

```

> restart:Digits:=5: with(plots):with(linalg):
  f:=(x,y)->-y; x0:=0.0; y0:=2.0; # f(x,y) in DE dy/dx=-y and
    # initial condition - exact solution is y=2*exp
(-x)
n:=100; xn:=1.0; h:=(xn-x0)/n; # make n=100 now
  xx:=vector(n+1): yy:=vector(n+1): xx[1]:=x0: yy[1]:=y0:
for i from 1 to n do
  x:=xx[i]: y:=yy[i]:
  xx[i+1]:=x+h: yy[i+1]:=y+h*f(x,y):
od:
# now plot computed and exact solns
points:= {seq([xx[i],yy[i]],i=1..n+1) }:
pointplot(points,symbol=asterisk,
  title="Approximate soln computed by Euler method" )
;
display(plot(2*exp(-t),t=0..1),pointplot(points),
  title="Exact and computed solns" )
;restart: Digits:=5:

```

```

> f:=(x,y)->-y; x0:=0.0; y0:=2.0; # f(x,y) in DE dy/dx=-y and
  # initial condition - exact solution is y=2*exp(-x)
(x,y)→-y (1)
0.
2.0

```

```

> # n:=5; h:=0.2; # number of points and stepsize
> n:=10; xn:=1.0; h:=(xn-x0)/n; # alternatively can assign the
right end point
10 (2)
1.0
0.10000

```

```

> x:=x0; y:=y0; # initialize x,y in the begininig of the loop
0. (3)
2.0

```

```

> for i from 1 to n do # hold shift key to prevent execution
  # when hit 'enter'
  k:= f(x,y): # calculate slope at pt (x,y)
  y:= y + h*f(x,y): # calculate next vaue of y
  x:= x + h: # calculate next point x
  print(x, y, 2*exp(-x), 2*exp(-x)-y);# print
approximate and exact solns end error
od: # means 'end of loop'

```

0.10000, 1.8000, 1.8097, 0.0097 (4)

```

0.20000, 1.6200, 1.6375, 0.0175
0.30000, 1.4580, 1.4816, 0.0236
0.40000, 1.3122, 1.3406, 0.0284
0.50000, 1.1810, 1.2131, 0.0321
0.60000, 1.0629, 1.0976, 0.0347
0.70000, 0.95661, 0.99318, 0.03657
0.80000, 0.86095, 0.89866, 0.03771
0.90000, 0.77486, 0.81314, 0.03828
1.0000, 0.69737, 0.73576, 0.03839

```

```

> # we need to save computed values if we want to plot computed
approximation

```

```

> restart:Digits:=5:

```

```

> with(plots):with(linalg):

```

```

> f:=(x,y)->-y; x0:=0.0; y0:=2.0; # f(x,y) in DE dy/dx=-y and
# initial condition - exact solution is y=2*exp(-x)

```

```

f:=(x,y)→-y
x0:=0.
y0:=2.0

```

(5)

```

> n:=10; xn:=1.0; h:=(xn-x0)/n;

```

```

n:=10
xn:=1.0
h:=0.10000

```

(6)

```

> xx:=vector(n+1):yy:=vector(n+1):

```

```

> xx[1]:=x0;yy[1]:=y0;

```

```

xx1:=0.
yy1:=2.0

```

(7)

```

> for i from 1 to n do

```

```

x:=xx[i]: y:=yy[i]:

```

```

xx[i+1]:=x+h: yy[i+1]:=y+h*f(x,y):

```

```

od:

```

```

> xx[5]; yy[5]; # arrays xx and yy now have the x- and y-
coordinates

```

```

# of the computed approximation

```

```

0.40000
1.3122

```

(8)

```

> s_exact:=plot(2*exp(-t), t=0..1):

```

```

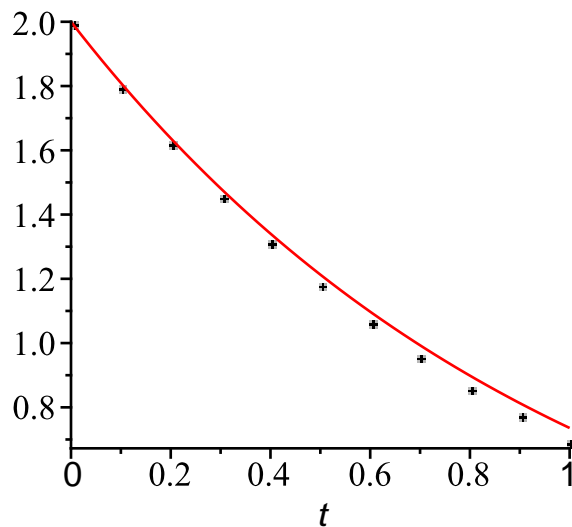
> s_comp:=pointplot({seq([xx[i],yy[i]], i=1..n+1)}):

```

```

> display({s_exact,s_comp});

```



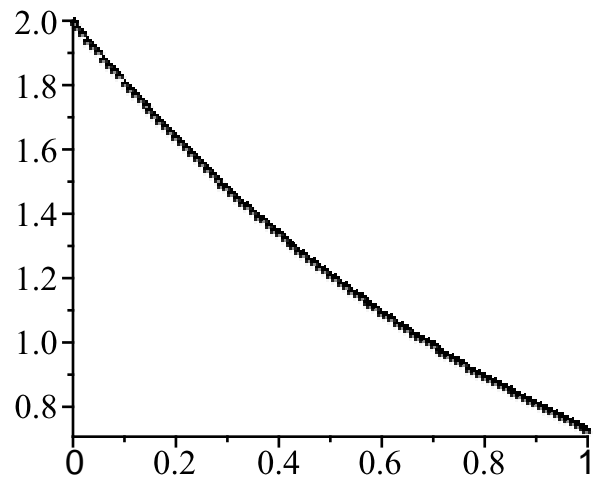
```

> # all the commands can be collected in one group
> restart:Digits:=5: with(plots):with(linalg):
  f:=(x,y)->-y; x0:=0.0; y0:=2.0; # f(x,y) in DE dy/dx=-y and
  # initial condition - exact solution is y=2*exp
  (-x)
n:=100; xn:=1.0; h:=(xn-x0)/n; # make n=100 now
  xx:=vector(n+1): yy:=vector(n+1): xx[1]:=x0: yy[1]:=y0:
for i from 1 to n do
  x:=xx[i]: y:=yy[i]:
  xx[i+1]:=x+h: yy[i+1]:=y+h*f(x,y):
od:
# now plot computed and exact solns
points:= {seq([xx[i],yy[i]],i=1..n+1) }:
pointplot(points,symbol=asterisk,
  title="Approximate soln computed by Euler method")
;
display(plot(2*exp(-t),t=0..1),pointplot(points),
  title="Exact and computed solns" )
;

f:= (x,y) -> -y
x0 := 0.
y0 := 2.0
n := 100
xn := 1.0
h := 0.010000

```

Approximate soln computed by Euler method



Exact and computed solns

