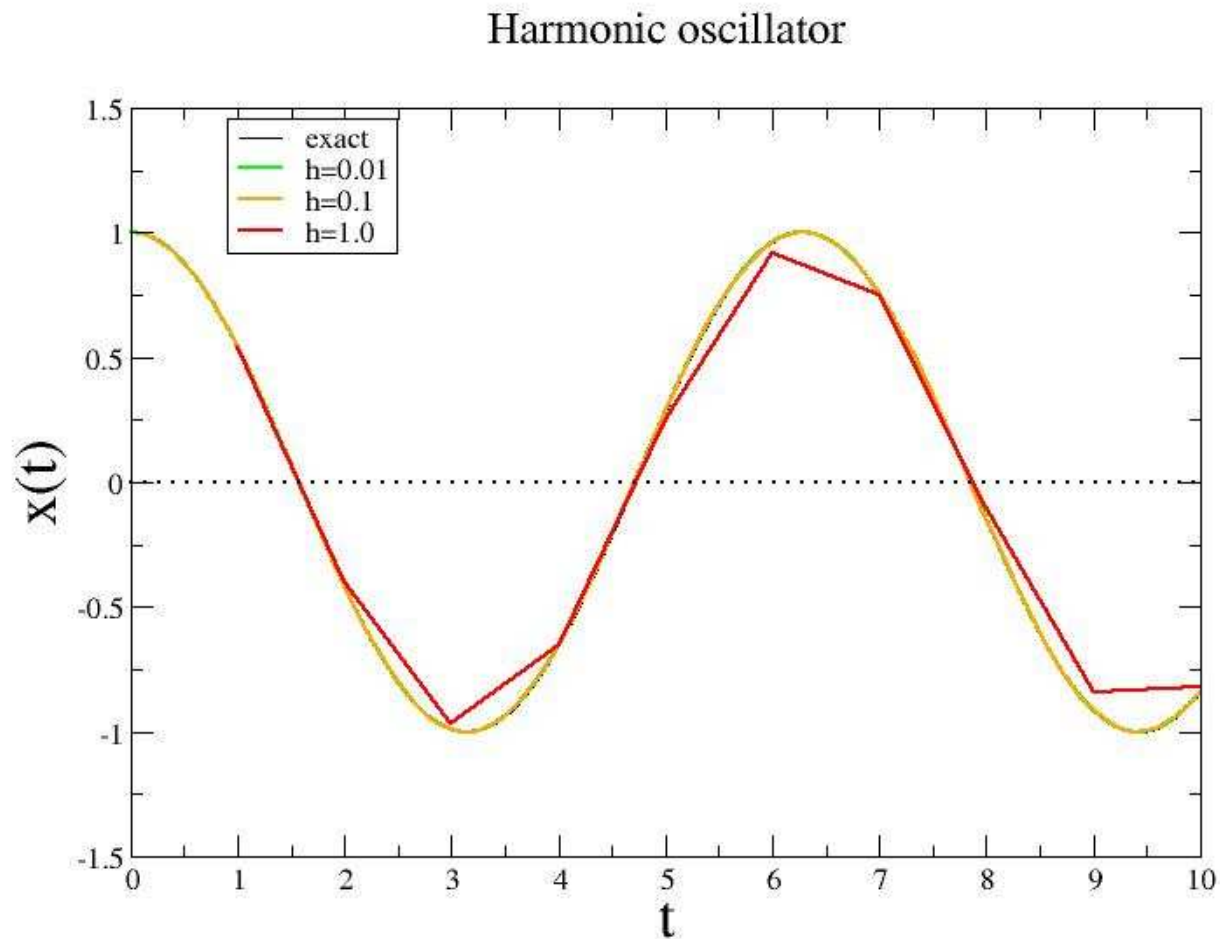
Oscillatore armonico ideale classico



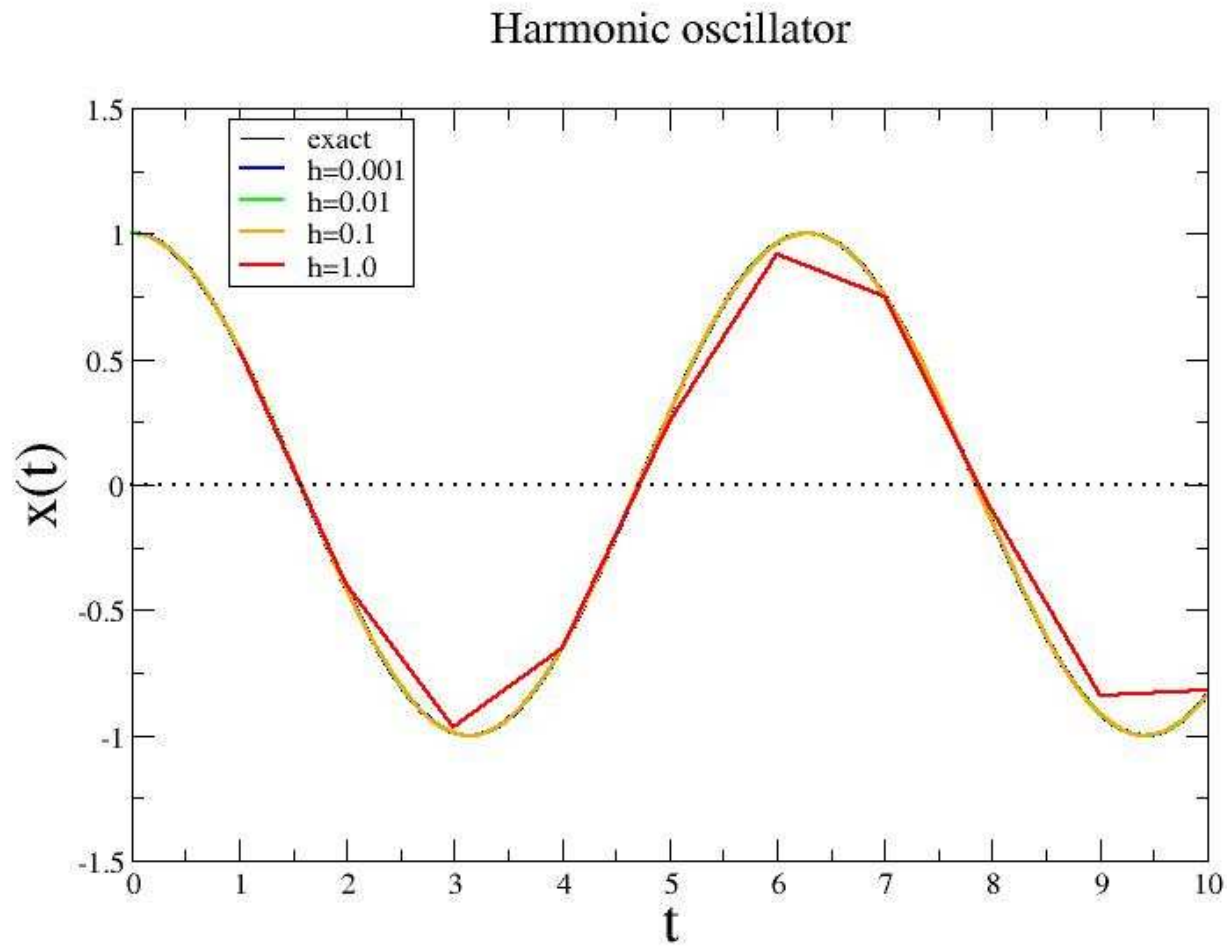
Oscillatore armonico ideale classico



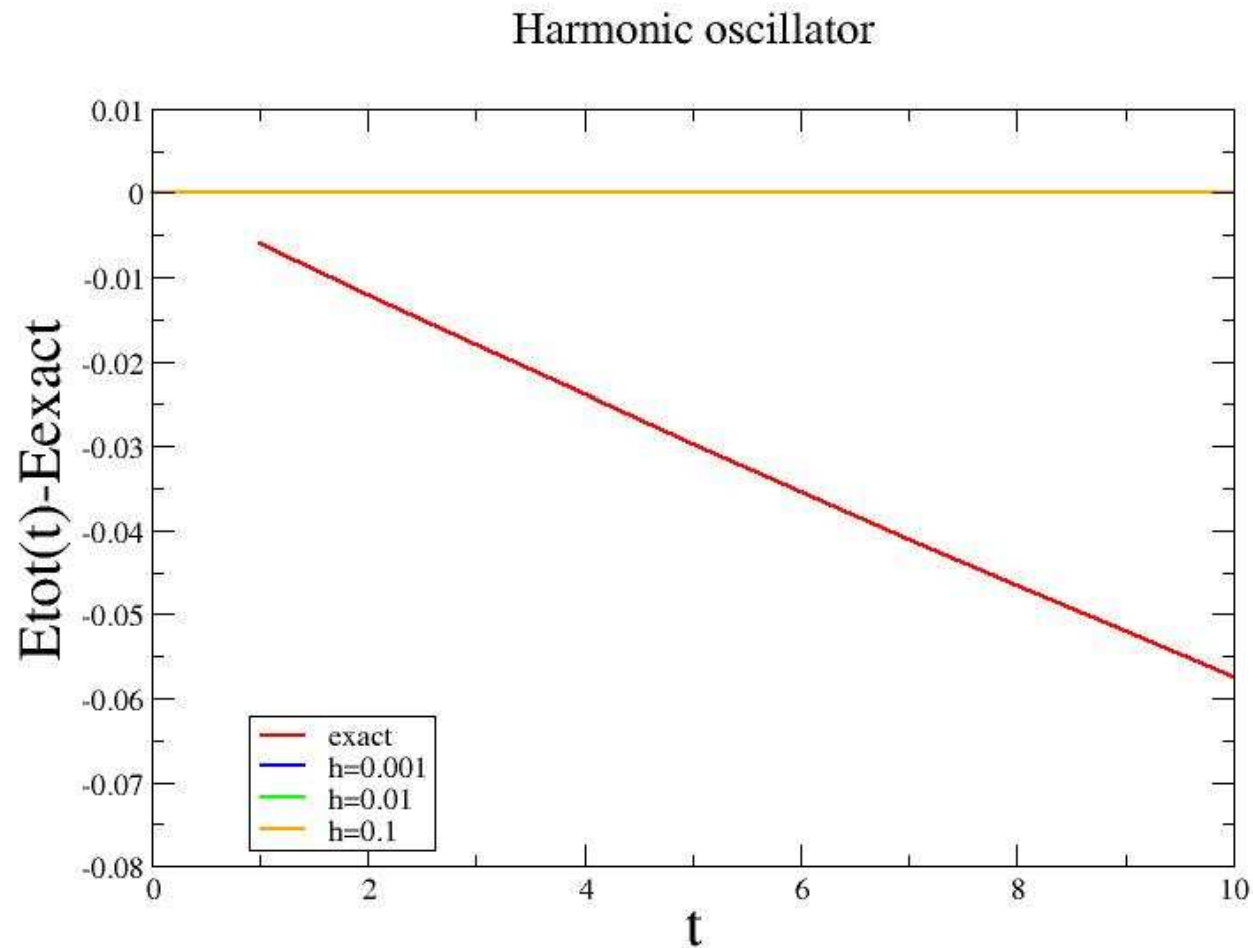
Oscillatore armonico ideale classico



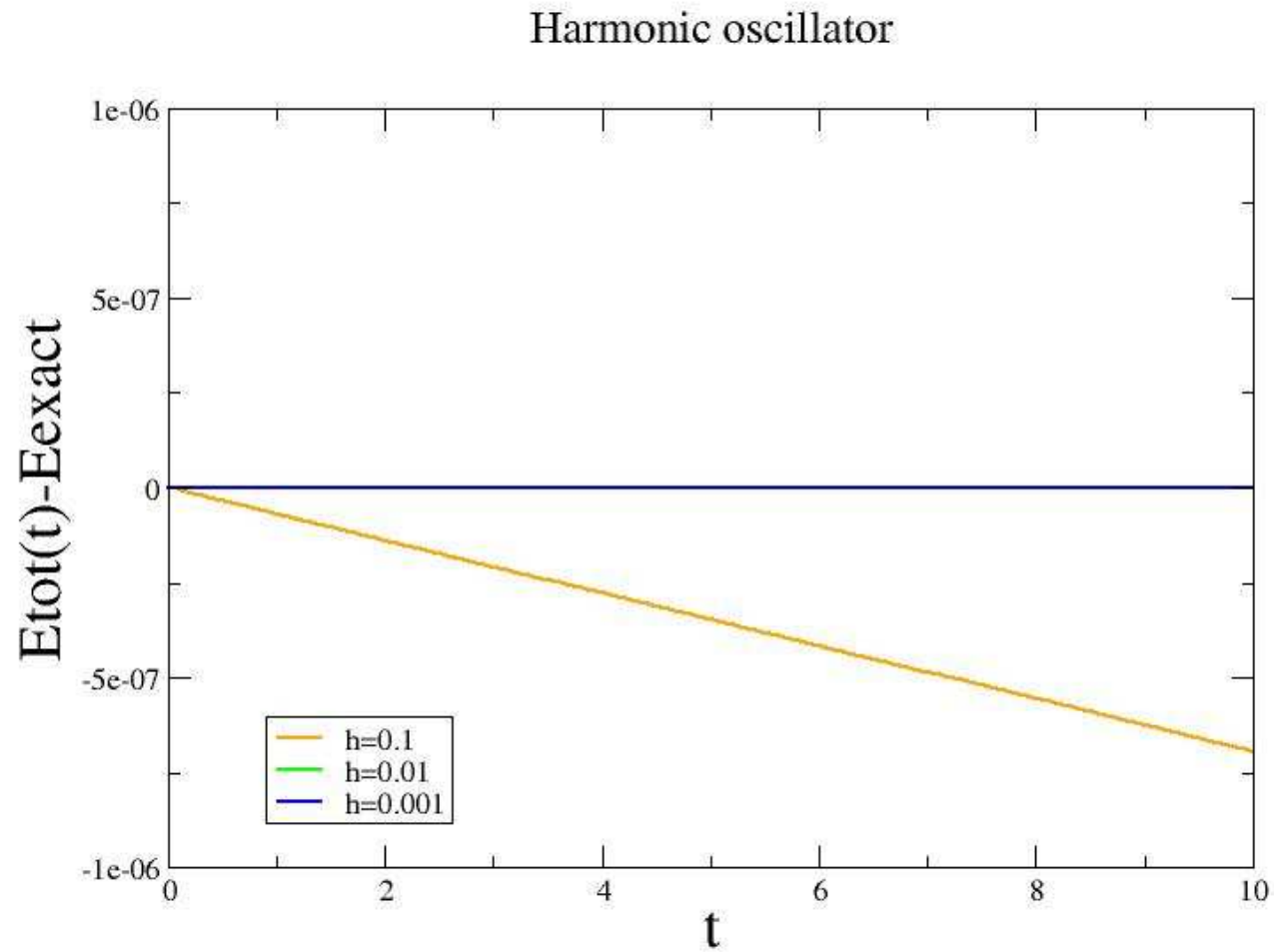
Oscillatore armonico ideale classico



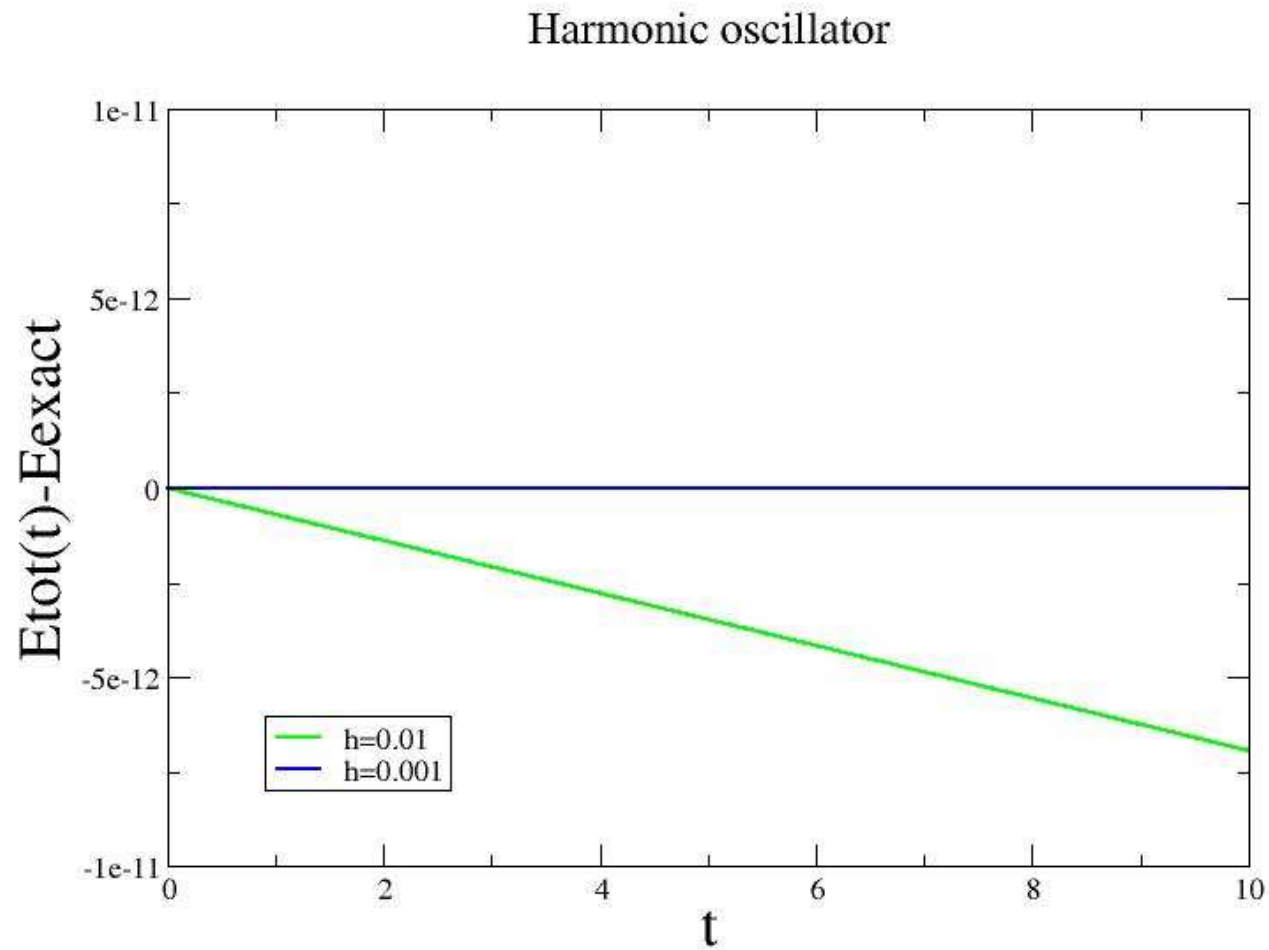
Oscillatore armonico ideale classico



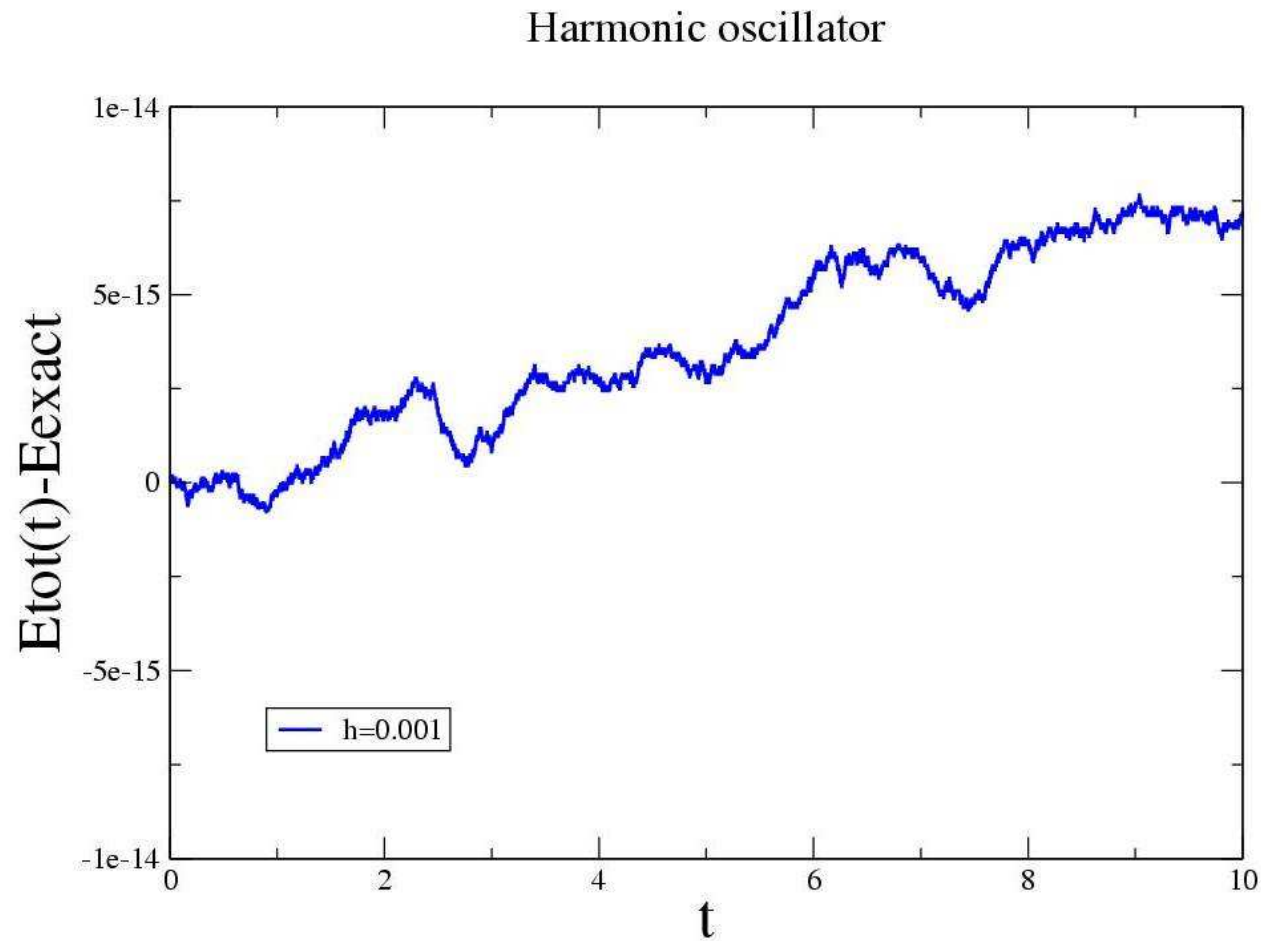
Oscillatore armonico ideale classico



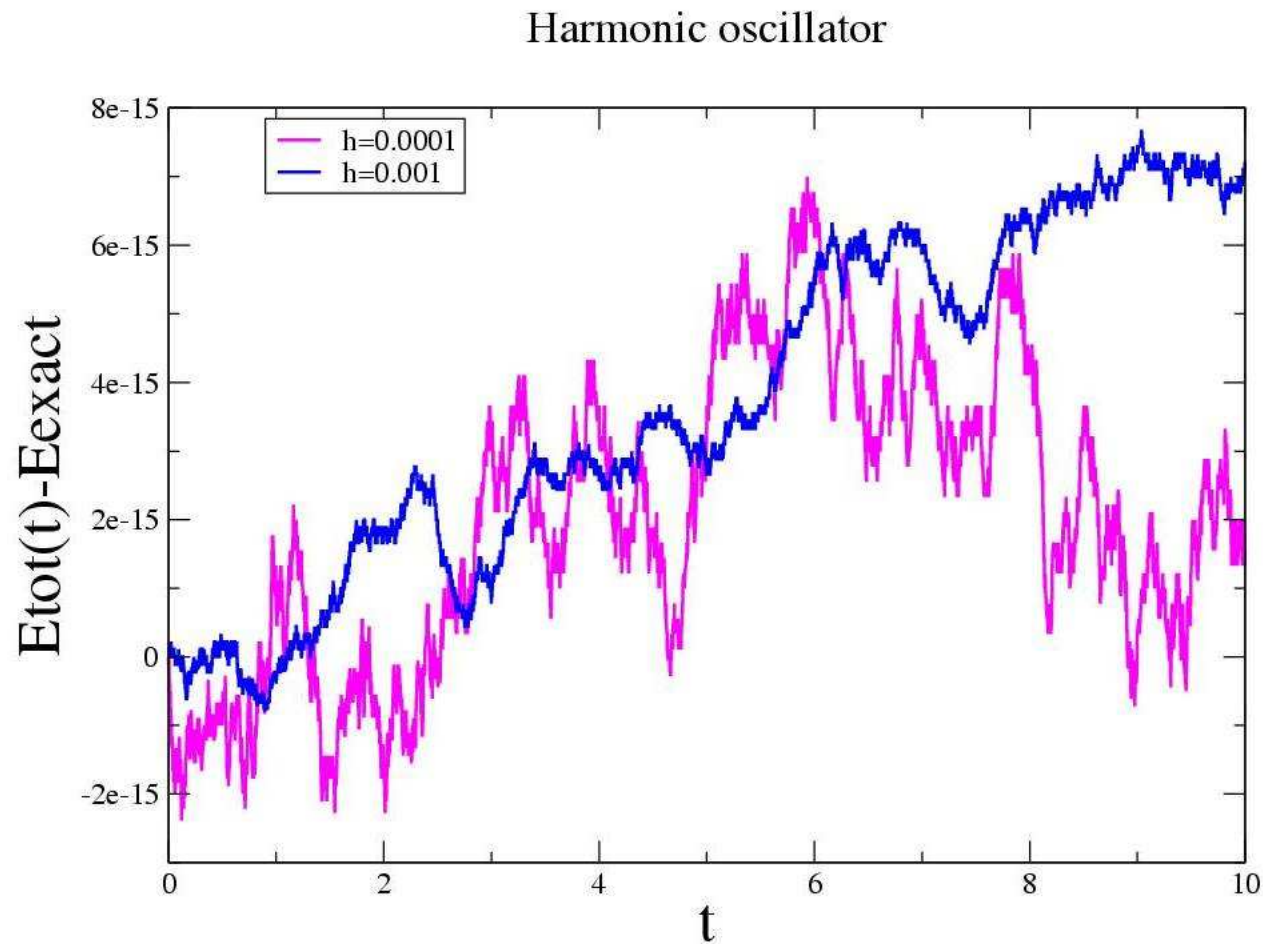
Oscillatore armonico ideale classico



Oscillatore armonico ideale classico



Oscillatore armonico ideale classico

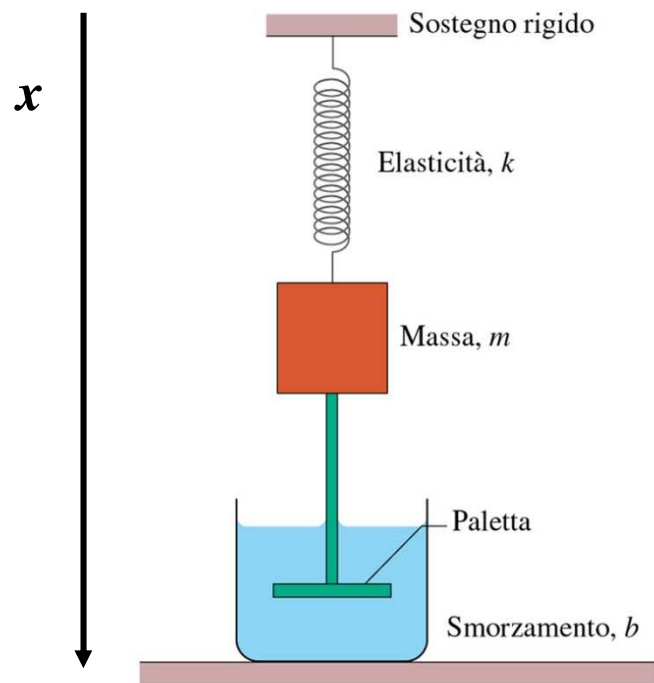


Oscillatore armonico ideale classico

Nelle figure precedenti sono riportati gli andamenti di $x(t)$ e di $\Delta E(t) = E_{calcolata} - E_{esatta}$ per diversi valori di h (e quindi di N); notare che mentre $x(t)$ è riprodotto abbastanza bene già con $h=0.1$, l'energia totale (notare le **scale diverse** !) mostra un “**drift**”, cioè $|\Delta E(t)|$ aumenta monotonamente, tranne che per il valore più piccolo di h (che corrisponde a $N=100000$!) per il quale la soluzione appare **stabile** (ci sono solo piccole **oscillazioni**).

Esercizio 3: oscillatore smorzato

Rispetto alla situazione precedente (oscillatore armonico **ideale**) si aggiunge una **forza (viscosa) di smorzamento**:



$$ma = -kx - bv, \quad v = \frac{dx}{dt}, \quad a = \frac{d^2x}{dt^2} \Rightarrow$$
$$\Rightarrow m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = 0$$

Esercizio 3: oscillatore smorzato

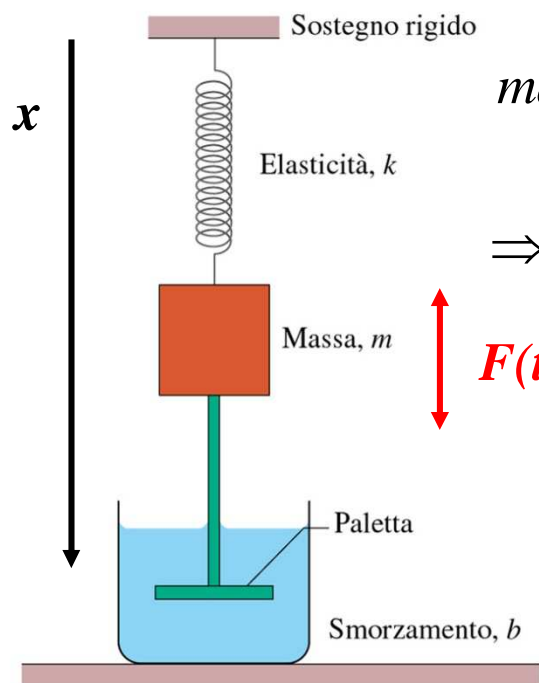
- **modificare** il programma precedente in modo da ripetere i calcoli con due diversi valori della costante di smorzamento: $b=m/5$ e $b=m$ (*forte smorzamento*);
- **confrontare** i risultati con quelli ottenuti con la soluzione analitica (realizzare **grafici** illustrativi e valutare la **non-conservazione** dell'energia meccanica) :

$$x(t) = e^{-\frac{b}{2m}t} \cos(\omega' t), \quad \omega' = \sqrt{1 - \frac{b^2}{4m^2}}$$

- **ripetere** i calcoli usando l'algoritmo di **Eulero** (verificare che è molto **più instabile** !)

Esercizio 3: oscillatore forzato

Rispetto alla situazione precedente (oscillatore armonico smorzato) si aggiunge una **forza oscillante $F(t)$** :



$$ma = -kx - bv + F_0 \sin(\omega t), \quad v = \frac{dx}{dt}, \quad a = \frac{d^2x}{dt^2} \Rightarrow$$

$$\Rightarrow m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = F_0 \sin(\omega t), \quad \omega_0 = \sqrt{\frac{k}{m}}$$

pulsazione propria

Esercizio 3: oscillatore forzato

- **modificare** il programma precedente in modo da ripetere i calcoli con due diversi valori della costante di smorzamento: $b=m/5$ e $b=m$ (*forte smorzamento*) e $F_0 = 0.5$ (**N.B. aumentare il tempo complessivo di simulazione a $T=100$!**) per **diversi** valori di ω ;
- **verificare** che, dopo un *transiente*, l'oscillatore oscilla con la frequenza ω della forza esterna;
- **verificare** che, quando il sistema raggiunge lo stato *stazionario*, l'energia **media** rimane **costante** (la potenza media fornita in un ciclo dalla forza esterna viene tutta dissipata dall'attrito);
- **verificare** che, la **potenza media** (fornita dalla forza esterna, $P=Fv$) raggiunge il suo **massimo** (**risonanza !**) per $\omega=\omega_0$, mentre invece l'**ampiezza** dell'oscillazione è **massima** per :

$$\omega_m = \sqrt{\omega_0^2 - \frac{b^2}{2m^2}} < \omega_0$$