# Equazioni Differenziali Ordinarie in MatLab

Manolo Venturin

Università degli Studi di Padova
Dip. Matematica Pura ed Applicata

2008

# Problema scalare

- Obiettivo
  - Risoluzione del problema di Cauchy

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

  - Equazione scalare

- Metodo risolutivo
  - Metodo di Eulero esplicito
  - Metodo di Heun
  - Metodo di Runge-Kutta del IV-ordine

- Problemi test
  1. $y' = \lambda y$ con $y(0) = 1$ e $\lambda = -1$
  2. $y' = \frac{1}{1+t^2} - 2y^2$ con $y(0) = 0$

# Equazioni test

- Equazione test usata per lo studio della stabilità del metodo

$$\begin{cases} y' = \lambda y \\ y(0) = 1 \end{cases}$$

con $\lambda = -1$. La soluzione teorica è $y(t) = y_0 exp(\lambda t)$.

- Seconda equazione test con soluzione teorica

$$\begin{cases} y' = \dfrac{1}{1+t^2} - 2y^2 \\ y(0) = 0 \end{cases}$$

La soluzione teorica è $y(t) = \frac{t}{1+t^2}$.

# Equazioni test - codice

```
function dydt = eqtest1(t,y)
% EQTEST1 Equazione test per la stabilita'
%  SINTASSI
%     dydt = eqtest1(t,y)
%  INPUT
%     t : Valore corrente della variabile indipendente
%     y : Valore corrente della variabile dipendente
% OUTPUT
%     dydt : Valore di f(t,y) in t e y
% NOTE
%     Con y(0) = 0 la soluzione teorica e'
%     y(y) = exp(-t)

lambda = -1;
dydt = lambda*y;
```

# Equazioni test - codice

```
% FILE : DISEGNO1.m
% Disegno della soluzione 1

% punti su cui valutare la soluzione
t = linspace(0,10,100);

% valutazione della soluzione
y = exp(-t);

% disegno della soluzione
h = plot(t,y,'k-');
set(h,'LineWidth',2);
title('Equazione test : y(t) = exp(-t)');
xlabel('t'); ylabel('y(t)');
```

# Equazioni test - codice

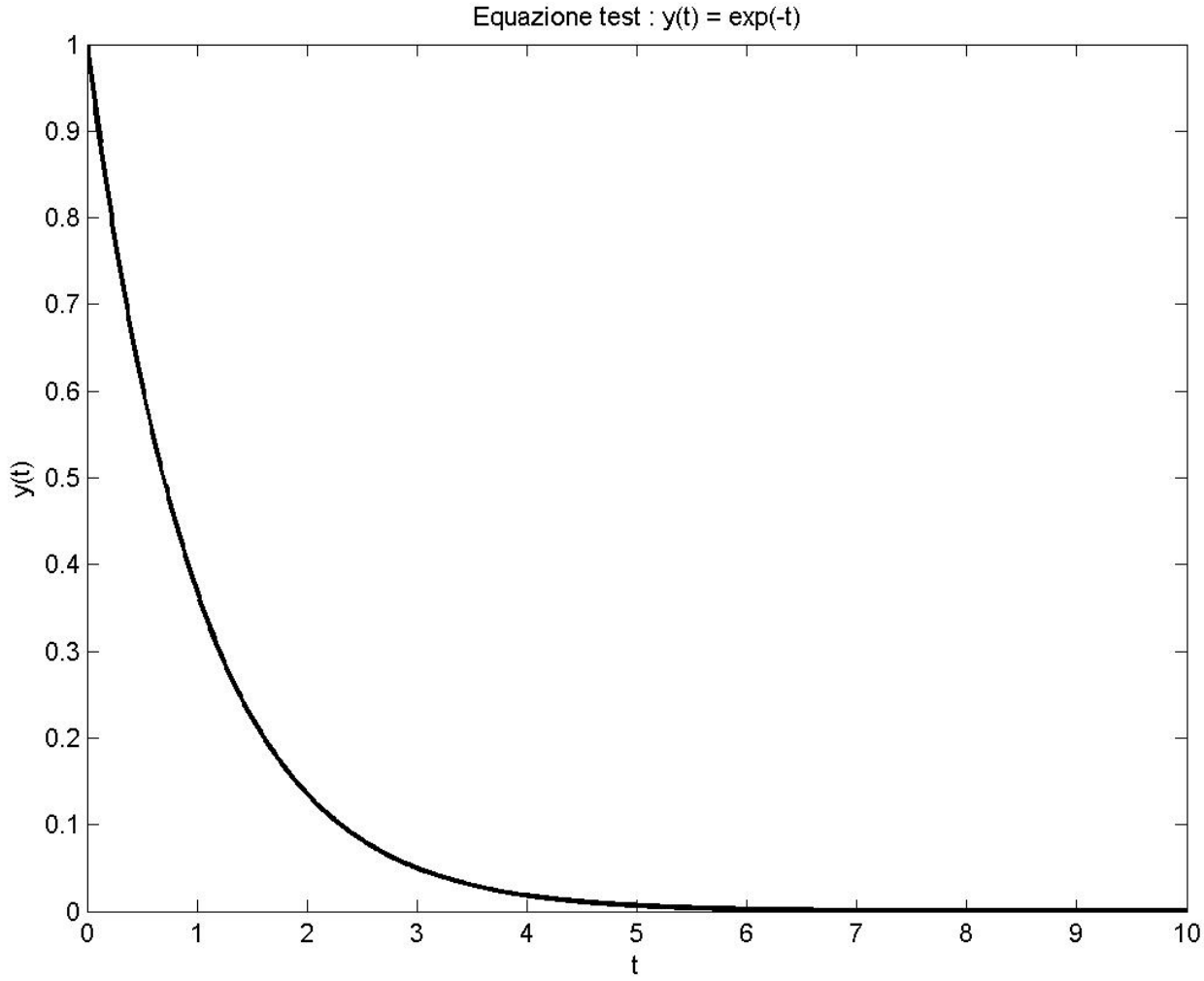


Figura: Andamento della soluzione 1

# Equazioni test - codice

```
function dydt = eqtest2(t,y)
% EQTEST2 Equazione test con soluzione analitica
%  SINTASSI
%     dydt = eqtest2(t,y)
%  INPUT
%     t : Valore corrente della variabile indipendente
%     y : Valore corrente della variabile dipendente
% OUTPUT
%     dydt : Valore di f(t,y) in t e y
% NOTE
%     Con y(0) = 0 la soluzione teorica e' :
%     y(t) = t/(1+t^2)

dydt = 1./(1+t.^2) - 2*y.^2;
```

# Equazioni test - codice

```
% FILE : DISEGNO2.m
% Disegno della soluzione 2

% punti su cui valutare la soluzione
t = linspace(0,10,100);

% valutazione della soluzione
y = t./(1+t.^2);

% disegno della soluzione
h = plot(t,y,'k-');
set(h,'LineWidth',2);
title('Equazione test : y(t) = t/(1+t^2)');
xlabel('t'); ylabel('y(t)');
```

# Equazioni test - codice

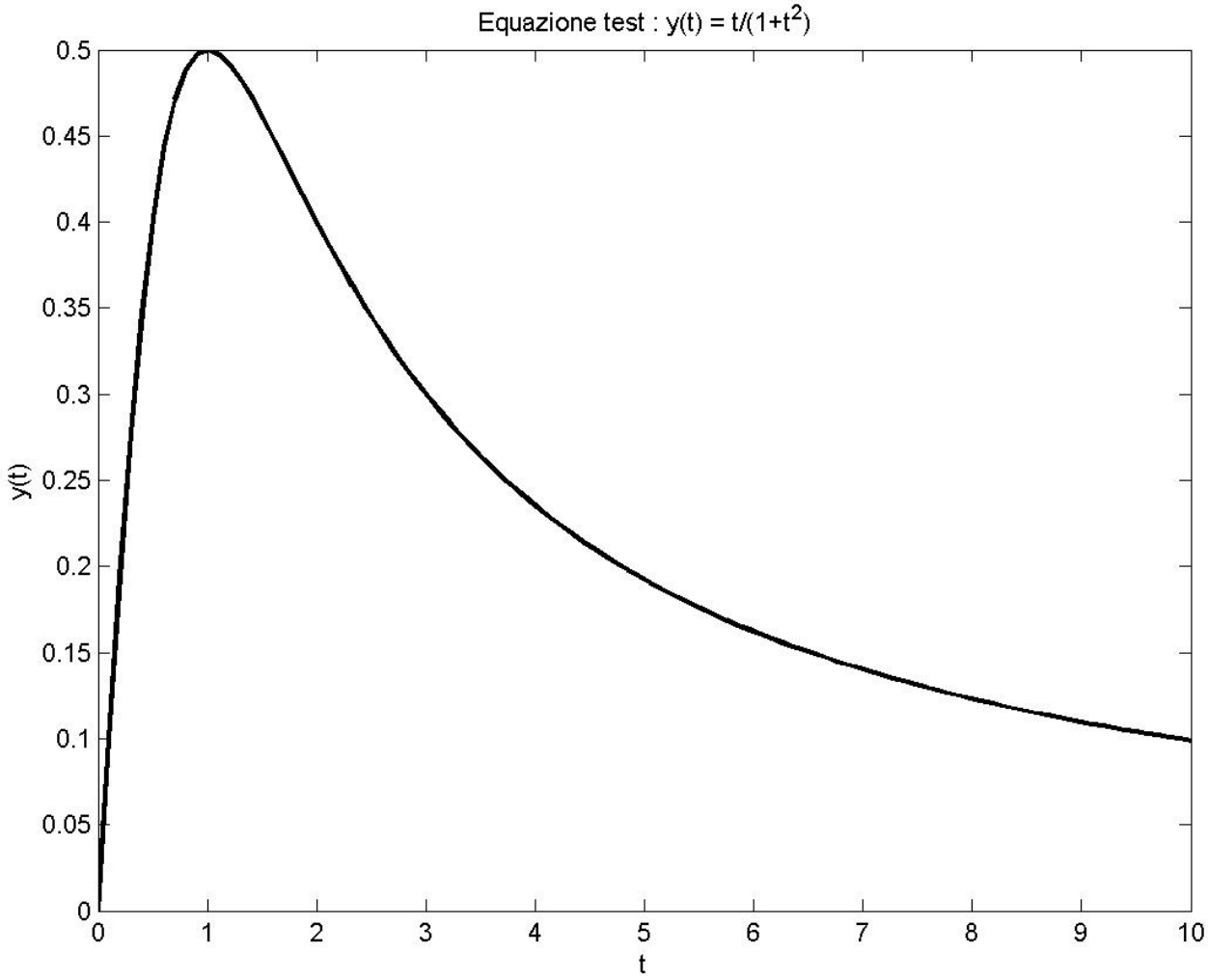


Figura: Andamento della soluzione 2

# Metodo di Eulero esplicito

- Formula di avanzamento del metodo

$$y_{n+1} = y_n + hf(x_n, y_n)$$

con $y_0$ fissato.

- Convergenza sull'equazione test $y' = \lambda y$ con $y(0) = 1$

$$|1 + h\lambda| < 1$$

- Ordine di convergenza $O(h)$

# Metodo di Eulero esplicito

```
% FILE: stabEulero.m
% Disegno regione di stabilita'
% metodo di Eulero esplicito

% Definizione della griglia
[x,y] = meshgrid(-3:0.01:1,-2:0.01:2);

% valutazione di |1+lambda h|
% Re(h lambda) = x(i) , Im(h lambda) = y(i)
z = double(abs(1+(x+i*y))>=1);

contourf(x,y,z,1);
colormap((gray+1)/2); axis ij; grid off
hold on
h = line([-3,1],[0,0]); set(h,'Color','k');
h = line([0,0],[-2,2]); set(h,'Color','k');
hold off
```

# Metodo di Eulero esplicito

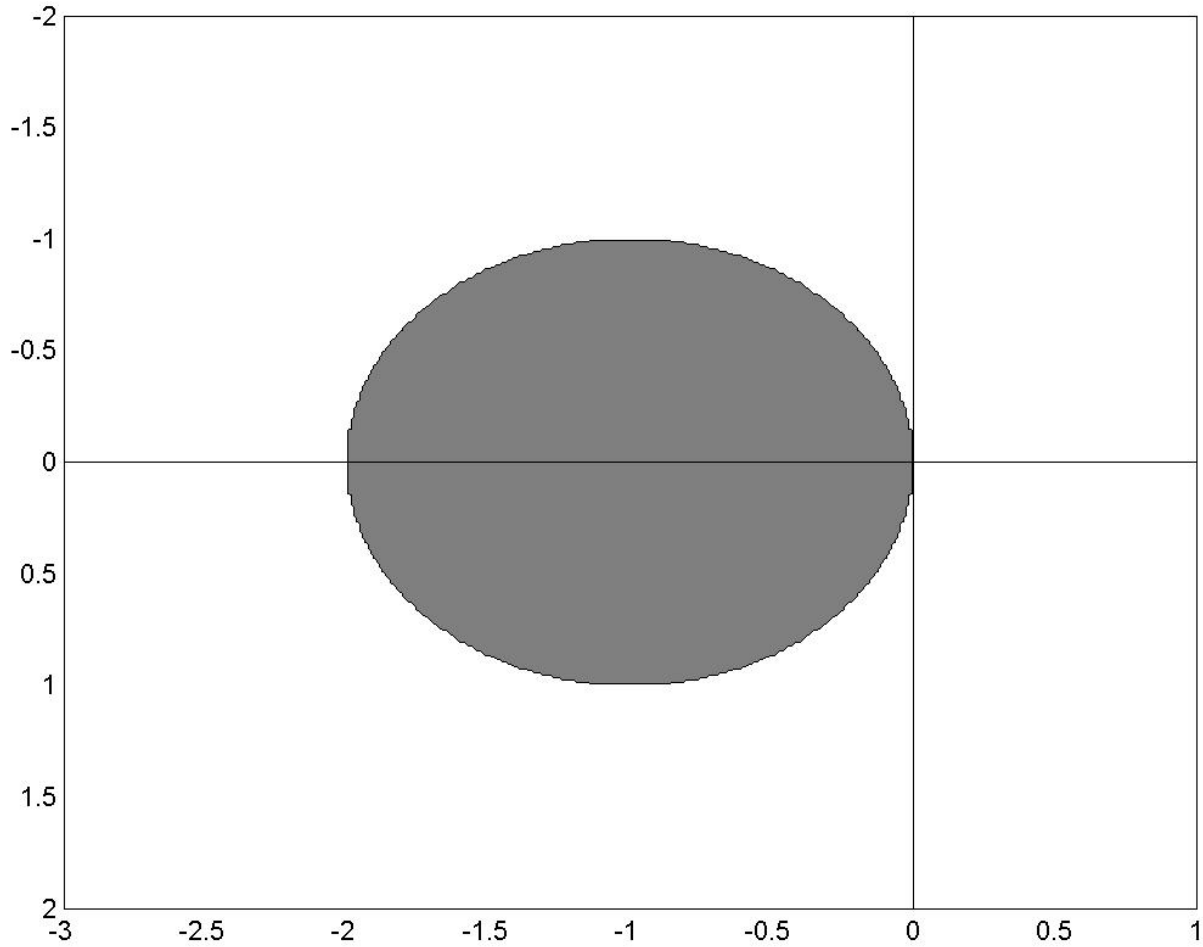


Figura: Regione di assoluta stabilità: Metodo di Eulero esplicito

# Metodo di Eulero esplicito - codice

```
function [t,y] = odeEulero(odefun,tspan,y0,h)
%ODEULERO Metodo di Eulero esplicito per ODE
%  SINTASSI
%     [t,y] = odeEuler(odefun,tspan,y0,h)
%  INPUT
%     odefun : La funzione f(t,y) del problema differenzale
%              y'= f(t,y)
%     tspan  : Valore iniziale tspan(1) e finale tspan(2)
%              della procedura di integrazione
%     y0     : Condizione initiale var. dipendente
%     h      : passo temporale
% OUTPUT
%     t : Vettore dei valori delle variabili indipendenti
%     y : Vettore dei valori delle variabili dipendenti

% Autore : M. Venturin
```

# Metodo di Eulero esplicito - codice

```
% pre-allocazione memoria var. indipendenti
t = (tspan(1):h:tspan(2))';
% numero totale di elementi
n = length(t);
% pre-allocazione memoria var. dipendenti
y = zeros(n,1);
% valore iniziale
y(1) = y0;

% ciclo principale
for j=1:n-1
   % y_{n+1} = y_{n} + h f(x_n,y_n)
   y(j+1) = y(j)+h*feval(odefun,t(j),y(j));
end
```

# Metodo di Eulero esplicito - codice

```
% FILE: main1_Eulero.m
% Esempio di utilizzo della routine di Eulero
% per diversi valori di h

tspan = [0,10]; y0 = 1;
h = 0.5; [t1,y1] = odeEulero('eqtest1',tspan,y0,h);
h = 1.5; [t2,y2] = odeEulero('eqtest1',tspan,y0,h);
h = 2.1; [t3,y3] = odeEulero('eqtest1',tspan,y0,h);

% Soluzione esatta
t = linspace(tspan(1),tspan(2),100); y = exp(-t);

% Disegno delle soluzioni
h = plot(t,y,'k-',t1,y1,'bo-',t2,y2,'ro-',t3,y3,'go-');
set(h,'LineWidth',2);
legend('exact','h=0.5','h=1.5','h=2.1');
```

# Metodo di Eulero esplicito - codice

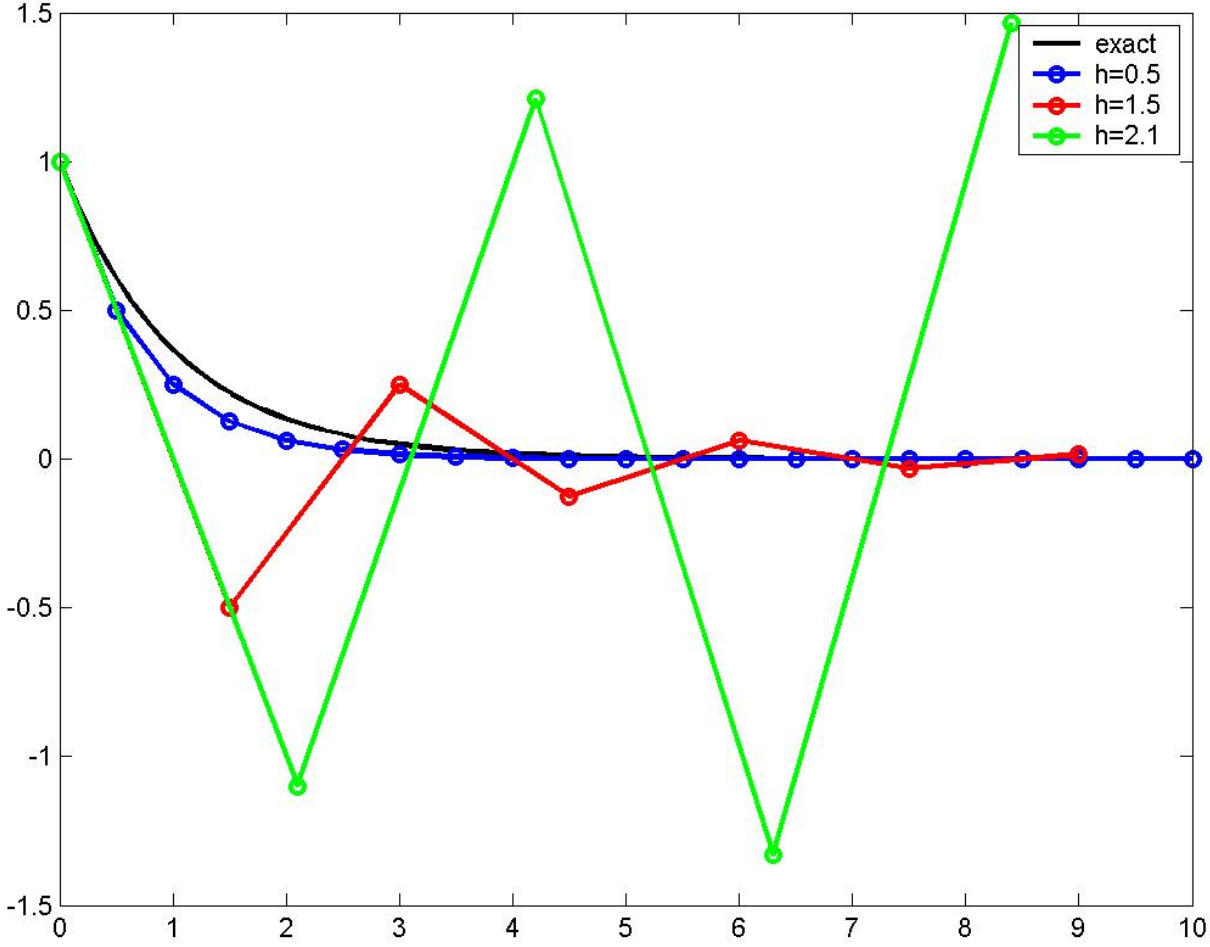


Figura: Andamento delle soluzioni per diversi valori di $h$

# Metodo di Eulero esplicito - codice

```
% FILE: main2_Eulero.m
% Esempio di utilizzo della routine di Eulero
% per diversi valori di h

% lambda stimato = -2 perche' f'_y = 4*y
% e dal grafico della soluzione teorica si ha che
% ymax = 1/2
tspan = [0,10]; y0 = 0;
h = 0.2; [t1,y1] = odeEulero('eqtest2',tspan,y0,h);
h = 0.9; [t2,y2] = odeEulero('eqtest2',tspan,y0,h);
h = 0.95; [t3,y3] = odeEulero('eqtest2',tspan,y0,h);

% Soluzione esatta
t = linspace(tspan(1),tspan(2),100); y = t./(1+t.^2);

% Disegno delle soluzioni
h = plot(t,y,'k-',t1,y1,'bo-',t2,y2,'ro-',t3,y3,'go-');
set(h,'LineWidth',2);
legend('exact','h=0.2','h=0.9','h=0.95');
```

# Metodo di Eulero esplicito - codice

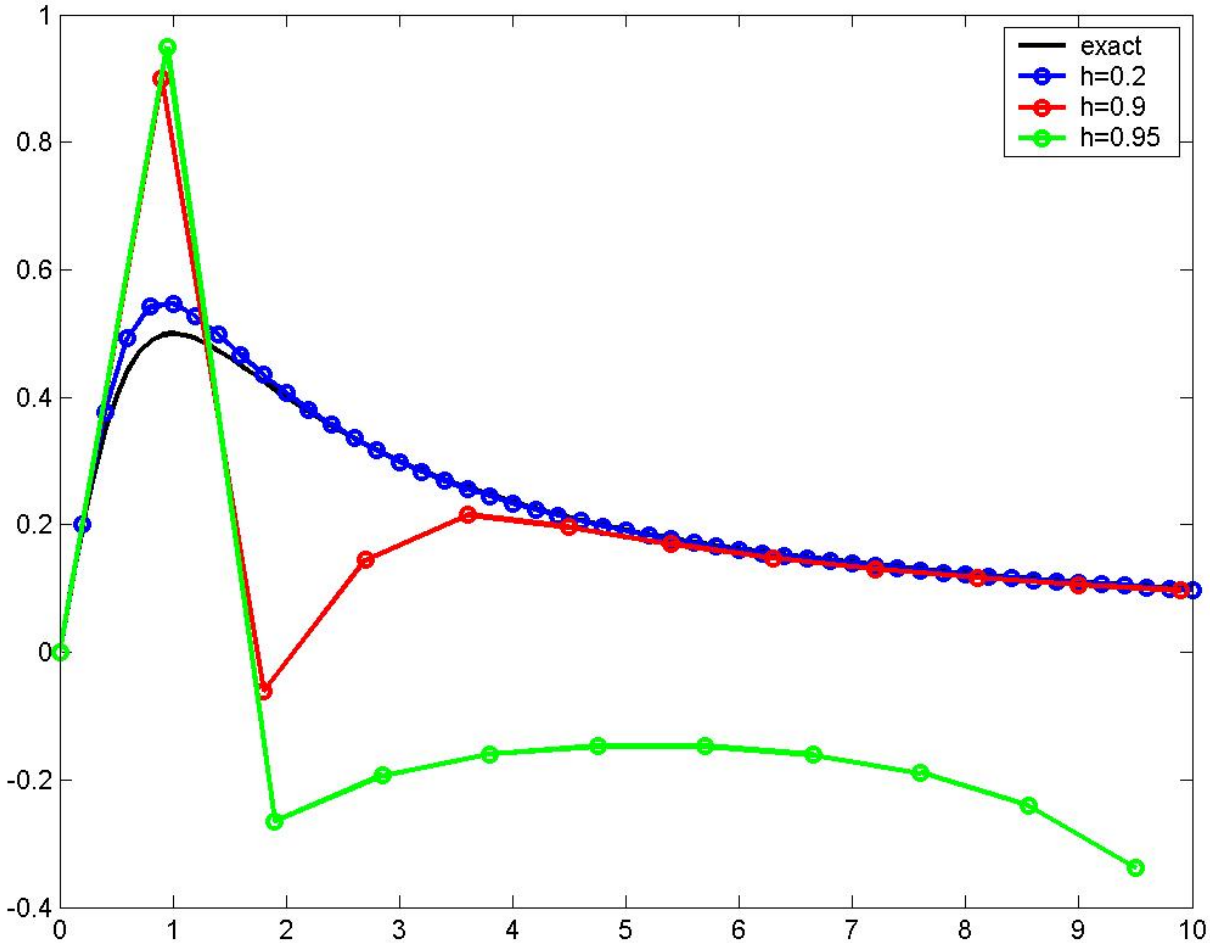


Figura: Andamento delle soluzioni per diversi valori di $h$

# Metodo di Heun

- Formula di avanzamento del metodo

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

con $y_0$ fissato.

- Convergenza sull'equazione test $y' = \lambda y$ con $y(0) = 1$

$$|1 + h\lambda + (h\lambda)^2/2| < 1$$

- Ordine di convergenza $O(h^2)$

# Metodo di Heun

```
% FILE: stabHeun.m
% Disegno regione di stabilita'
% metodo di Heun

% Definizione della griglia
[x,y] = meshgrid(-3:0.01:1,-2:0.01:2);

% valutazione di |1+lambda h +1/2(lambda h)^2|
% Re(h lambda) = x(i) , Im(h lambda) = y(i)
z = double(abs(1+(x+i*y)+0.5*(x+i*y).^2)<=1);

contourf(x,y,z,1);
colormap((gray+1)/2); axis ij; grid off
hold on
h = line([-3,1],[0,0]); set(h,'Color','k');
h = line([0,0],[-2,2]); set(h,'Color','k');
hold off
```

# Metodo di Heun

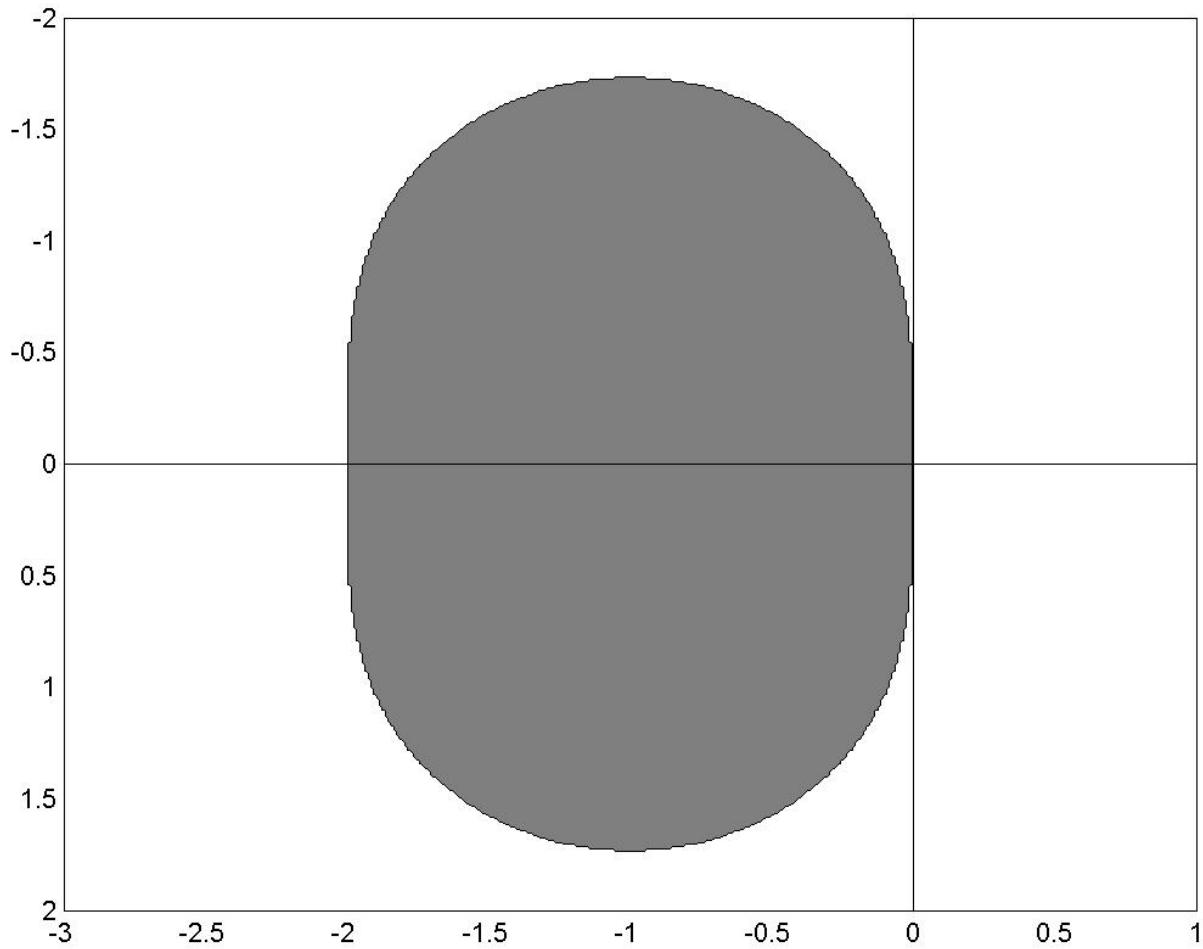


Figura: Regione di assoluta stabilità: Metodo di Heun

# Metodo di Heun - codice

```
function [t,y] = odeHeun(odefun,tspan,y0,h)
%ODEHEUN Metodo di Heun per ODE
%  SINTASSI
%     [t,y] = odeHeun(odefun,tspan,y0,h)
%  INPUT
%     odefun : La funzione f(t,y) del problema differenzale
%              y'= f(t,y)
%     tspan  : Valore iniziale tspan(1) e finale tspan(2)
%              della procedura di integrazione
%     y0     : Condizione initiale var. dipendente
%     h      : passo temporale
% OUTPUT
%     t : Vettore dei valori delle variabili indipendenti
%     y : Vettore dei valori delle variabili dipendenti

% Autore : M. Venturin
```

# Metodo di Heun - codice

```
% pre-allocazione memoria var. indipendenti
t = (tspan(1):h:tspan(2))';
% numero totale di elementi
n = length(t);
% pre-allocazione memoria var. dipendenti
y = zeros(n,1);
% valore iniziale
y(1) = y0;

% ciclo principale
for j=1:n-1
   % f(x_n,y_n)
   fxyn = feval(odefun,t(j),y(j));
   % aggiornamento soluzione
   y(j+1) = y(j)+...
      0.5*h*(fxyn+feval(odefun,t(j+1),y(j)+h*fxyn));
end
```

# Metodo di Heun - codice

```
% FILE: main1_Heun.m
% Esempio di utilizzo della routine di Heun
% per diversi valori di h

tspan = [0,10]; y0 = 1;
h = 0.5; [t1,y1] = odeHeun('eqtest1',tspan,y0,h);
h = 1.5; [t2,y2] = odeHeun('eqtest1',tspan,y0,h);
h = 2.1; [t3,y3] = odeHeun('eqtest1',tspan,y0,h);

% Soluzione esatta
t = linspace(tspan(1),tspan(2),100); y = exp(-t);

% Disegno delle soluzioni
h = plot(t,y,'k-',t1,y1,'bo-',t2,y2,'ro-',t3,y3,'go-');
set(h,'LineWidth',2);
legend('exact','h=0.5','h=1.5','h=2.1');
```

# Metodo di Heun - codice

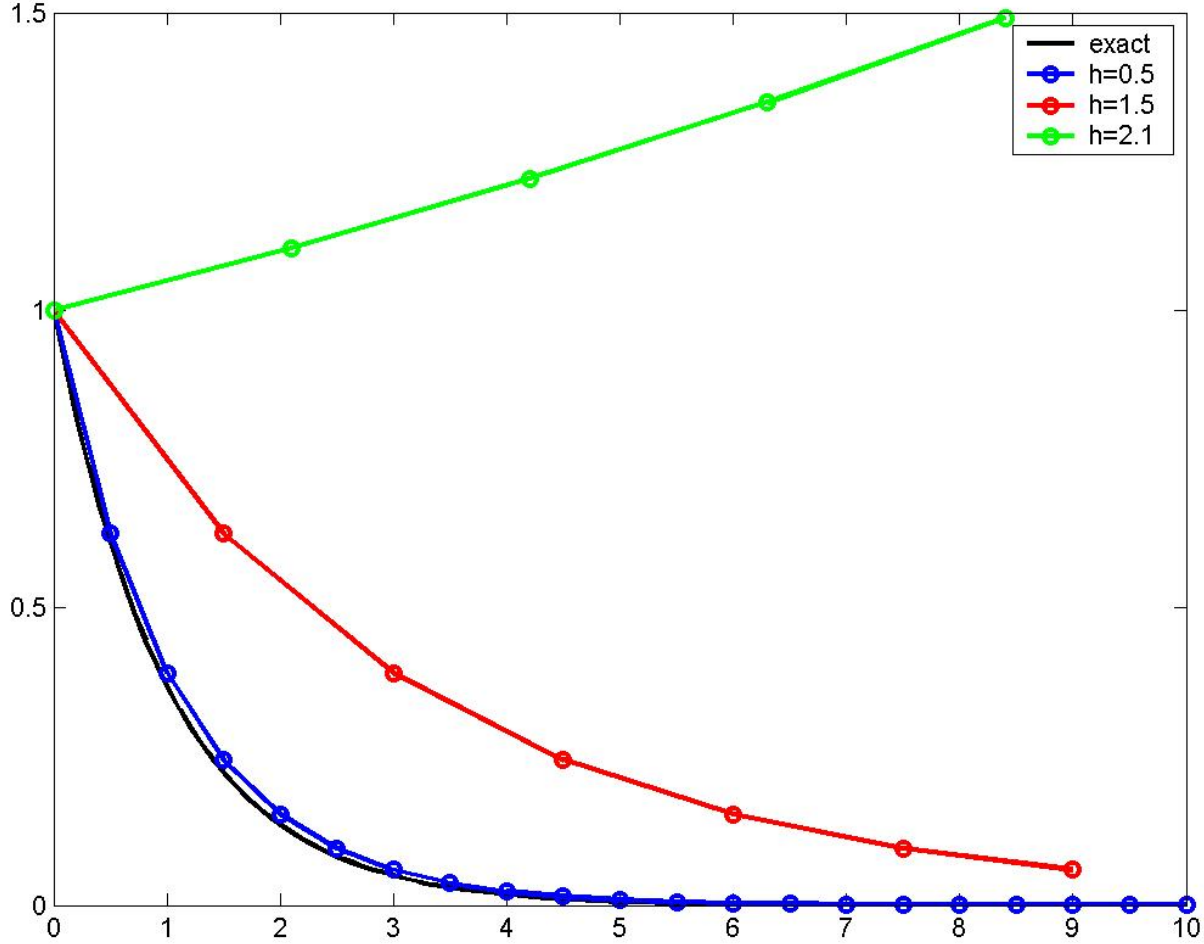


Figura: Andamento delle soluzioni per diversi valori di $h$

# Metodo di Heun - codice

```
% FILE: main2_Heun.m
% Esempio di utilizzo della routine di Heun
% per diversi valori di h

tspan = [0,10]; y0 = 0;
h = 0.2; [t1,y1] = odeHeun('eqtest2',tspan,y0,h);
h = 0.9; [t2,y2] = odeHeun('eqtest2',tspan,y0,h);
h = 0.95; [t3,y3] = odeHeun('eqtest2',tspan,y0,h);

% Soluzione esatta
t = linspace(tspan(1),tspan(2),100); y = t./(1+t.^2);

% Disegno delle soluzioni
h = plot(t,y,'k-',t1,y1,'bo-',t2,y2,'ro-',t3,y3,'go-');
set(h,'LineWidth',2);
legend('exact','h=0.2','h=0.9','h=0.95');
```

# Metodo di Heun - codice

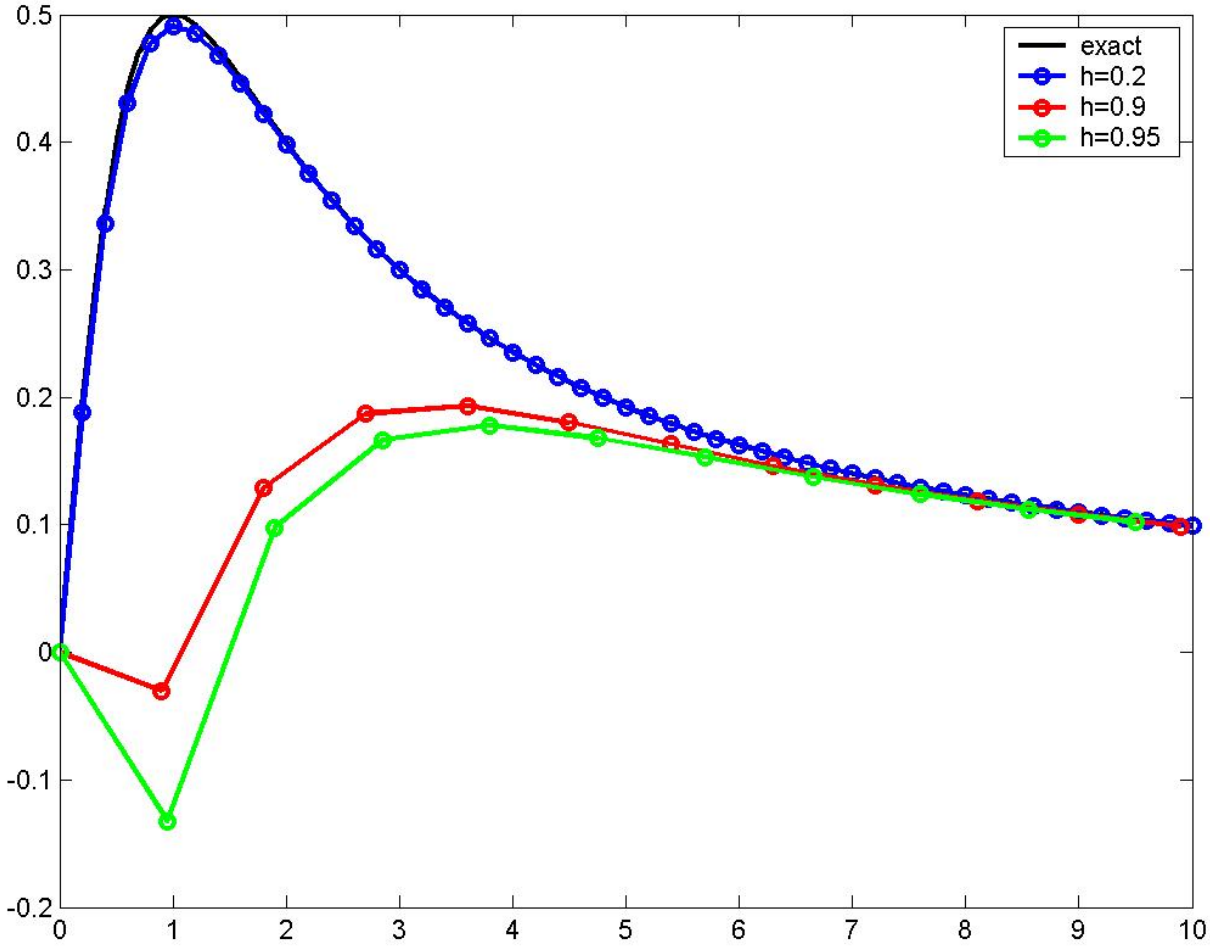


Figura: Andamento delle soluzioni per diversi valori di $h$

# Metodo di Runge-Kutta del IV-ordine

- Formula di avanzamento del metodo

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= hf(x_n + \frac{h}{2}, y_n + + \frac{k_2}{2}) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

con $y_0$ fissato.

- Convergenza sull'equazione test $y' = \lambda y$ con $y(0) = 1$

$$|1 + h\lambda + (h\lambda)^2/2 + (h\lambda)^3/3! + (h\lambda)^4/4!| < 1$$

- Ordine di convergenza $O(h^4)$

# Metodo di Runge-Kutta del IV-ordine

```
% FILE: stabRK4.m
% Disegno regione di stabilita'
% metodo di Runge–Kutta IV ordine

% Definizione della griglia
[x,y] = meshgrid(–3:0.01:1,–3:0.01:3);

% valutazione di |1+lambda h +1/2(lambda h)^2+
%                 1/6(lambda h)^3+1/24(lambda h)^4|
% Re(h lambda) = x(i) , Im(h lambda) = y(i)
z = double(abs(1+(x+i*y) + 1/2*(x+i*y).^2 ...
   +1/6*(x+i*y).^3 + 1/24*(x+i*y).^4)≥1);

contourf(x,y,z,1);
colormap((gray+1)/2); axis ij; grid on
hold on
h = line([–3,1,2],[0,0,2]); set(h,'Color','k');
h = line([0,0],[–3,3]); set(h,'Color','k');
hold off
```

# Metodo di Runge-Kutta del IV-ordine

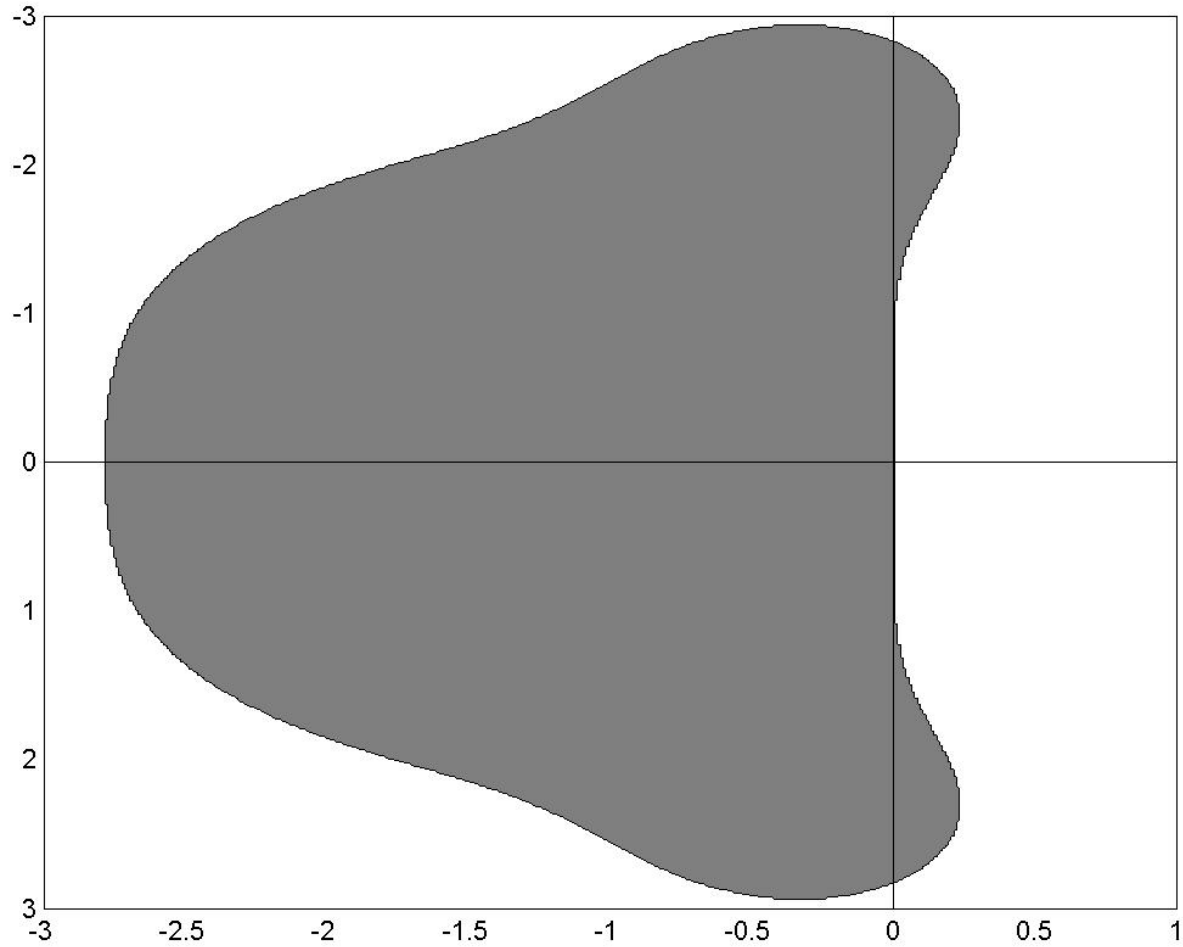


Figura: Regione di assoluta stabilità: Metodo Runge-Kutta IV-ordine

# Metodo di Runge-Kutta IV-ordine - codice

```
function [t,y] = odeRK4(odefun,tspan,y0,h)
%ODERK4 Metodo di Runge–Kutta IV per ODE
%  SINTASSI
%     [t,y] = odeRK4(odefun,tspan,y0,h)
%  INPUT
%     odefun : La funzione f(t,y) del problema differenzale
%              y'= f(t,y)
%     tspan  : Valore iniziale tspan(1) e finale tspan(2)
%              della procedura di integrazione
%     y0     : Condizione initiale var. dipendente
%     h      : passo temporale
% OUTPUT
%     t : Vettore dei valori delle variabili indipendenti
%     y : Vettore dei valori delle variabili dipendenti

% Autore : M. Venturin
```

# Metodo di Runge-Kutta IV-ordine - codice

```
% pre-allocazione memoria var. indipendenti
t = (tspan(1):h:tspan(2))';
% numero totale di elementi
n = length(t);
% pre-allocazione memoria var. dipendenti
y = zeros(n,1);
% valore iniziale
y(1) = y0;

% ciclo principale
for j=1:n-1
    k1 = h*feval(odefun,t(j),y(j));
    k2 = h*feval(odefun,t(j)+h/2,y(j)+k1/2);
    k3 = h*feval(odefun,t(j)+h/2,y(j)+k2/2);
    k4 = h*feval(odefun,t(j)+h,y(j)+k3);
    y(j+1) = y(j)+1/6*(k1+2*k2+2*k3+k4);
end
```

# Metodo di Runge-Kutta IV-ordine - codice

```
% FILE: main1_RK4.m
% Esempio di utilizzo della routine di RK4
% per diversi valori di h

tspan = [0,10]; y0 = 1;
h = 0.5; [t1,y1] = odeRK4('eqtest1',tspan,y0,h);
h = 1.5; [t2,y2] = odeRK4('eqtest1',tspan,y0,h);
h = 3.0; [t3,y3] = odeRK4('eqtest1',tspan,y0,h);

% Soluzione esatta
t = linspace(tspan(1),tspan(2),100); y = exp(-t);

% Disegno delle soluzioni
h = plot(t,y,'k-',t1,y1,'bo-',t2,y2,'ro-',t3,y3,'go-');
set(h,'LineWidth',2);
legend('exact','h=0.5','h=1.5','h=3.0');
```

# Metodo di Runge-Kutta IV-ordine - codice

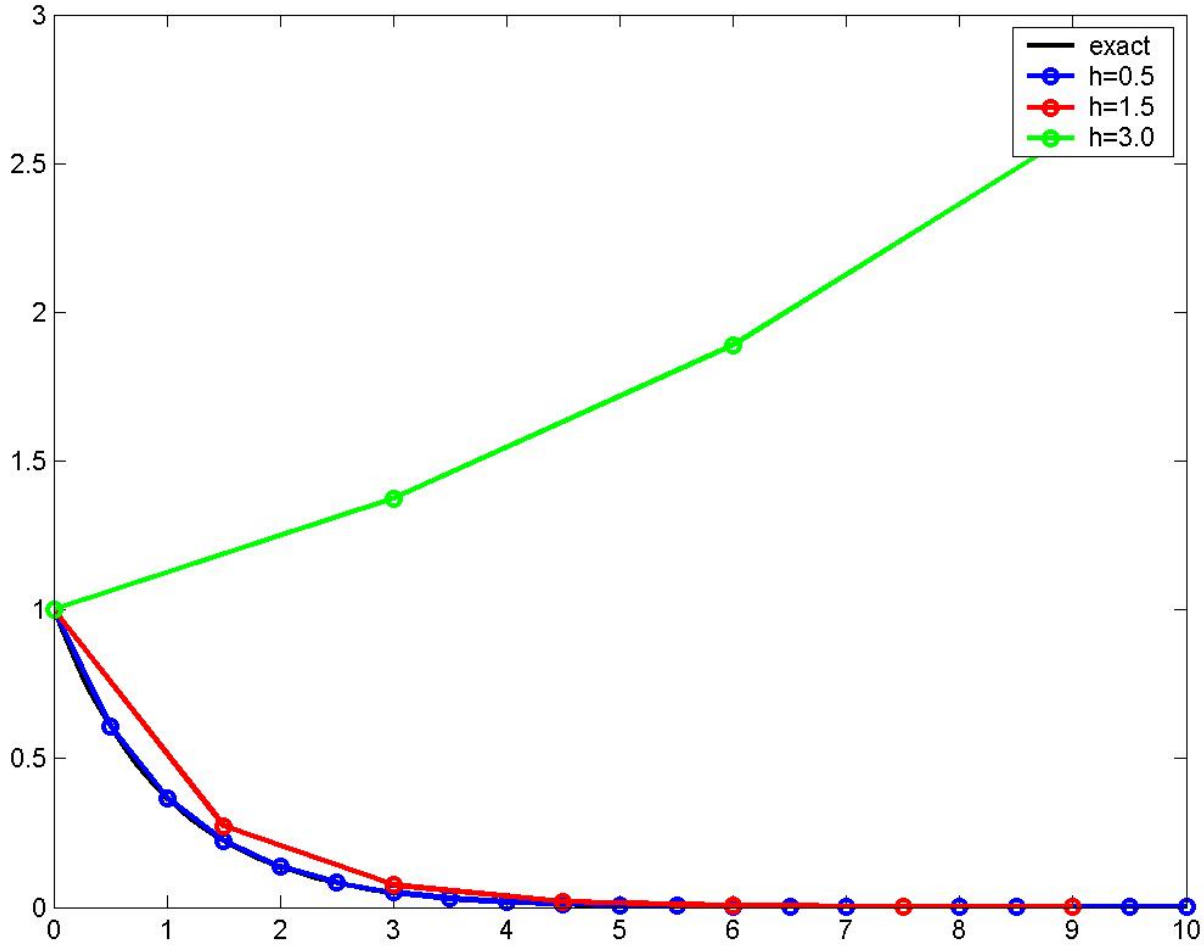


Figura: Andamento delle soluzioni per diversi valori di $h$

# Metodo di Runge-Kutta IV-ordine - codice

```
% FILE: main2_RK4.m
% Esempio di utilizzo della routine di RK4
% per diversi valori di h

tspan = [0,10]; y0 = 0;
h = 0.5; [t1,y1] = odeRK4('eqtest2',tspan,y0,h);
h = 1.5; [t2,y2] = odeRK4('eqtest2',tspan,y0,h);
h = 1.65; [t3,y3] = odeRK4('eqtest2',tspan,y0,h);

% Soluzione esatta
t = linspace(tspan(1),tspan(2),100); y = t./(1+t.^2);

% Disegno delle soluzioni
h = plot(t,y,'k-',t1,y1,'bo-',t2,y2,'ro-',t3,y3,'go-');
set(h,'LineWidth',2);
legend('exact','h=0.5','h=1.5','h=1.65');
```

# Metodo di Runge-Kutta IV-ordine - codice

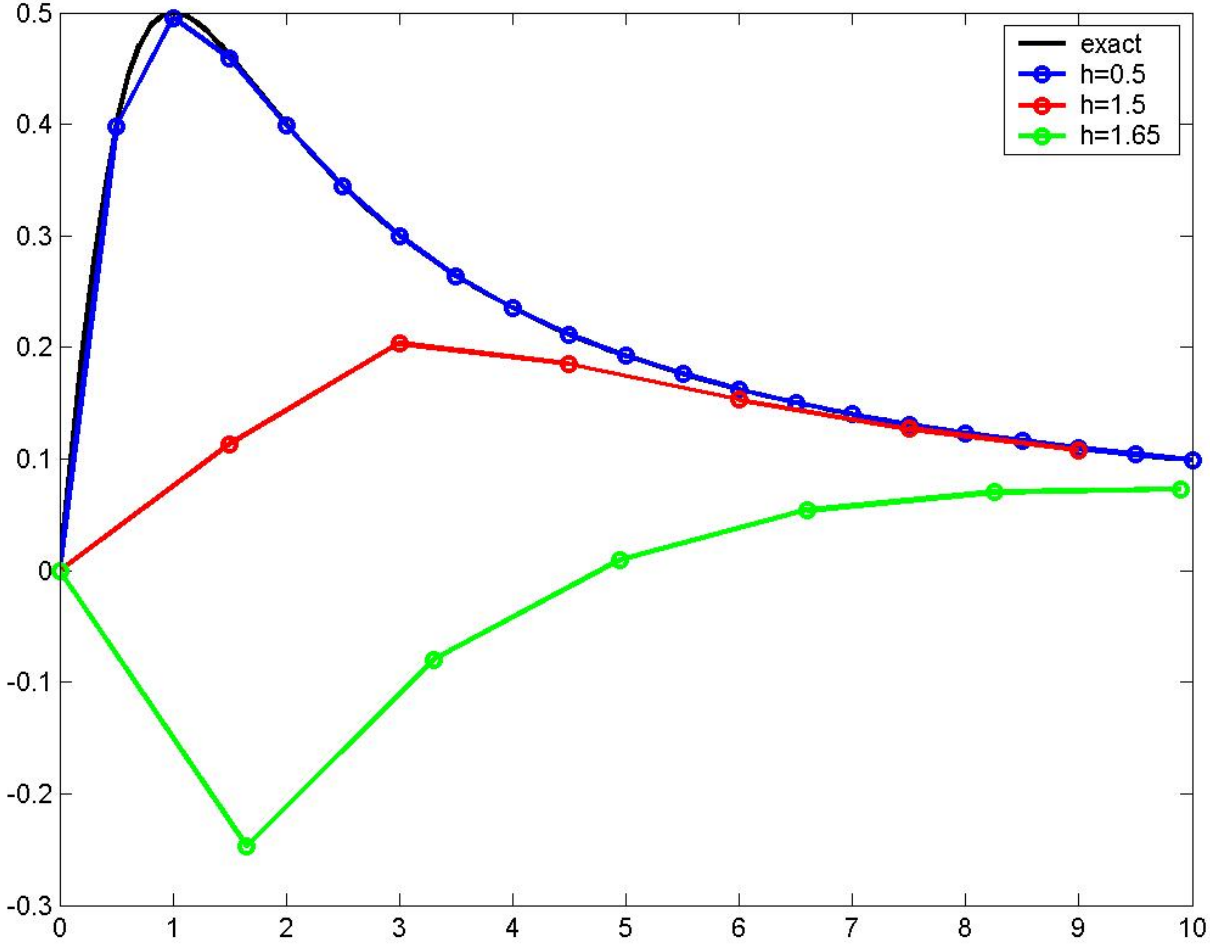


Figura: Andamento delle soluzioni per diversi valori di $h$

# Problema vettoriale

- Obiettivo
  - Risoluzione del problema di Cauchy

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

  - Equazione vettoriale in y.
- Metodo risolutivo
  - Metodo esplito di Runge-Kutta(4,5): ODE45.
- Problemi test
  1. Equazione del pendolo
  2. Oscillatore armonico forzato

# Pendolo

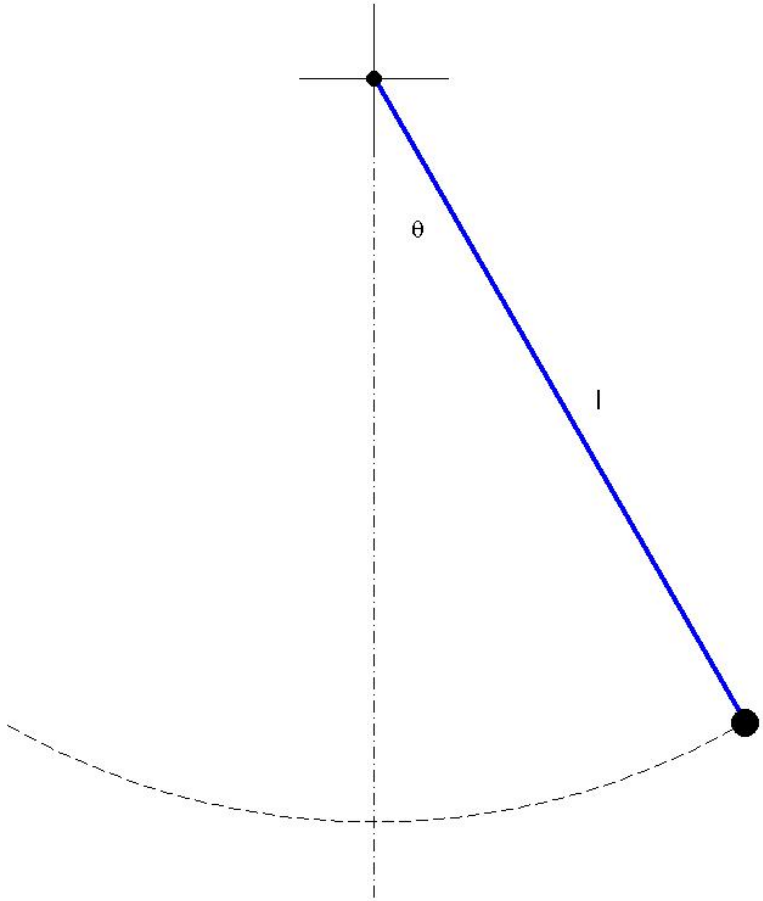


Figura: Pendolo

# Pendolo

- Equazione
$$m\theta'' + \frac{mg}{l}\sin\theta = 0$$

- Scrittura in forma di sistema (vettoriale)
Pongo $y_1 = \theta$ e $y_2 = \theta'$.
$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{l}\sin\theta \end{cases}$$

- Condizioni iniziali
$$\begin{cases} y_1(0) = \theta_0 \\ y_2(0) = \theta_0' \end{cases}$$

# Pendolo - codice

```
function dydt = eqPendolo(t,y)
% EQPENDOLO Equazione diff. pendolo
%  SINTASSI
%     dydt = eqPendolo(t,y)
%  INPUT
%     t : Valore corrente della variabile indipendente
%     y : Valore corrente della variabile dipendente
% OUTPUT
%     dydt : Valore di f(t,y) in t e y

% costante di gravita'
g = 9.81;
% lunghezza del pendolo
l = 10;

% equazione differenziale
dydt = [y(2);-g/l*sin(y(1))];
```

# Pendolo - codice

```
% FILE: mainPendolo.m
% Simulazione del pendolo

% tempo di simulazione
tspan = [0,20];
% pendolo parte con velocita' iniziale nulla
y0 = [pi/6;0];
% simulazione
options = odeset('Vectorized','on','InitialStep',0.01);
[t,y]=ode45('eqPendolo',tspan,y0,options);

% angolo
figure; plot(t,y(:,1));

% velocita' angolare
figure; plot(t,y(:,2));

% Mappa di Poincare' (posizione–velocita')
figure; plot(y(:,1),y(:,2));
```

# Pendolo - codice

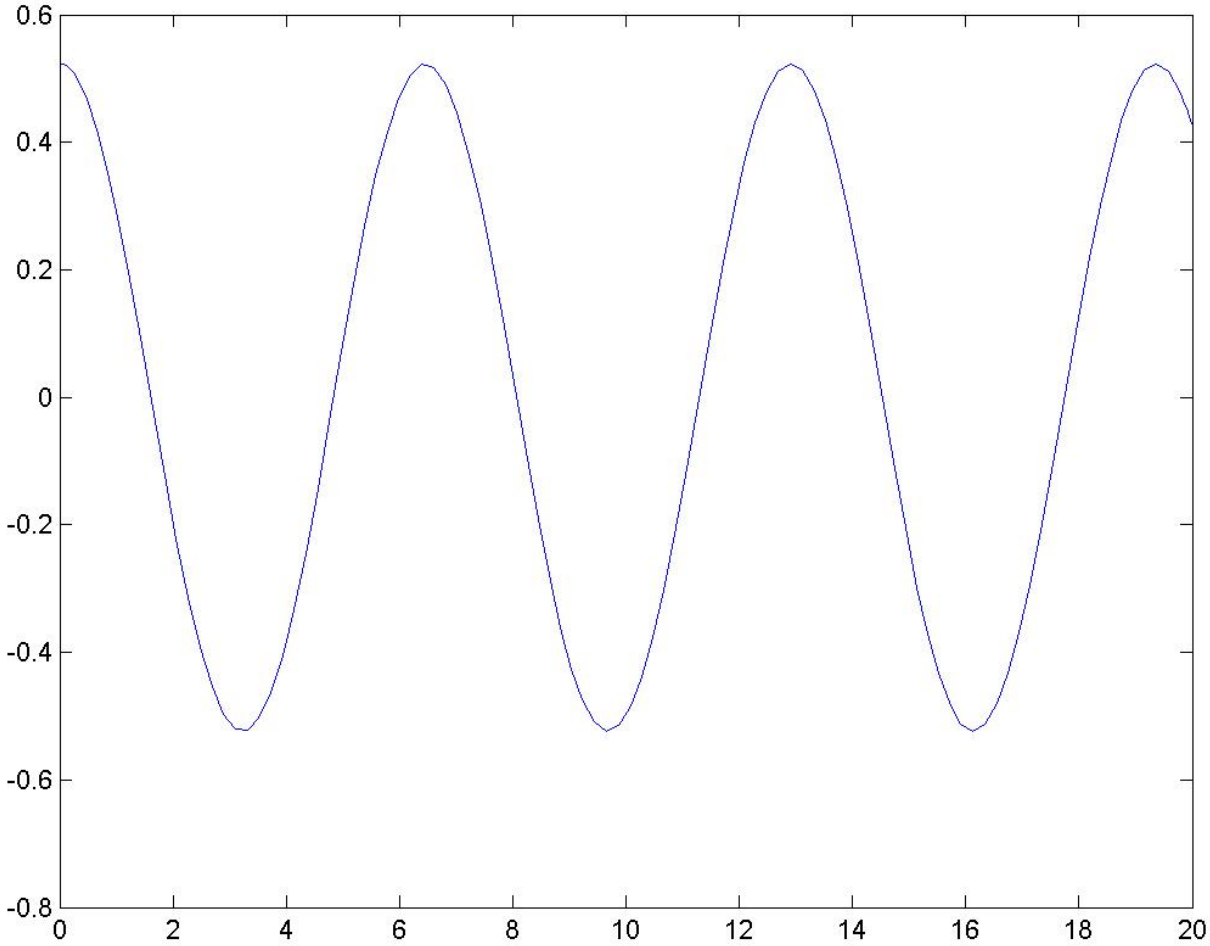


Figura: Andamento del angolo nel pendolo

# Pendolo - codice

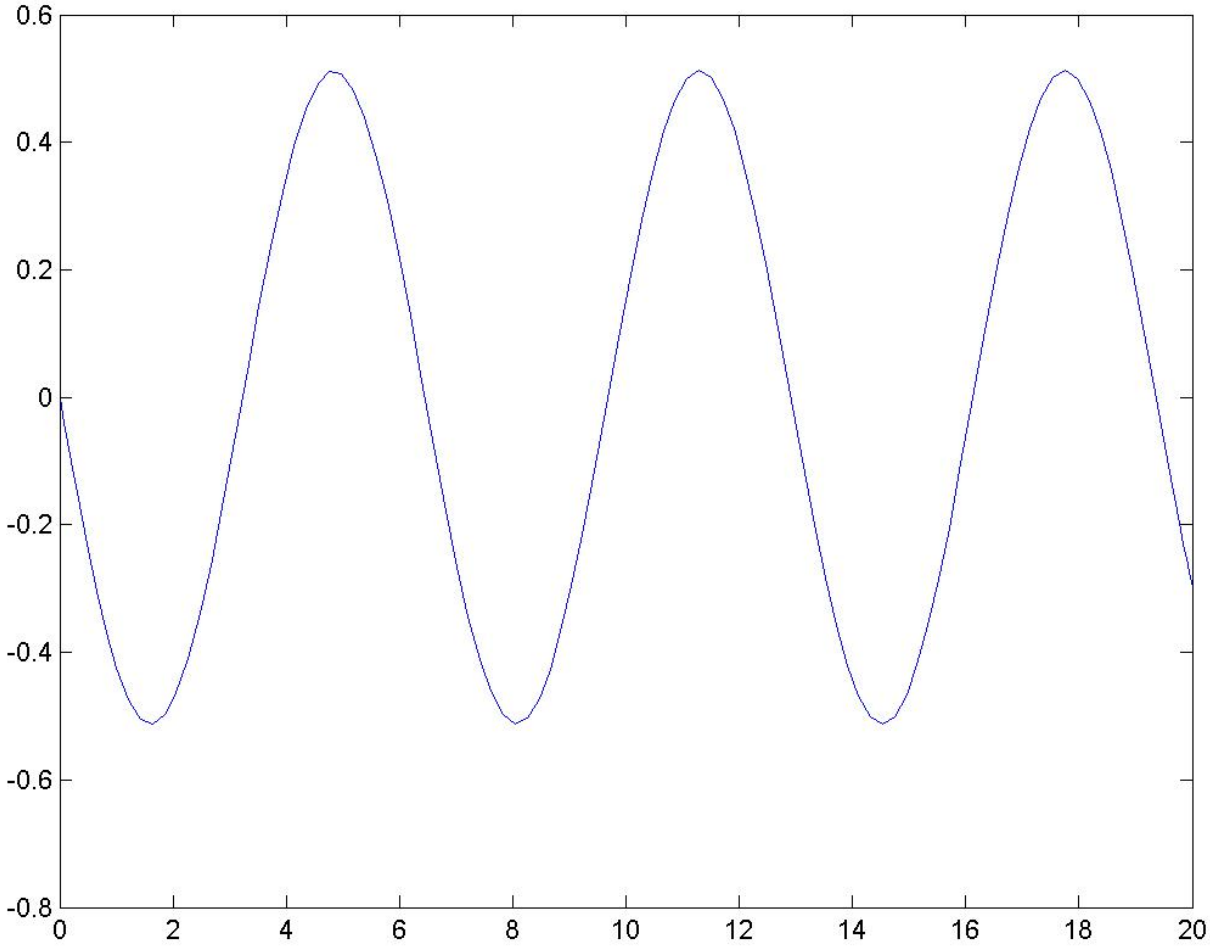


Figura: Andamento della velocità angolare nel pendolo

# Pendolo - codice

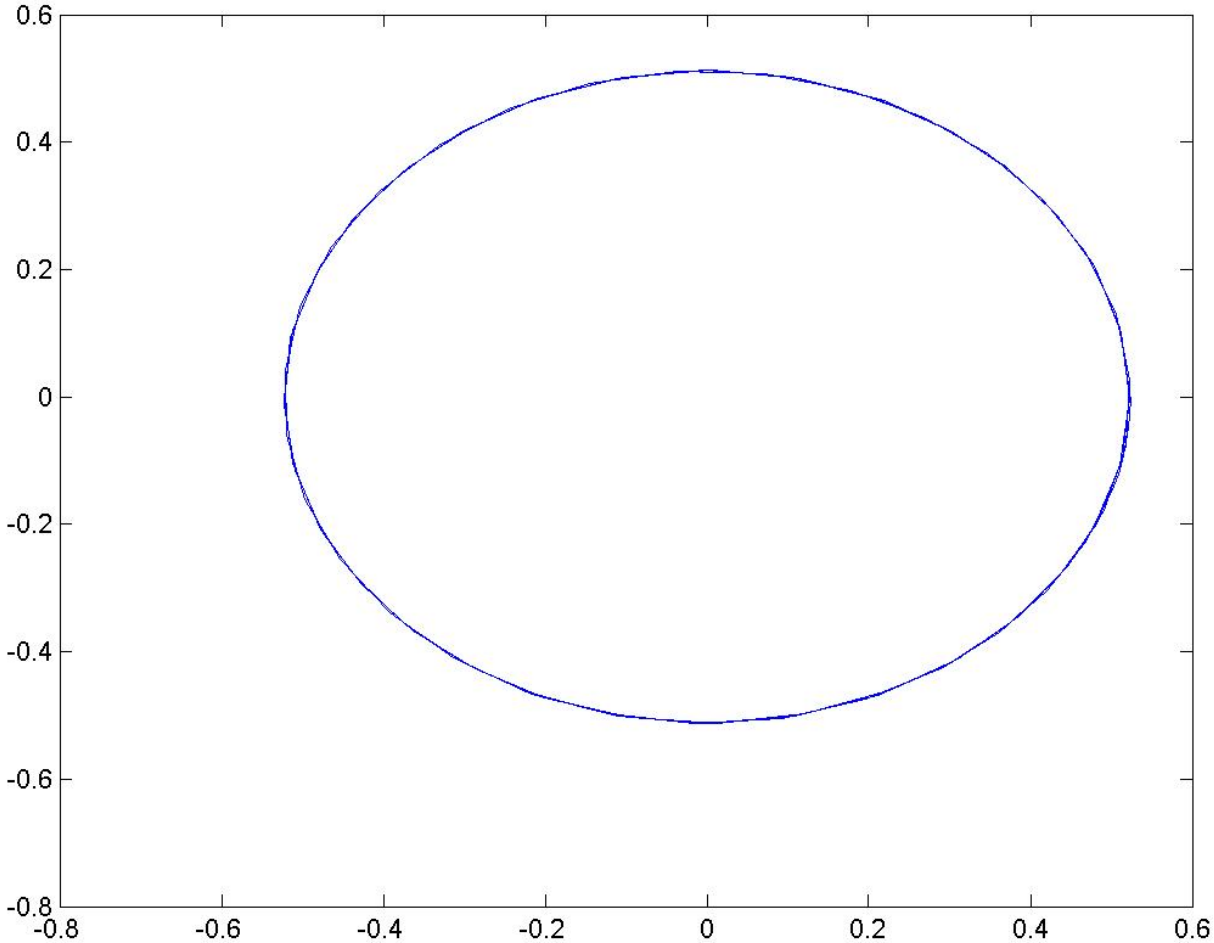


Figura: Mappa di Poincarè (posizione-velocità) del pendolo

# Oscillatore armonico forzato

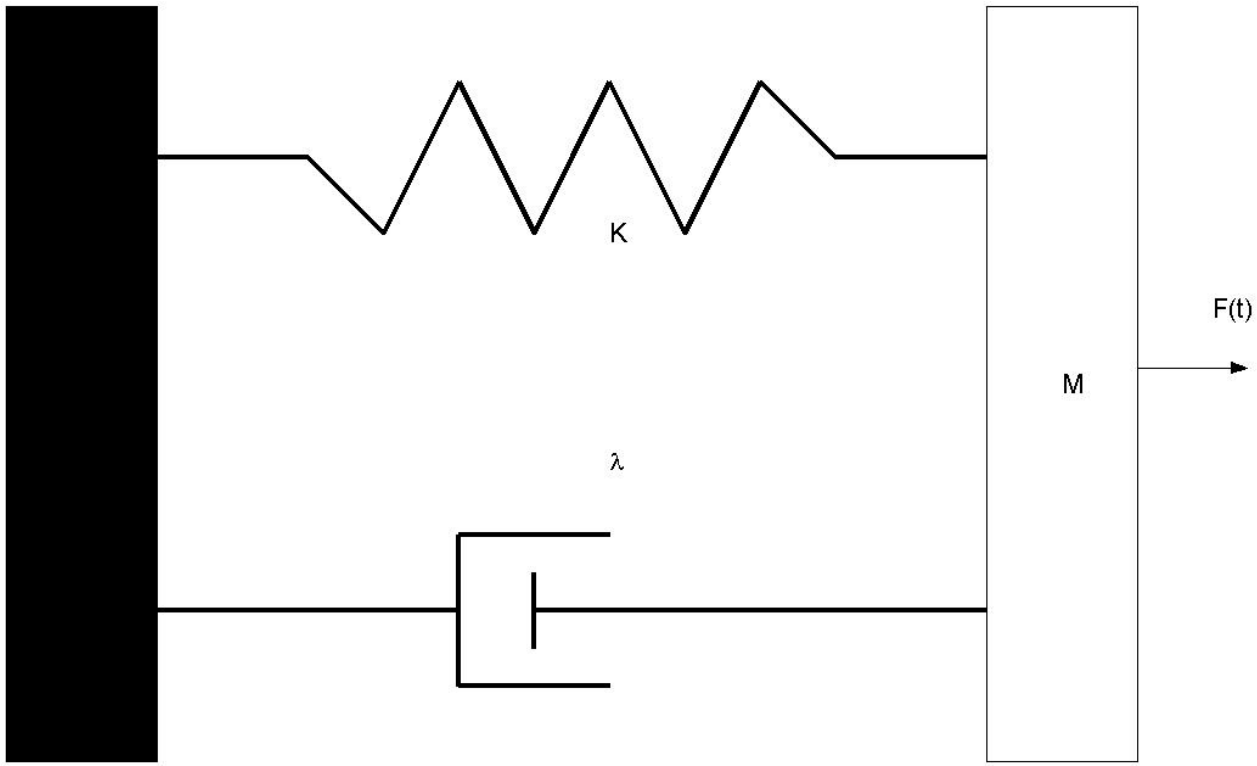


Figura: Oscillatore armonico

# Oscillatore armonico forzato

- Equazione
$$mx'' + \lambda x' + kx = F_0 \sin(\omega t)$$

- Scrittura in forma di sistema (vettoriale)
Pongo $y_1 = x$ e $y_2 = x'$.
$$\begin{cases} y_1' = y_2 \\ y_2' = -\dfrac{\lambda}{m} y_2 - \dfrac{k}{m} y_1 - \dfrac{F_0}{m} \sin(\omega t) \end{cases}$$

- Condizioni iniziali
$$\begin{cases} y_1(0) = x_0 \\ y_2(0) = x_0' \end{cases}$$

# Oscillatore - codice

```
function dydt = eqOsc(t,y,varargin)
% EQOSC Equazione diff. oscillatore forzato
%  SINTASSI
%     dydt = eqOsc(t,y,m,k,l,f0,w)
%  INPUT
%     t  : Valore corrente della variabile indipendente
%     y  : Valore corrente della variabile dipendente
%     m  : Massa
%     k  : Costante della molla
%     l  : Costante dello smorzatore
%     f0 : Costante della forzante
%     w  : Omega della formzante
% OUTPUT
%     dydt : Valore di f(t,y) in t e y

% Ottieni parametri (help odefile)
m  = varargin{2}; k  = varargin{3}; l  = varargin{4};
f0 = varargin{5}; w  = varargin{6};
% equazione differenziale
dydt = [y(2);-l/m*y(2)-k/m*y(1)+f0/m*sin(w*t)];
```

# Oscillatore - codice

```
% FILE: mainOsc.m
% Simulazione oscillatore armonico forzato
% Diverse casistiche

% OSCILLATORE LIBERO SMORZATO
% parametri del modello
m = 1; k = 10; l = 0.2; f0 = 0; w = 0;
% tempo di simulazione
tspan = [0,7];
% configurazione iniziale
y0 = [5;0];
% simulazione
options = odeset('Vectorized','on','InitialStep',0.01);
[t,y]=ode45('eqOsc',tspan,y0,options,m,k,l,f0,w);
% Posizione
figure; plot(t,y(:,1));xlabel('tempo');ylabel('ampiezza');
title('oscillazione libera');
```

# Oscillatore - codice

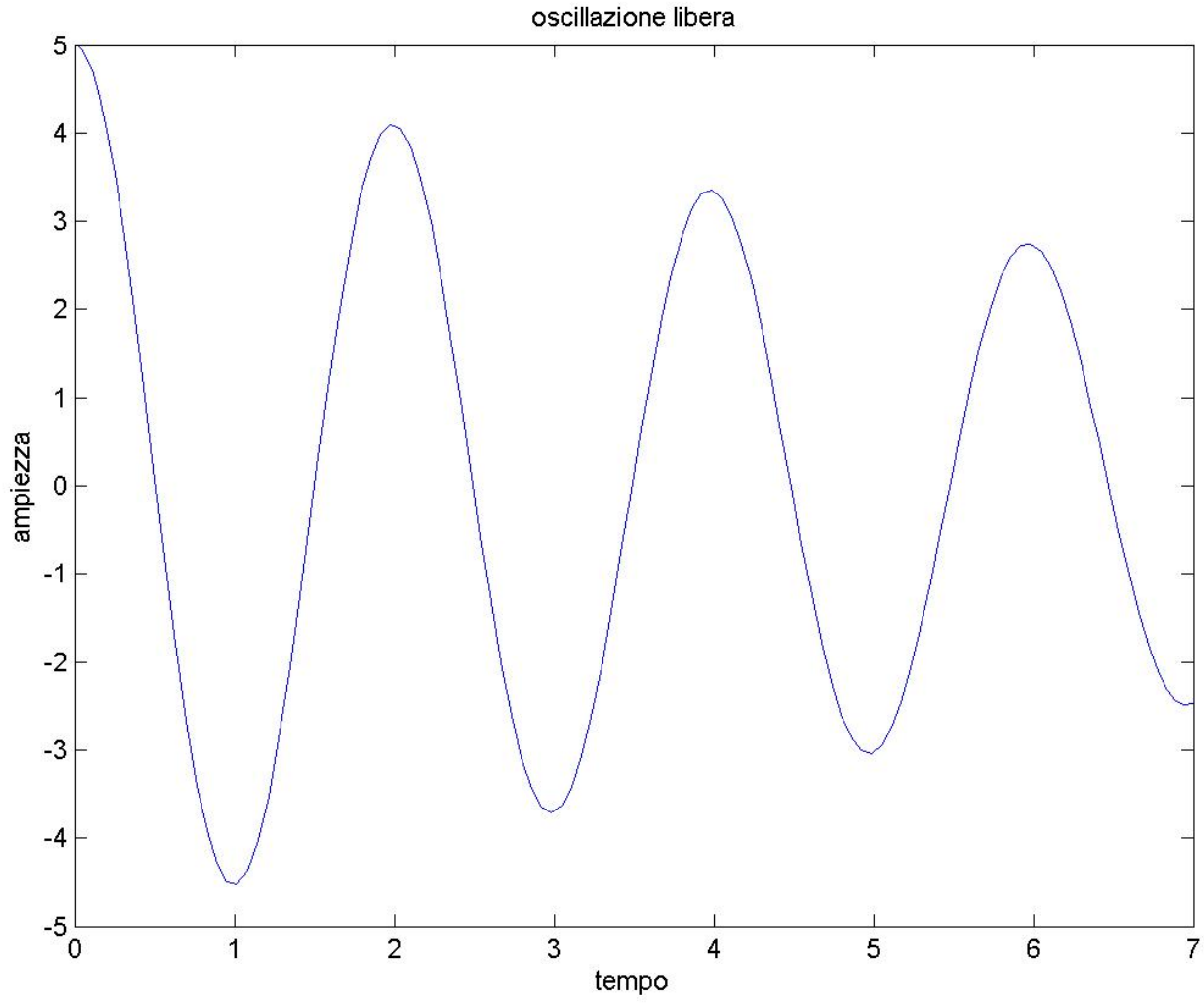


Figura: Oscillazione libera

# Oscillatore - codice

```
% OSCILLATORE LIBERO FORZATO – BATTIMENTO
% parametri del modello
m = 1; k = 100; l = 0; f0 = 1; w = 9;
% tempo di simulazione
tspan = [0,12];
% configurazione iniziale
y0 = [0;0];
% simulazione
options = odeset('Vectorized','on','InitialStep',0.01);
[t,y]=ode45('eqOsc',tspan,y0,options,m,k,l,f0,w);
% Posizione
figure; plot(t,y(:,1));xlabel('tempo');ylabel('ampiezza');
title('battimento');
```

# Oscillatore - codice

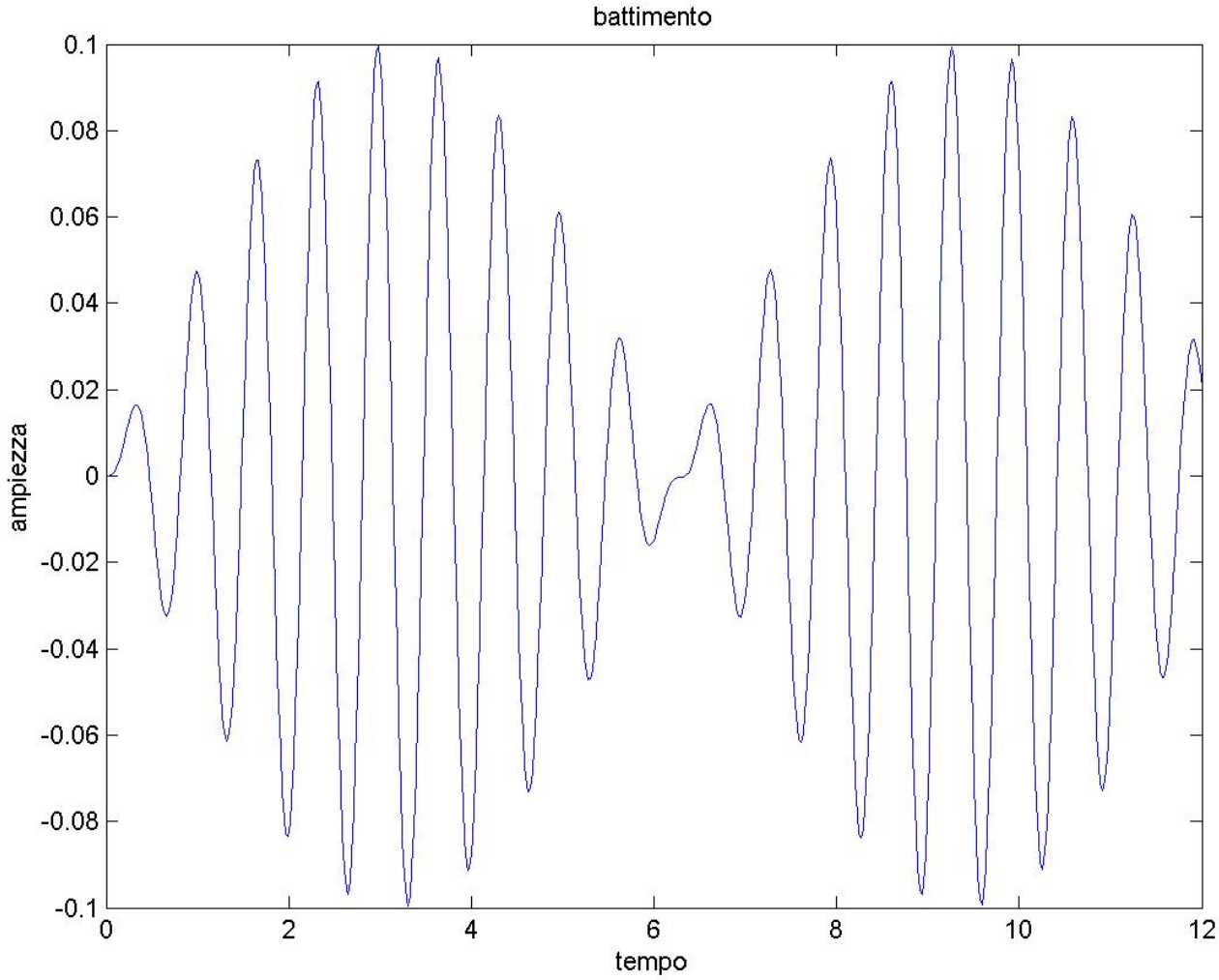


Figura: Battimento

# Oscillatore - codice

```
% OSCILLATORE LIBERO FORZATO – SISTEMA SOTTOSMORZATO
% parametri del modello
m = 1; k = 2; l = 1.0; f0 = 30; w = 20;
% tempo di simulazione
tspan = [0,10];
% configurazione iniziale
y0 = [5;16];
% simulazione
options = odeset('Vectorized','on','InitialStep',0.001);
[t,y]=ode45('eqOsc',tspan,y0,options,m,k,l,f0,w);
% Posizione
figure; plot(t,y(:,1));xlabel('tempo');ylabel('ampiezza');
title('sistema sottosmorzato');
```

# Oscillatore - codice

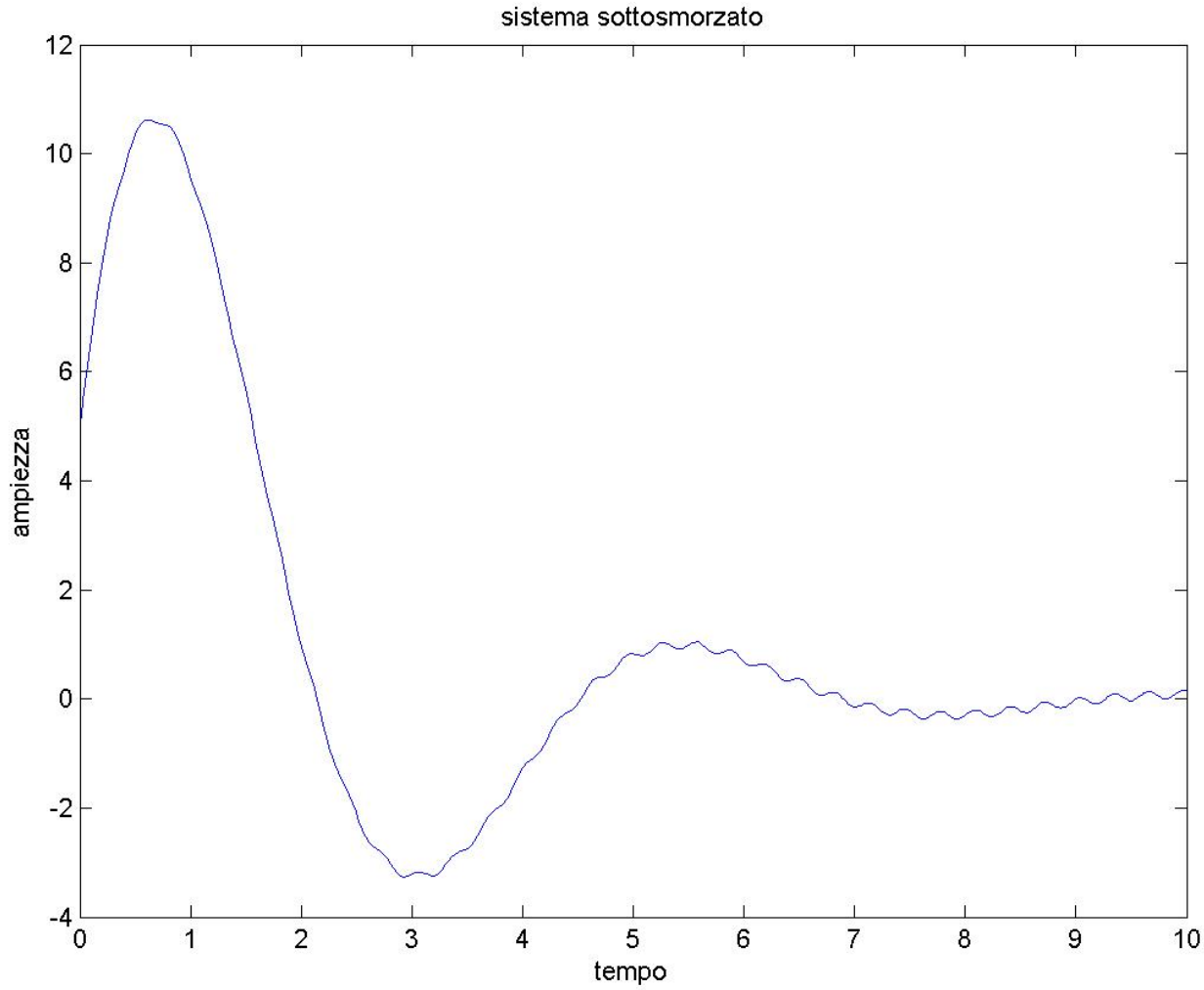


Figura: Sistema sottosmorzato

FINE