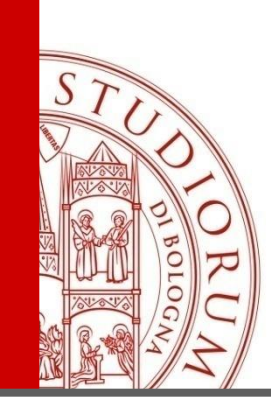


# Equazioni Differenziali Ordinarie – IVP

## Esempi

# Soluzione simbolica di ODE con MATLAB

- A volte è possibile determinare la soluzione in forma analitica di un'equazione differenziale.
- Tale funzione è detta soluzione simbolica e si ottiene con la function dsolve del **Symbolic Math Toolbox.**
- Potenzialità del **Symbolic Math Toolbox:**
  - algebra simbolica;
  - risoluzione di equazioni algebriche e trascendenti;
  - calcolo di integrali, derivate, serie, ecc…;
  - metodi simbolici per risolvere ODE.

# La funzione dsolve

| | |
|---|---|
| **dsolve('eqn')** | Restituisce la soluzione dell'equazione differenziale specificata in 'eqn' |
| **dsolve('eq1', 'eq2',...)** | restituisce la soluzione del sistema di equazioni differenziali eq1, eq2, … |
| **dsolve('eq1', 'cond1', 'cond2',...)** | Risolve l'equazione differenziale eq1 con condizioni iniziali cond1, cond2,… |
| **dsolve('eq1', 'eq2',..., 'cond1', 'cond2',...)** | risolve il sistema di equazioni differenziali con condizioni iniziali cond1, cond2,… |

# Esempi

**1)** Risolvere l'equazione: $\frac{dy}{dt} + 2y = 12$

```
>> dsolve('Dy+2*y=12')
ans =
6+exp(-2*t)*C1
```

**N.B.** Per indicare la derivata prima si utilizza la lettera D, per la derivata seconda D2, e così via.

**2)** Risolvere l'equazione: $\frac{d^2y}{dt^2} = c^2 y$

```
>> dsolve('D2y=c^2*y')
ans =
C1*sinh(c*t)+C2*cosh(c*t)
```

La soluzione è trovata in funzione di costanti arbitrarie C1, C2

# Esempi

**3)** Risolvere il sistema di equazioni:

$$\frac{dx}{dt} = 3x + 4y$$
$$\frac{dy}{dt} = -4x + 3y$$

```
>> [x,y]=dsolve('Dx=3*x+4*y','Dy=-4*x+3*y')
x = exp(3*t)*(cos(4*t)*C1+sin(4*t)*C2)
y = -exp(3*t)*(sin(4*t)*C1-cos(4*t)*C2)
```

**4)** Risolvere l'equazione:

$$\frac{dy}{dt} = \sin(bt)$$

$$y(0) = 0$$

```
>> dsolve('Dy=sin(b*t)','y(0)=0')
ans =
(-cos(b*t)+1)/b
```

# Esempi

**5)** Risolvere il sistema di equazioni:

$$\frac{dx}{dt} = 3x + 4y$$
$$\frac{dy}{dt} = -4x + 3y$$

$$x(1) = 0$$
$$y(0) = 1$$

**>> [x,y]=dsolve('Dx=3*x+4*y','Dy=-4*x+3*y','x(1)=0','y(0)=1')**

x = exp(3*t)*(-cos(4*t)*sin(4)/cos(4)+sin(4*t))

y = -exp(3*t)*(-sin(4*t)*sin(4)/cos(4)-cos(4*t))

# Esempi

**6)** MATLAB risolve anche molte equazioni differenziali del primo ordine non lineari:

$$\frac{dy}{dt} = 4 - y^2 \qquad y(0) = 1$$

```
>> dsolve('Dy=4-y^2','y(0)=1')
ans =
(2*exp(4*t+log(-3))+2)/(-1+exp(4*t+log(-3)))

>> simple(ans)
ans =
(-6*exp(4*t)+2)/(-1-3*exp(4*t))
```

# Esempi

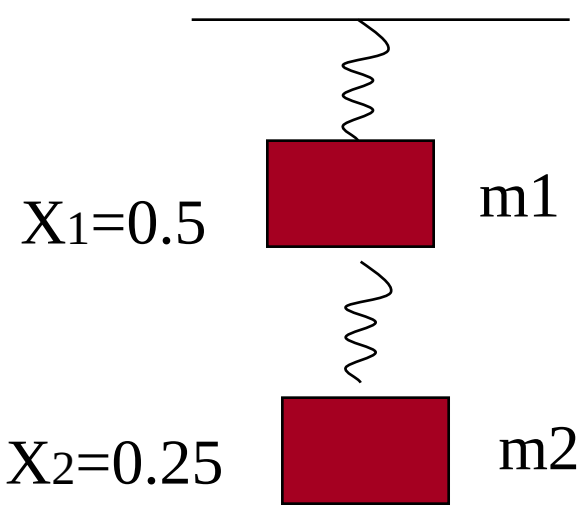
**7)** Risolvere l'equazione: $\frac{dy}{dt} = x + y \quad y(0) = 2$

```
>> sol=dsolve('Dy=x+y‘,’y(0)=2’,’x’)
sol =
-x-1+3*exp(x)
```

Fare il grafico della soluzione calcolata

```
>> ezplot(sol,[-1,2])
```

# Esempio: sistema massa-molla




Lo spostamento verticale di due masse sospese in serie da molle di costante elastica k1, k2, è dato dalla legge di Hooke come un sistema di due ODE del secondo ordine:

$$m_1 \frac{d^2 x_1}{dt^2} - k_2(x_2 - x_1) + k_1 x_1 = 0$$

$$m_2 \frac{d^2 x_2}{dt^2} + k_2(x_2 - x_1) = 0$$

# Esempio: sistema massa-molla

- Sostituzione di variabili per avere un sistema di 4 ODE del primo ordine:

$$u_1 = x_1$$

$$u_3 = x_2$$

$$\begin{cases} u_1' = u_2 \\ u_2' = +\dfrac{k_2}{m_1}(u_3 - u_1) - \dfrac{k_1}{m_1}u_1 \\ u_3' = u_4 \\ u_4' = -\dfrac{k_2}{m_2}(u_3 - u_1) \end{cases}$$

Condizioni iniziali: $u_1(0) = a_1 \quad u_2(0) = a_2 \quad u_3(0) = a_3 \quad u_4(0) = a_4$

# Esempio: sistema massa-molla

- Passo 1: Scrivere la function che definisce il sistema:

$$\begin{cases} u_1' = u_2 \\ u_2' = +\dfrac{k_2}{m_1}(u_3 - u_1) - \dfrac{k_1}{m_1}u_1 \\ u_3' = u_4 \\ u_4' = -\dfrac{k_2}{m_2}(u_3 - u_1) \end{cases}$$

```
function dydt=massamolla(t,y)
global k1 m1 k2 m2
dydt=[y(2);
    (-k1*y(1)+k2*(y(3)-y(1)))/m1;
    y(4);
    -k2*(y(3)-y(1))/m2];
```

# script **esempioODE.m**

## Passo 2: Definizione dei dati del problema:

```
% Dati del problema
a=0; b=5; tspan=[a b];          % intervallo di integrazione
y0=[0.5 0 0.25 0];              % condizioni iniziali
global k1 m1 k2 m2
k1=100; m1=10; k2=120; m2=2; % misure in Kg, metri, Newton
```

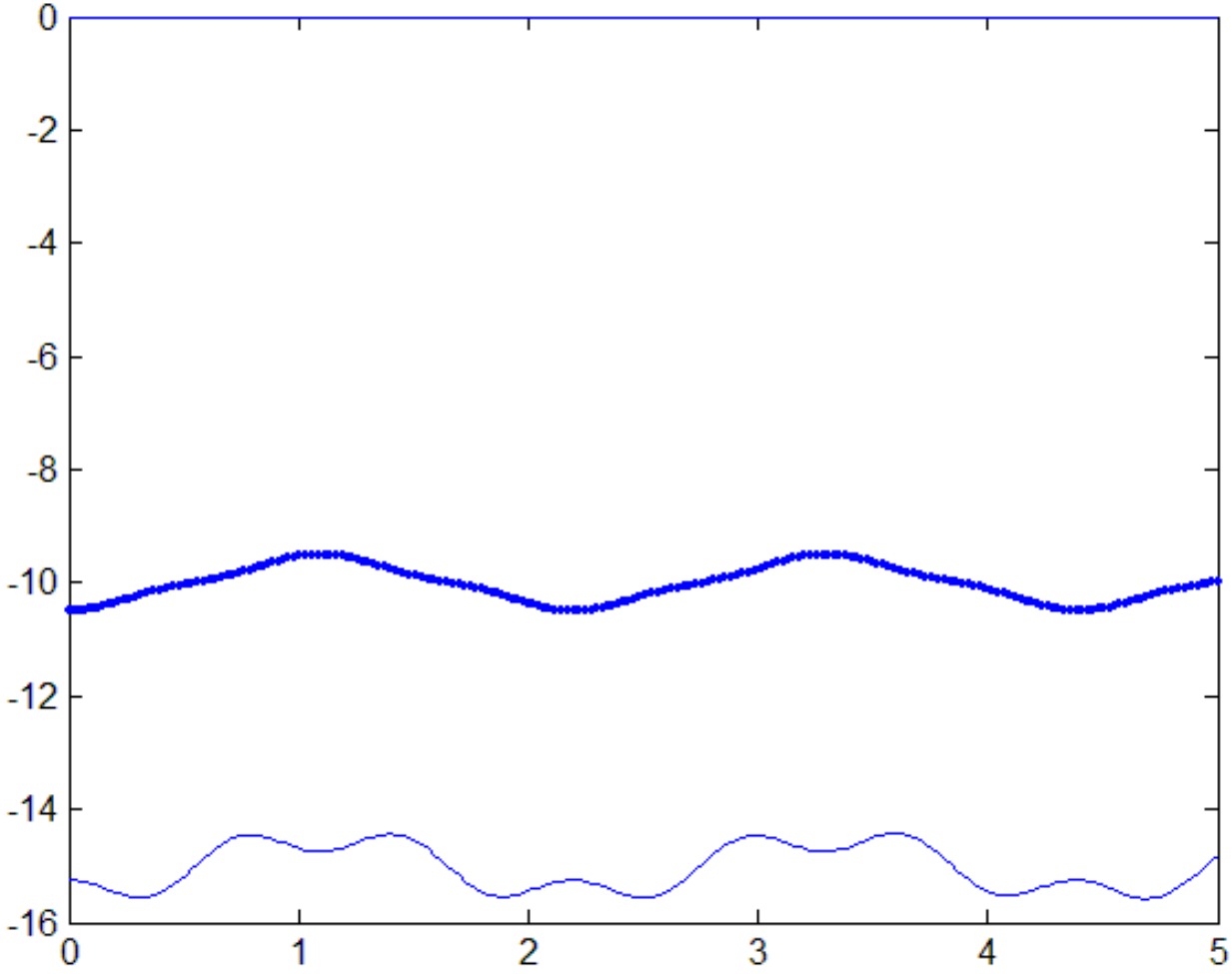
## Passo 3: Richiamo solutore del sistema di ODE:

```
% Solutore del sistema di ode:
[t,y]=ode45('massamolla',tspan,y0);
```

## Passo 4: Visualizzazione della soluzione:

```
% Visualizzazione soluzione ottenuta:
% la direzione è verso il basso, quindi il grafico è di -y
% la posizione reale dipende anche dalla lunghezza delle molle
% quindi consideriamo r1 ed r2:
r1=10; r2=15;
plot(t,-r1-y(:,1),'.-');
hold on
plot(t,-r2-y(:,3),'-');
plot([a b],[0 0]);
```

# Esempio: sistema massa - molla



# Solutori disponibili

| *solver* | *metodo implementato* | *osservazioni* |
|---|---|---|
| **ode45** | Runge-Kutta 4-5 esplicito ad un passo | da usare come primo tentativo |
| **ode23** | Runge-Kutta 2-3 esplicito ad un passo | tipicamente più efficiente di **ode45** |
| **ode113** | Adams-Bashforth-Moulton | consigliabile se la valutazione di **F** è costosa |
| **ode15s** | multistep implicito di ordine variabile | da usare come primo tentativo |
| **ode23s** | solutore ad un passo | tipicamente più efficiente di **ode15s** |
| **ode15s** | multistep implicito di ordine variabile | da usare come primo tentativo |
| **ode23s** | solutore ad un passo | tipicamente più efficiente di **ode15s** |
| **ode23t** | schema basato sul metodo del trapezio | non dissipativo |
| **ode23tb** | trapezio + differenze all'indietro | equivalente a Runge-Kutta implicito |

Tabella : Descrizione e caratteristiche degli **ode solver**. I solver con suffisso "s" sono specifici per problemi *stiff*

# Event in MATLAB

- Soluzione in $[t_0, t_{finale}]$
- Quando $t_{finale}$ non è noto

Trovare una funzione y(t) ed un valore finale t* tali che

$$y' = f(t, y)$$

$$y(t_0) = y_0,$$

$$g(t^*, y(t^*)) = 0$$

# Event in MATLAB

ESEMPIO:

corpo in caduta libera sotto la forza di gravità. Quando colpisce il terreno?

$$y'' = -1 + (y')^2$$

$$y(0) = 1, \quad y'(0) = 0,$$

Per quale t y(t)=0?

# Event in MATLAB

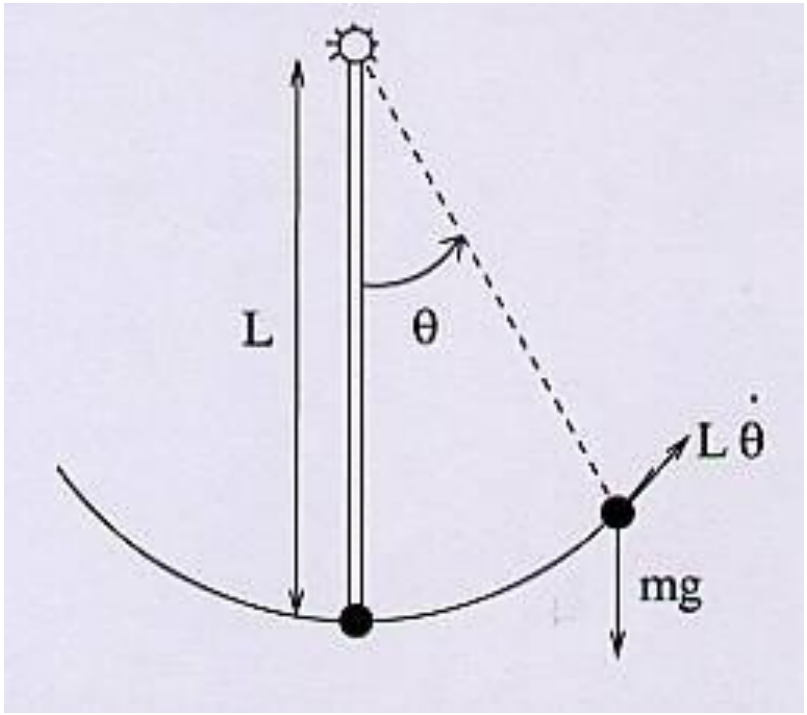
```
function ydot = f(t,y)
ydot = [y(2); -1+y(2).^2];

function [gs,isterm,direct] = gstop(t,y)
gs = y(1);
isterm = 1;
direct = [];
```

% Valore che deve annullarsi
% 1 -> L'ODE termina quando gs=0

```
opts = odeset('events',@gstop);
[t,y,tfinal] = ode23(@f,[0 Inf],[1; 0],opts);
tfinal
```

# Equazione di moto del pendolo senza attrito




Per semplicità $\omega^2=1$

Pendolo di **massa m**, **lunghezza L**

Evoluzione temporale dell'angolo $\theta$ formato dal pendolo con la verticale passante per la cerniera a cui esso è vincolato

$$\begin{cases} \ddot{\theta} + \omega^2 \sin\theta = 0 \qquad \text{con} \qquad \omega^2 = \dfrac{g}{L} \\ \theta(0) = \theta_0 \\ \dot{\theta}(0) = \dot{\theta}_0 \end{cases}$$

# Equazione di moto del pendolo senza attrito

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -\sin(y_1) \end{cases}$$

Condizioni iniziali

$$\begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} = \dot{y}_1 \end{cases}$$

Il corrispondente **ode file** ***pendolo.m*** è:

```
function yp=pendolo(t,y)
% pendolo senza attrito
yp=[y(2); -sin(y(1))];
```

Risolviamo il problema con **ode45** nell'intervallo [0  100], con condizioni iniziali

$$\theta_0 = \pi - \delta, \quad \dot{\theta}_0 = 0.$$

Per δ=0.01 (il pendolo si trova quindi inizialmente in alto in posizione quasi verticale con velocità nulla)

# Equazione di moto del pendolo senza attrito

```
>> [t,y]=ode45('pendolo',[0  100],[pi-0.01  0]);
>>x=plot(t,y(:,1),y(:,2));
```

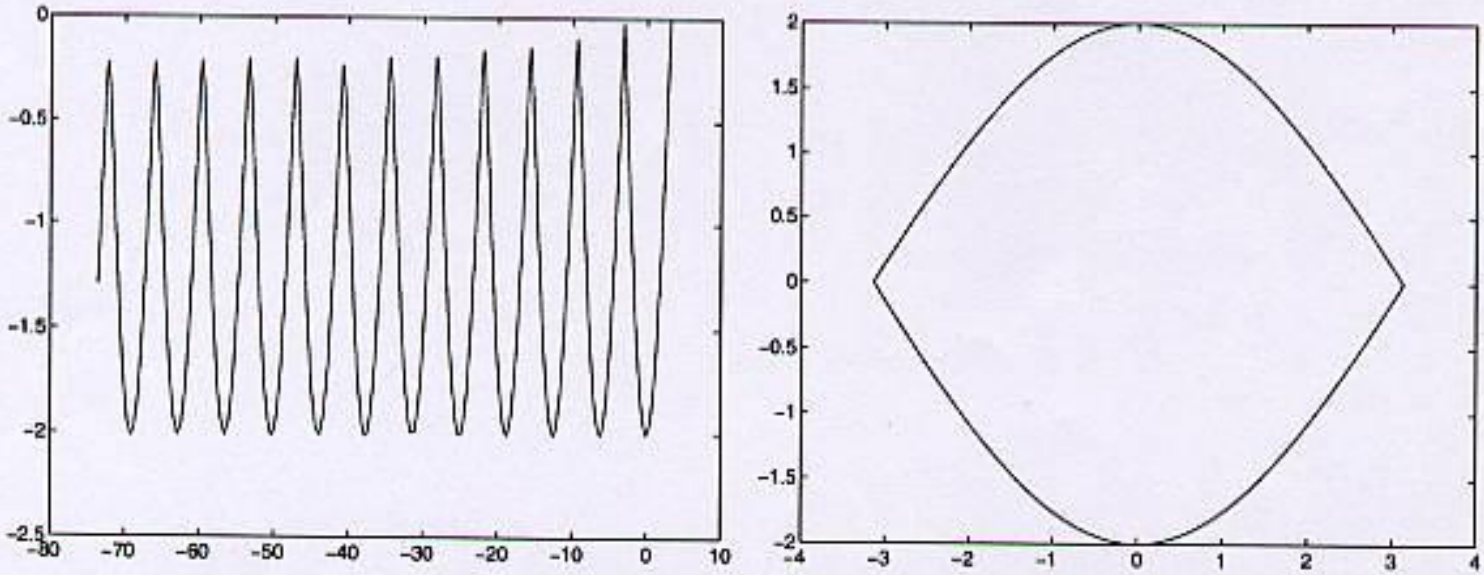


Figura 35: Soluzione nel piano delle fasi per il pendolo con velocità angolare iniziale al di sotto della velocità limite: a sinistra è rappresentata la soluzione fisicamente non accettabile ottenuta con i parametri di default di ode45 mentre a destra è rappresentata la soluzione corretta ottenuta variando i parametri `RelTol` e `AbsTol`

# Equazione di moto del pendolo senza attrito

- Infatti per $\dot{\theta}_0 = 0$ ci troviamo sicuramente al di sotto della velocità limite del pendolo $\dot{\theta}_{\lim} = 2$, oltre la quale il pendolo compie delle rotazioni complete.
- Quindi le traiettorie nel piano delle fasi devono essere linee chiuse, ossia il pendolo non riesce a compiere un giro completo ma continua ad oscillare fra l'angolo

$$\theta_0 = \pi - \delta \quad \text{e} \quad \theta_f = -(\pi - \delta)$$

- Come rimediare a questa situazione? Dobbiamo intervenire sulle impostazioni (dette ***proprietà***) che determinano la precisione con cui opera il solutore.
- A questo scopo utilizziamo la ***struttura options***: essa viene costruita dal comando ***odeset*** e viene accettata in ingresso da tutti i solutori.

# Equazione di moto del pendolo senza attrito

La sintassi per assegnare alla ***proprietà NomeProp-i*** il ***valore Val-i*** è:

**options=odeset('NomeProp1',Val1,'NomePro2',Val2,…)**

Proviamo a modificare tramite le proprietà ***RelTol*** e ***AbsTol*** la tolleranza relativa all'errore locale commesso sulla componente i-esima della soluzione ad ogni passo di integrazione. Matlab cerca di soddisfare il seguente criterio sull'errore commesso nell'approssimazione della i-esima componente della soluzione ***y(i):***

$$|e(i)| \leq \max(RelTol * abs(y(i)), AbsTol)$$

```
>>options=odeset('RelTol',1e-6,'AbsTol',1e-7)
>>[t,y]=ode45('pendolo',[0 100],[pi-0.01 0],options);
>>plot(y(:,1),y(:,2));
```

La soluzione ottenuta in questo modo mostra un comportamento corretto.

# Equazione di moto del pendolo senza attrito

**Help odeset**

Comando ***odeset***

{ } valori di default

```
AbsTol: [ positive scalar or vector {1e-6} ]
           BDF: [ on | {off} ]
        Events: [ on | {off} ]
   InitialStep: [ positive scalar ]
      Jacobian: [ on | {off} ]
     JConstant: [ on | {off} ]
      JPattern: [ on | {off} ]
          Mass: [ {none} | M | M(t) | M(t,y) ]
  MassSingular: [ yes | no | {maybe} ]
      MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
       MaxStep: [ positive scalar ]
    NormControl: [ on | {off} ]
     OutputFcn: [ string ]
     OutputSel: [ vector of integers ]
        Refine: [ positive integer ]
        RelTol: [ positive scalar {1e-3} ]
         Stats: [ on | {off} ]
    Vectorized: [ on | {off} ]
```

**Prima delle modifiche**

```
AbsTol: [ positive scalar or vector {1e-7} ]
           BDF: [ on | {off} ]
        Events: [ on | {off} ]
   InitialStep: [ positive scalar ]
      Jacobian: [ on | {off} ]
     JConstant: [ on | {off} ]
      JPattern: [ on | {off} ]
          Mass: [ {none} | M | M(t) | M(t,y) ]
  MassSingular: [ yes | no | {maybe} ]
      MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
       MaxStep: [ positive scalar ]
    NormControl: [ on | {off} ]
     OutputFcn: [ string ]
     OutputSel: [ vector of integers ]
        Refine: [ positive integer ]
        RelTol: [ positive scalar {1e-6} ]
         Stats: [ on | {off} ]
    Vectorized: [ on | {off} ]
```

**Dopo le modifiche**

# Equazione di moto del pendolo senza attrito

Per uno studio completo del comportamento della soluzione di un'equazione differenziale ordinaria è utile rappresentare le traiettorie percorse dalle componenti della soluzione ***nel piano delle fasi*** ***al variare delle condizioni iniziali***.

Consideriamo ancora il problema del pendolo senza attrito e tracciamo le orbite corrispondenti all'insieme delle condizioni iniziali:

$$\begin{cases} \theta_i(0) = 0 \\ \dot{\theta}_i(0) = -3 + i\Delta\dot{\theta} \end{cases}$$

dove, per i=0,1,…,12, l'incremento $\Delta\dot{\theta} = 1/2$ fa sì che la velocità angolare iniziale descriva l'intervallo di valori

$$-3 \le \dot{\theta}_0 \le 3$$

in modo da comprendere tutta la gamma di situazioni possibili, ovvero:

$$\begin{cases} |\dot{\theta}_0| > \dot{\theta}_{\text{lim}} \\ |\dot{\theta}_0| = \dot{\theta}_{\text{lim}} = 2\omega = 2 \\ |\dot{\theta}_0| < \dot{\theta}_{\text{lim}} \end{cases}$$

# Equazione di moto del pendolo senza attrito

A tale scopo, trovata la soluzione corrispondente ad una certa condizione iniziale, eseguiamo un ***plot*** con il comando ***hold*** per sovrapporre i vari grafici delle traiettorie.

```
clear all
close all
options=odeset('relTol',1e-6,'AbsTol',1e-7);
hold on
N=12;
deltatp=6/N;
for i=0:N
    tetap=-3+i*deltatp;
    [t,y]=ode45('pendolo',[0 25],[0 tetap],options);
    plot(y(:,1),y(:,2),'b', y(:,1),-y(:,2),'b');
end
axis('equal')
grid
```

# Equazione di moto del pendolo senza attrito

**Comando zoom**

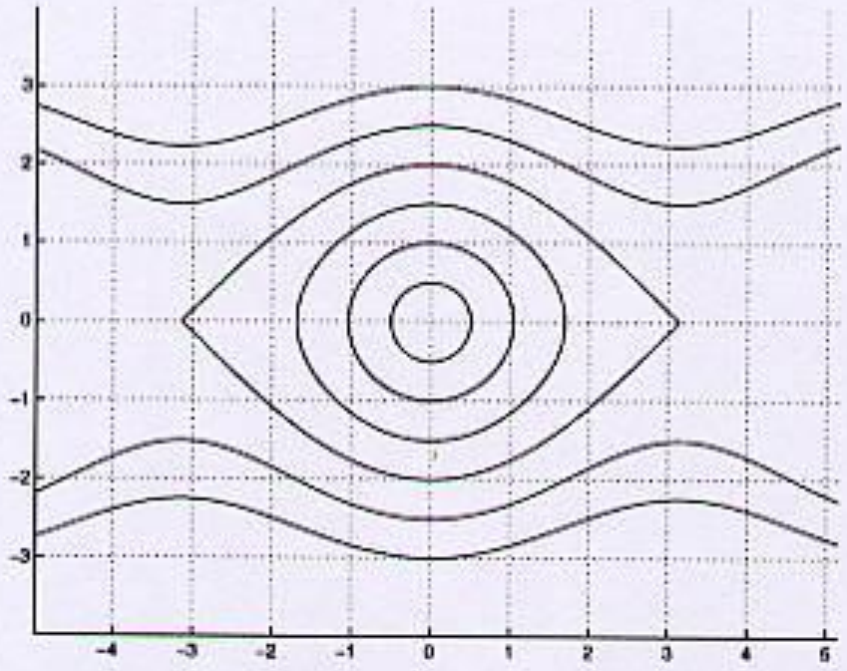



Figura 36: Studio parametrico delle orbite per il problema del pendolo. Le orbite chiuse interne corrispodono a $|\dot{\theta}_0| < \dot{\theta}_{lim}$, l'orbita chiusa più esterna corrisponde a $|\dot{\theta}_0| = \dot{\theta}_{lim} = 2$, mentre le curve che intersecano l'asse delle ordinate per valori in modulo maggiori di 2 corrispondono a $|\dot{\theta}_0| > \dot{\theta}_{lim}$

# Equazione di Van der Pol

$$\begin{cases} y'' - \mu\left(1 - y^2\right)y' + y = 0 \\ y(0) = 2 \qquad y'(0) = 0 \end{cases}$$

$$\mu \in \Re^+$$

smorzamento del sistema

$$\begin{cases} y_1' = y_2 \\ y_2' = \mu\left(1 - y_1^2\right)y_2 - y_1 \end{cases}$$

$$\begin{cases} y_1(0) = 2 \\ y_2(0) = 0 \end{cases}$$

# Equazione di Van der Pol

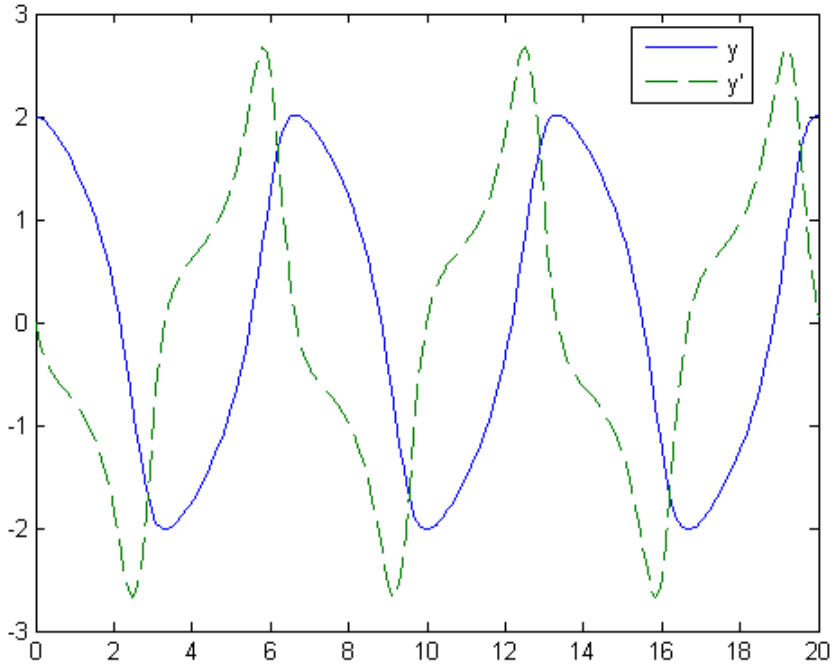
Fissiamo **μ=1** (smorzamento debole) e creiamo l' **ode file** ***vdpol.m***

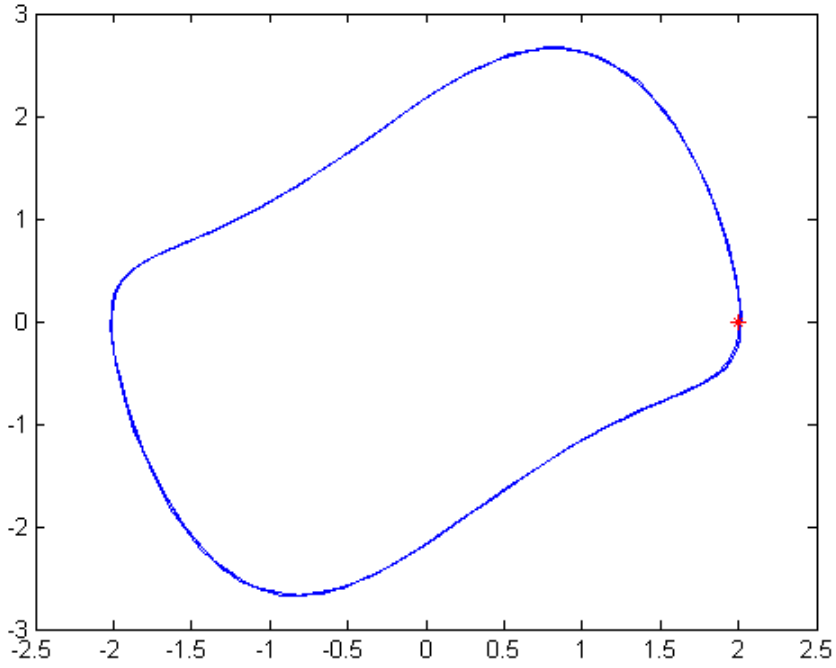
```
function yp=vdpol(t,y)
% equazione di Van der Pol
mu=1;
yp=[y(2);mu*(1-y(1)^2)*y(2)-y(1)];
```

Lanciamo il solver ***ode45*** nell'intervallo [0 20] e plottiamo i risultati componente per componente...

```
>> [t,y]=ode45(@vdpol,[0 20],[2 0]);
>>plot(t,y(:,1),'-',t,y(:,2),'—');
```

# Equazione di Van der Pol




plot nel piano delle fasi (y,y')

**>>plot(y(:,1),y(:,2))**
**>>hold**
**>>plot(2,0,'*')**

(***piano y,y'***, l'asterisco evidenzia la condizione iniziale)

# Equazione di Van der Pol

Fissiamo $\mu$=**1000** (forte smorzamento) e correggiamo ***vdpol.m***

```
function yp=vdpol(t,y)
 % equazione di Van der Pol
mu=1000;
yp=[y(2);mu*(1-y(1)^2)*y(2)-y(1)];
```

Rilanciamo il solver ***ode45*** nell'intervallo [0 1000] e plottiamo i risultati

```
>> [t,y]=ode45(@vdpol,[0  1000],[2  0]);
```

# Equazione di Van der Pol

- Osserviamo come il solutore impieghi molto tempo per integrare il sistema di Van der Pol; il problema è diventato ***stiff*** perché le **componenti della soluzione Y subiscono variazioni su scale temporali molto diverse tra loro.**
- Il metodo numerico risente di questo squilibrio nelle scale temporali e ne paga il prezzo in termini di scelta sul passo di integrazione che in alcuni punti risulta molto stringente.
- In questo caso è opportuno ricorrere alla classe di solutori, indicati con il suffisso ***"s"*** appositamente studiati per questo genere di problemi,
- ad esempio ***ode15s*** e ***ode23s***.

# Equazione di Van der Pol

```
>> [t,y]=ode15s(@vdpol,[0  1000],[2  0]);
>>semilogy(t,y(:,1),'-',t,y(:,2),'o');
```

- Ora i tempi di calcolo sono notevolmente diminuiti.
- Il grafico in scala semilogaritmica delle due componenti evidenzia chiaramente le diverse scale temporali caratteristiche di ciascuna componente, da cui la natura ***stiff*** del problema.

# Equazione di Van der Pol

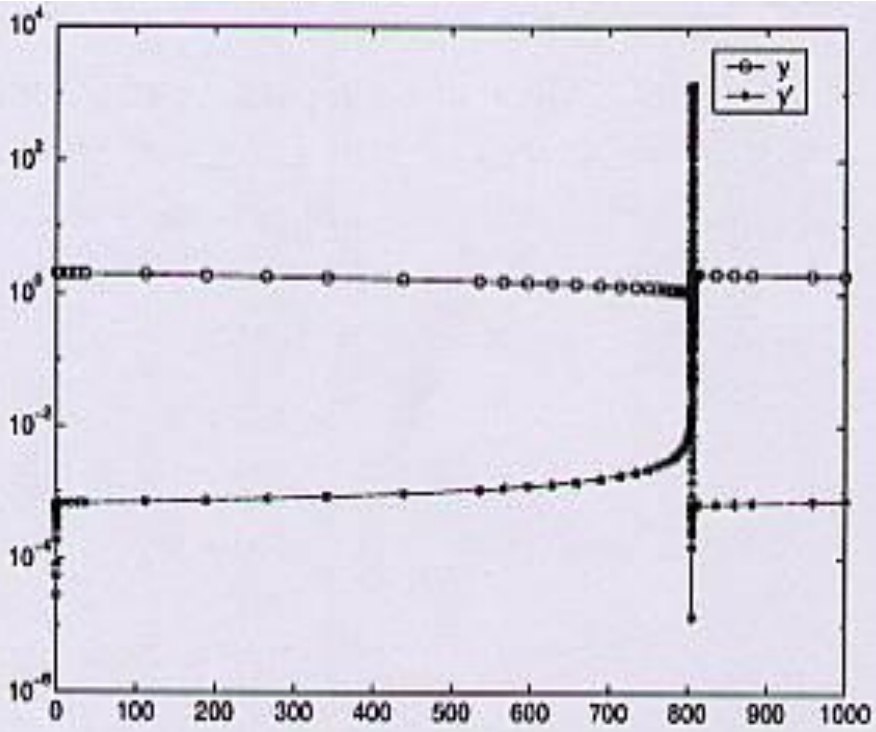



Figura 33: Soluzione dell'equazione di Van der Pol per $\mu = 1000$ ottenuta con un solver specifico per problemi *stiff*. Si notino le diverse scale di variazione delle due componenti (rappresentate in valore assoluto)

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA


# Serena Morigi

Dipartimento di Matematica

morigi@dm.unibo.it

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