Esempio Metodo di Eulero con effetti distruttivi del floating point

```
> with(plots):
```

Esempio di equazione differenziale

```
> f := (x,y) -> y/(1+x^2) ;
  a := 0 ;
  b := 40 ;
```

$$f := (x, y) \to \frac{y}{1 + x^2}$$

$$a := 0$$

$$b := 40 \qquad \textbf{(1)}$$

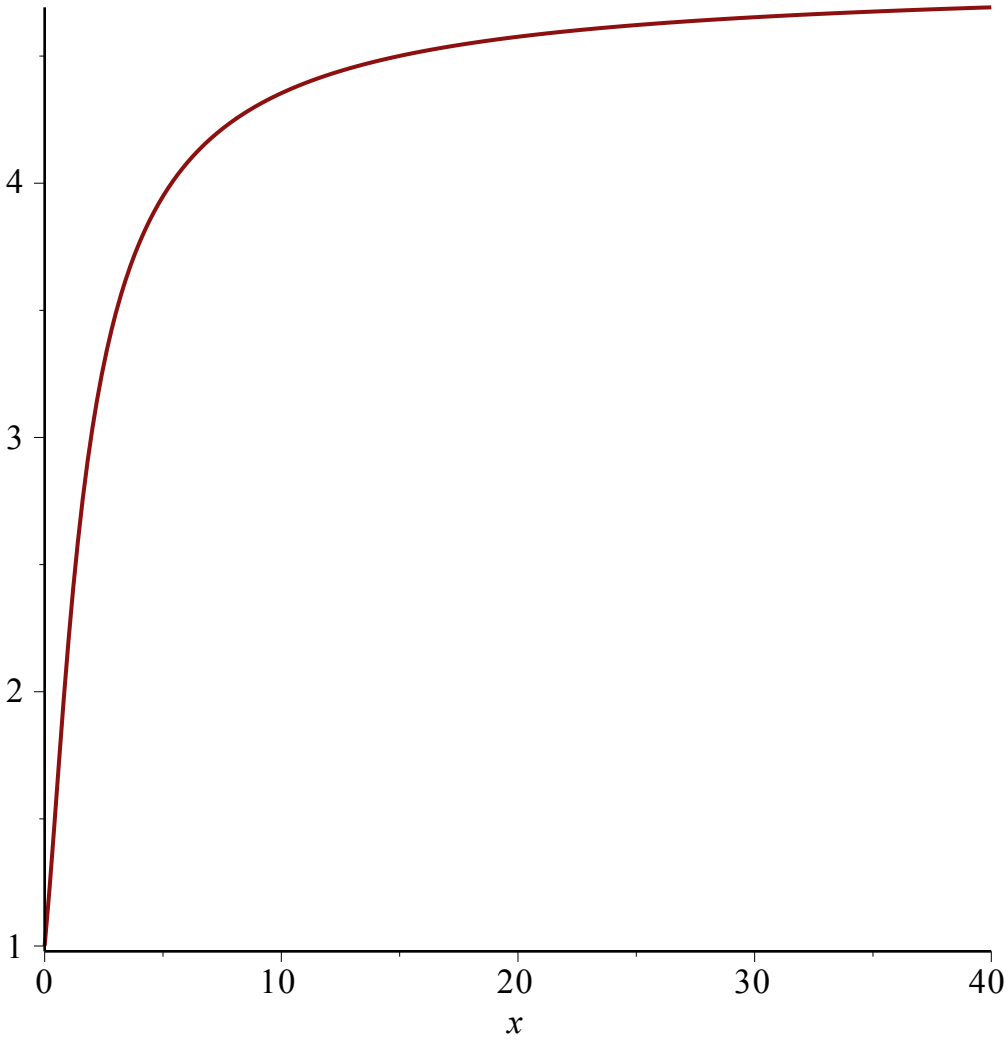
Soluzione esatta dell'ODE

```
> ESATTA := dsolve( {diff(y(x),x) - f(x,y(x)), y(a)=1} ) ;
```

$$ESATTA := y(x) = e^{\arctan(x)} \qquad \textbf{(2)}$$

Plot della soluzione in [a,b]

```
> AA := plot( subs(ESATTA,y(x)), x=a..b ) :
  display(AA) ;
```



Funzione Eulero, calcola la soluzione in [a,b] (Eulero a precisione infinita)

```
> EuleroSolver := proc( n::integer )
    local X, Y, h, k, xk, yk ;
    h := (b-a)/n ;
    X := [a] ;
    Y := [1] ; # dato iniziale
    for k from 1 to n do
      # calcolo yk
      yk := Y[-1] + h*f(X[-1],Y[-1]) ;
      xk := X[-1] + h ;
      # aggiungo alla lista la soluzione
      X := [op(X),xk] ;
      Y := [op(Y),yk] ;
    end ;
    X,Y ;
  end proc ;
```

EuleroSolver := **proc**(*n*::*integer*) **(3)**

local *X*, *Y*, *h*, *k*, *xk*, *yk*;

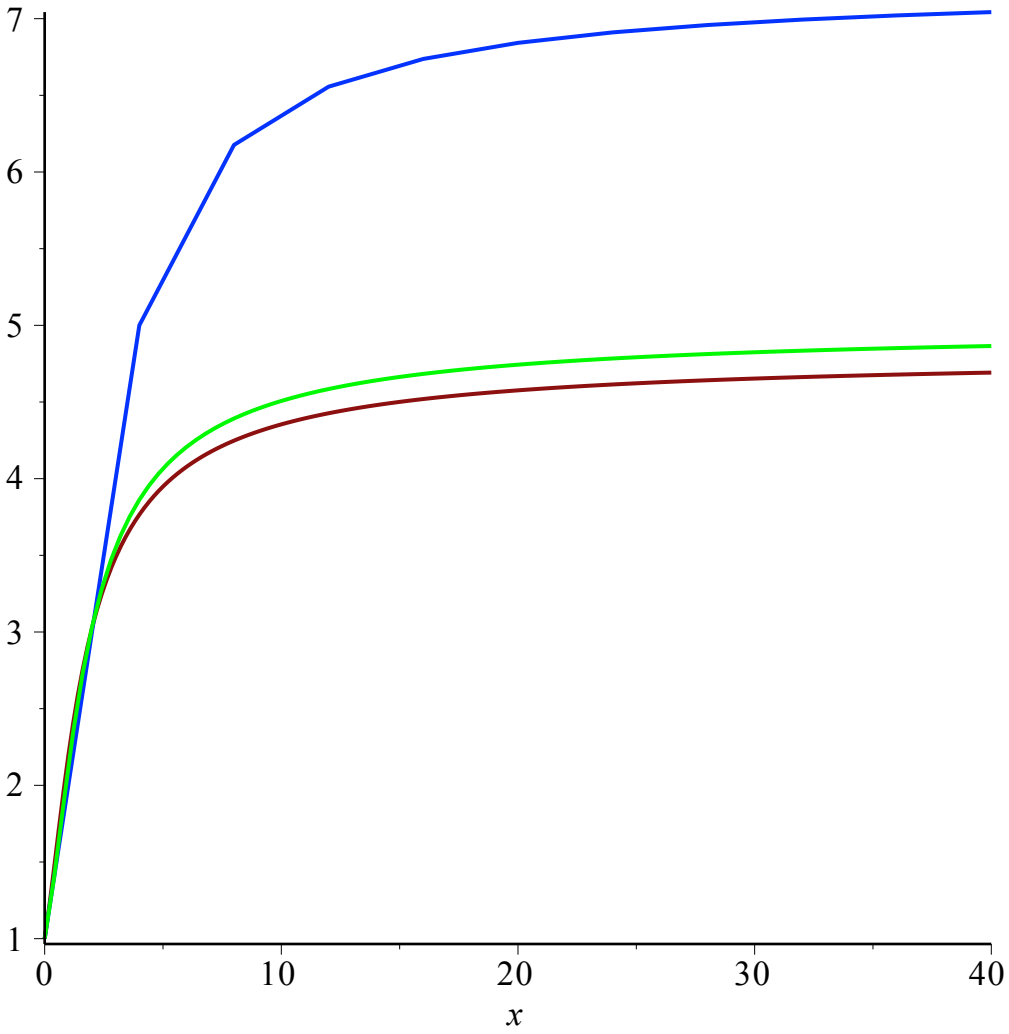
$h := (b - a) / n$;

$X := [a]$;

$Y := [1]$;

```
    for k to n do
        yk := Y[ -1 ] + h*f(X[ -1 ], Y[ -1 ]);
        xk := X[ -1 ] + h;
        X := [op(X), xk];
        Y := [op(Y), yk]
    end do;
    X, Y
end proc
```

```
> X10, Y10   := EuleroSolver(10) :
  X100,Y100 := EuleroSolver(100) :
> BB := plot( [seq([X10[k],Y10[k]],k=1..nops(X10))], color=blue ):
  CC := plot( [seq([X100[k],Y100[k]],k=1..nops(X100))], color=green )
  :
  display(AA,BB,CC) ;
```




```
> roll := rand(0..1); # rumore casuale
```

roll := **proc**( ) **proc**( ) **option** *builtin* = *RandNumberInterface*; **end proc**(6, 2, 1) **end proc** **(4)**

Funzione Eulero, calcola la soluzione in [a,b] (Eulero a precisione FINITA)

```
> EuleroSolverF := proc( n::integer, m::integer )
```

```
    local X, Y, h, k, xk, yk ;
    h := (b-a)/n ;
    X := [evalf(a,m)] ;
    Y := [1] ; # dato iniziale
    for k from 1 to n do
      # calcolo yk
      yk := evalf(Y[-1] + h*f(X[-1],Y[-1])+roll()*0.01,m);
      xk := evalf(X[-1] + h) ;
      # aggiungo alla lista la soluzione
      X := [op(X),xk] ;
      Y := [op(Y),yk] ;
    end ;
    X,Y ;
  end proc ;
```

$$EuleroSolverF := \mathbf{proc}(n::integer, m::integer) \tag{5}$$

local $X, Y, h, k, xk, yk;$

$h := (b - a) / n;$

$X := [evalf(a, m)];$

$Y := [1];$

for k **to** n **do**

$yk := evalf(Y[-1] + h * f(X[-1], Y[-1]) + roll() * 0.01, m);$

$xk := evalf(X[-1] + h);$

$X := [op(X), xk];$

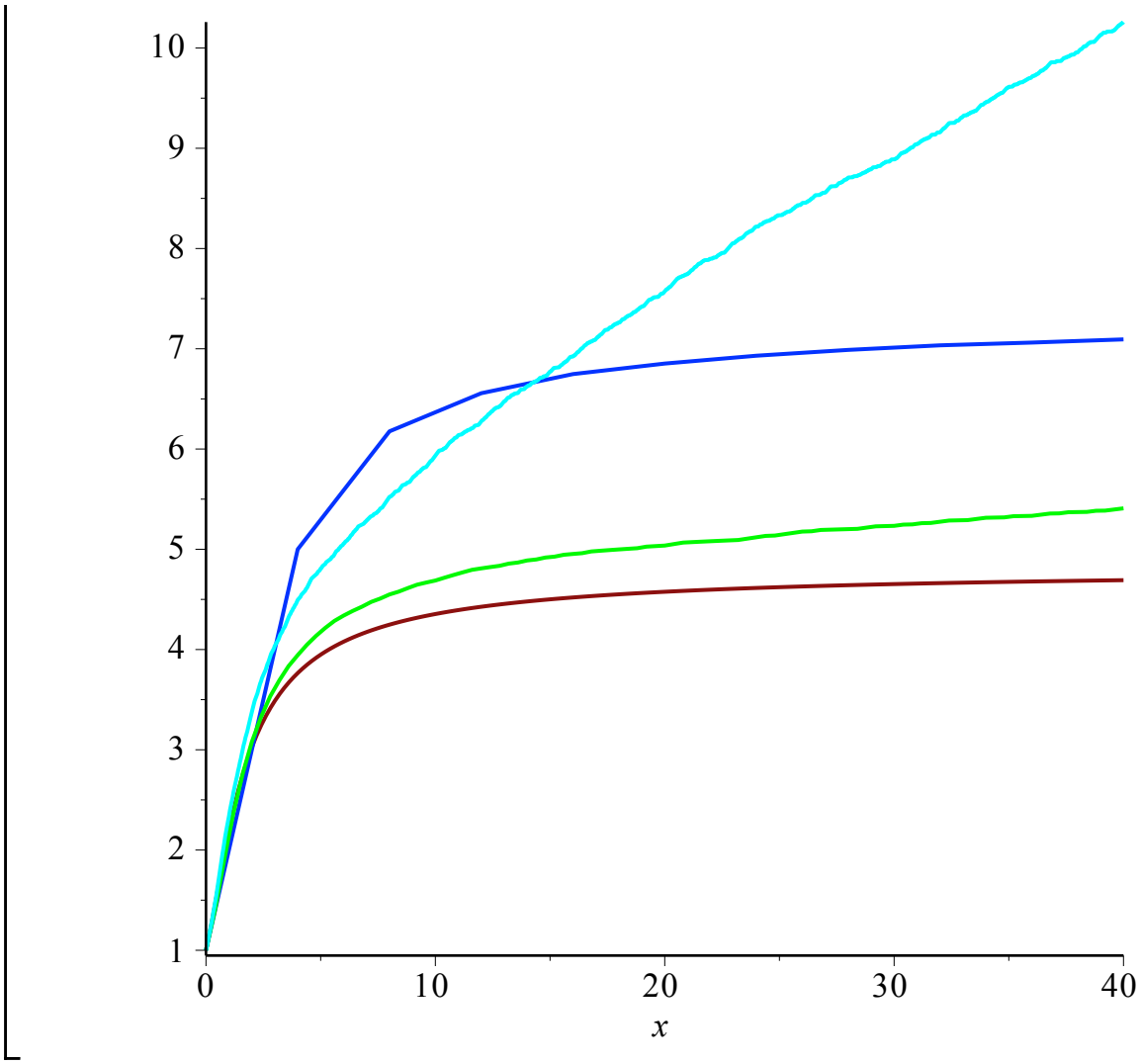
$Y := [op(Y), yk]$

end do;

X, Y

end proc

```
> X10,  Y10   := EuleroSolverF(10,10) :
  X100, Y100  := EuleroSolverF(100,10) :
  X1000,Y1000 := EuleroSolverF(1000,10) :
> BB := plot( [seq([X10[k],Y10[k]],k=1..nops(X10))], color=blue ):
  CC := plot( [seq([X100[k],Y100[k]],k=1..nops(X100))], color=green )
  :
  DD := plot( [seq([X1000[k],Y1000[k]],k=1..nops(X1000))], color=cyan
  ):
  display(AA,BB,CC,DD) ;
```


10
9
8
7
6
5
4
3
2
1
0
10
20
30
40
x