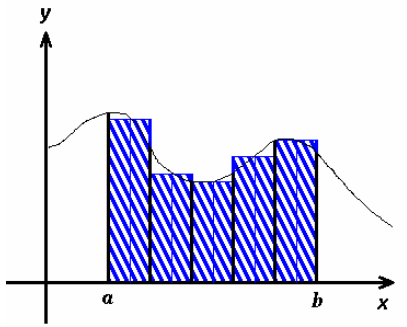
Calcolo numerico e ODE

- Calcolo numerico
  - Integrazione numerica
  - Derivazione numerica
- Equazioni differenziali

Integrazione Numerica



Metodo dei rettangoli

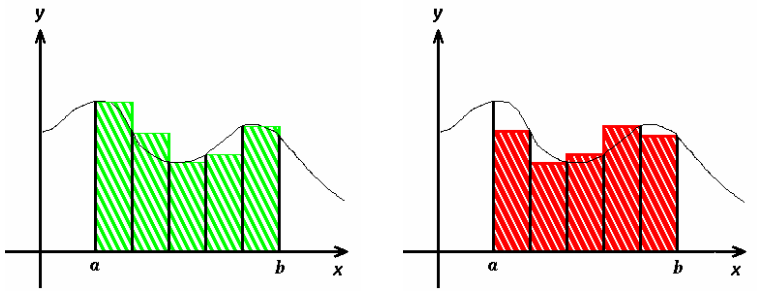
Con altezza relativa al punto medio

Implementazione:

```
function integra
a=0;
b=2*3.14;
numpunti=101;
punti= linspace(a,b,numpunti);

area=0;
for i=2:numpunti
    base=punti(i)- punti(i-1);
    puntomedio=(punti(i)+punti(i-1))/2;
    rettangolo=base*fun(puntomedio);
    area=area+rettangolo;
end
area
```

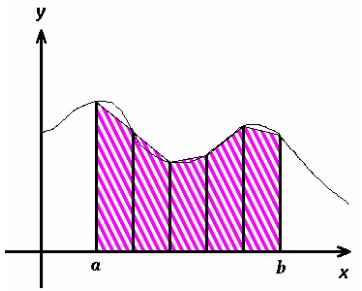
```
function y=fun(x)
y = x + sin(x);
```


Due varianti: altezza su punto sinistro, altezza su punto destro…



Metodo dei trapezi

- Le aree delle sezioni vengono calcolate come aree di trapezi



Implementazione:

```
function integratrap
a=0;
b=2*3.14;
numpunti=101;
punti= linspace(a,b,numpunti);

area=0;
for i=2:numpunti
  base1=fun(punti(i-1));
  base2=fun(punti(i));
  altezzatrap=punti(i)-punti(i-1);
  trapezio=(base1+base2)*altezzatrap*0.5;
  area=area+trapezio;
end
area
```

```
function y=fun(x)
y = x + sin(x);
```

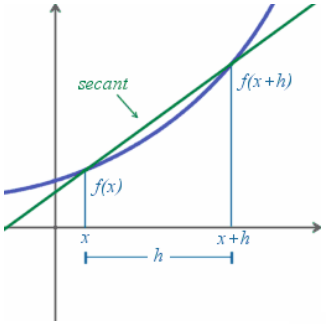

Funzioni predefinite:

`trapz(x,y) :` Applica il metodo dei trapezi per calcolare l'integrale di y(x). y deve contenere i valori della funzione in corrispondenza dei punti contenuti in x.
Es. **`x=0:0.1:pi ; y=sin(x); trapz(x,y)`**

`quad('fun',a,b,tol) :` Applica la regola di Simpson per calcolare l'integrale di fun tra a e b. Tol (tolleranza di errore) è opzionale

`quadl('fun',a,b,tol) :` Applica il metodo di quadratura di Lobatto. Sintassi analoga a quad.

Derivazione Numerica



$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

Differenze in avanti

```
function deriva
a=0;
b=2*3.14;
n=101;
x= linspace(a,b,n);
y=fun(x);
```

```
function y=fun(x)
y = x + sin(x);
```

```
%differenze in avanti
derivata=(y(1:n-1)-y(2:n))./(x(1:n-1)-x(2:n));
plot(x(1:n-1),y(1:n-1),x(1:n-1),derivata,...
  x(1:n-1),cos(x(1:n-1)));
```


Differenze all'indietro

```
function deriva
a=0;
b=2*3.14;
n=101;
x= linspace(a,b,n);
y=fun(x);
```

```
function y=fun(x)
y = x + sin(x);
```

```
%differenze all'indietro
derivata=(y(2:n)-y(1:n-1))./(x(2:n)-x(1:n-1));
plot(x(2:n),y(2:n),x(2:n),derivata,...
  x(2:n),cos(x(2:n)));
```


Differenze centrali

```
function deriva
a=0;
b=2*3.14;
n=101;
x= linspace(a,b,n);
y=fun(x);
```

```
function y=fun(x)
y = x + sin(x);
```

```
%differenze centrali
derivata=(y(3:n)-y(1:n-2))./(x(3:n)-x(1:n-2));
plot(x(2:n-1),y(2:n-1),x(2:n-1),derivata,...
  x(2:n-1),cos(x(2:n-1)));
```


Equazioni differenziali

- Equazioni differenziali del primo ordine: Eulero

$$\frac{dy}{dt} = f(t, y(t))$$

$$\frac{dy}{dt} = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t} \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}$$

$$\frac{y(t+\Delta t) - y(t)}{\Delta t} = f(t, y(t))$$

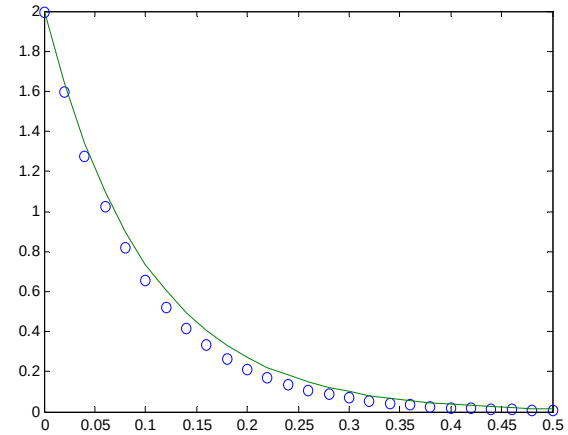
$$y(t+\Delta t) = y(t) + f(t, y(t))\Delta t$$

$$y(t_{k+1}) = y(t_k) + f(t_k, y(t_k))\Delta t$$

```
function eulero
delta=0.02; tfin=0.5; y(1)=2;
k=0;
for t=delta:delta:tfin
    k=k+1;
    y(k+1)=y(k)+delta*ydot(t-delta,y(k));
end

t_true=[0:delta:tfin];
y_true=2*exp(-10*t);

plot(t_true,y,'o',t_true,y_true);
```

```
function out=ydot(t,y)
out=-10*y;
```

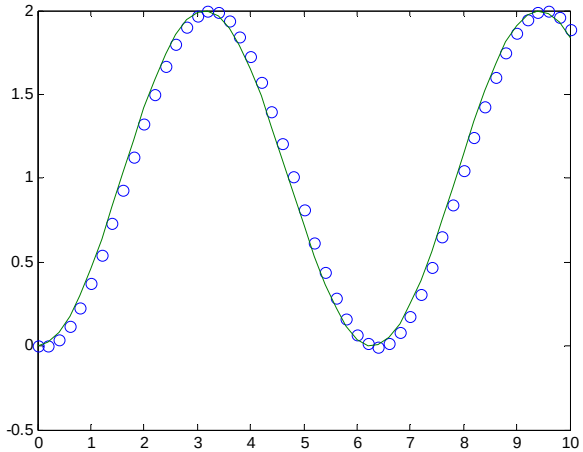

Soluzione di y' = - 10 y , con y(0)=2




```
function eulero2
delta=0.2; tfin=10; y(1)=0;
k=0;
for t=delta:delta:tfin
    k=k+1;
    y(k+1)=y(k)+delta*ydot(t-delta,y(k));
end

t_true=[0:delta:tfin];
y_true=1-cos(t_true);
plot(t_true,y,'o',t_true,y_true);
```

```
function out=ydot(t,y)
out=sin(t);
```


Soluzione di y' = sin(t) , con y(0) = 0



- Metodo di Eulero con correzione

$$\frac{dy}{dt} = f(t, y(t))$$

$$y(t_{k+1}) = y(t_k) + \frac{\Delta t}{2}\left[f(t_k, y(t_k)) + f(t_{k+1}, \hat{y}(t_{k+1}))\right]$$

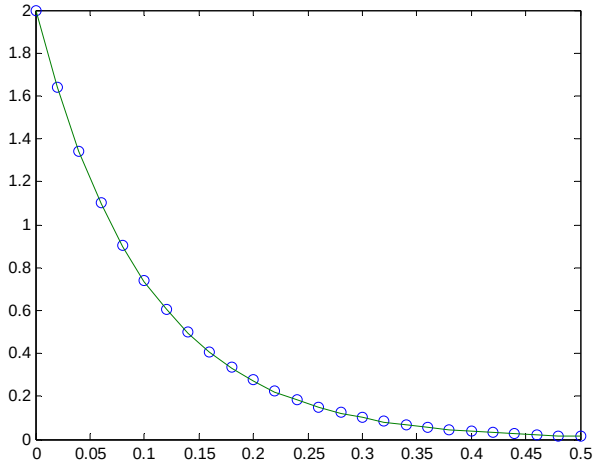
$$\hat{y}(t_{k+1}) = y(t_k) + f(t_k, y(t_k))\Delta t$$

Stima con il metodo di Eulero!


```
function eulerocorr
delta=0.02; tfin=0.5; y(1)=2;
k=0;
for t=delta:delta:tfin
  k=k+1;
  y_stima=y(k)+delta*ydot(t-delta,y(k));
  y(k+1)=y(k)+delta/2*(ydot(t-delta,y(k))...
              +ydot(t,y_stima));
end

t_true=[0:delta:tfin];
y_true=2*exp(-10*t_true);
plot(t_true,y,'o',t_true,y_true);
```

```
function out=ydot(t,y)
out=-10*y;
```

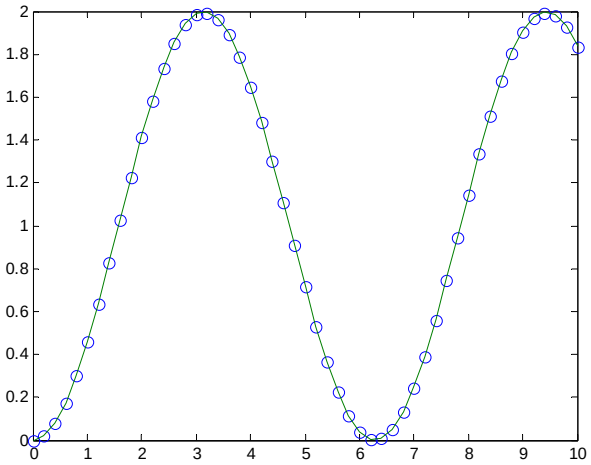

Soluzione di y' = - 10 y , con y(0)=2




```
function eulerocorr2
delta=0.2; tfin=10; y(1)=0;
k=0;
for t=delta:delta:tfin
  k=k+1;
  y_stima=y(k)+delta*ydot(t-delta,y(k));
  y(k+1)=y(k)+delta/2*(ydot(t-delta,y(k))+…
              ydot(t,y_stima));
end
t_true=[0:delta:tfin];
y_true=1-cos(t_true);
plot(t_true,y,'o',t_true,y_true);
```

```
function out=ydot(t,y)
out=sin(t);
```

Soluzione di y' = sin(t) , con y(0) = 0



Runge kutta

- Metodi basati sullo sviluppo in serie di taylor della funzione r(t,y(t))

ODE

`[t,y]=ode23('ydot', tspan, y0)`

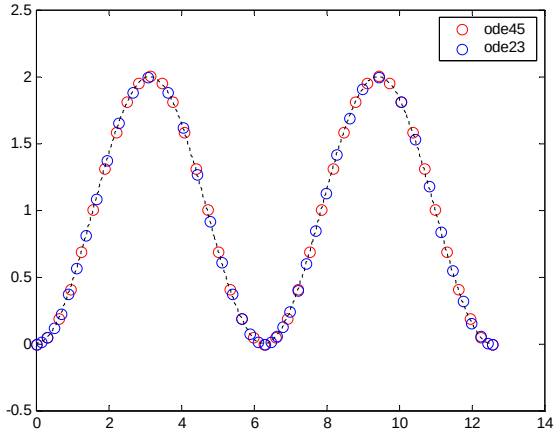
Risolve l'equazione differenziale y'=f(t,y) specificata nel file di funzione ydot, i cui input sono t ed y e il cui output è un vettore colonna che rappresenta dy/dt , cioè f(t,y). Il vettore tspan contiene i valori iniziale e finale della variabile t. y0 rappresenta il valore iniziale y(t0). Esiste anche la ode45.

Uno script…

```
t_true=0:0.01:4*pi ;
y_true=1-cos(t_true);

[t23,y23]=ode23('ydot',[0, 4*pi],0);

[t45,y45]=ode45('ydot',[0, 4*pi],0);

plot(t45,y45,'ro',t23,y23,'bo',...
          t_true,y_true,'k:');
```

```
function out=ydot(t,y)
out=sin(t);
```

Soluzione di y' = sin(t) , con y(0) = 0



Equazioni di secondo grado

$$5y''+7y'+4y = f(t) \rightarrow y'' = \frac{1}{5}f(t) - \frac{4}{5}y - \frac{7}{5}y'$$

$$x_1 = y \qquad x_2 = y'$$

$$\begin{cases} x_1' = x_2 \\ x_2' = \frac{1}{5}f(t) - \frac{4}{5}x_1 - \frac{7}{5}x_2 \end{cases}$$

x e out sono due vettori 2x1 !!!!

```
function out=xdot(t,x)
out= [ x(2);
1/5*(sin(t)-4*x(1)-7*x(2))];
```


Chiamiamo ode45

```
>> [t,x]=ode45('xdot',[0 6], [3 9]);
```

Intervallo di t da 0 a 6

$x_1(0)$

$x_2(0) = x'(0)$

```
>> plot(t,x(:,1))
```