Numerical methods for first order systems of DEs

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

Example 4.1.8 - the slope function is defined by a matrix

```
> A :=Matrix([[0,-1],[1.01,-0.2]]);
  f := (t,x) -> A . x;
```

$$A := \begin{bmatrix} 0 & -1 \\ 1.01 & -0.2 \end{bmatrix}$$

$$f := (t, x) \rightarrow A.x \tag{1}$$

Initial condition

```
> x0 := Vector([0,-1]);
```

$$x0 := \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{2}$$

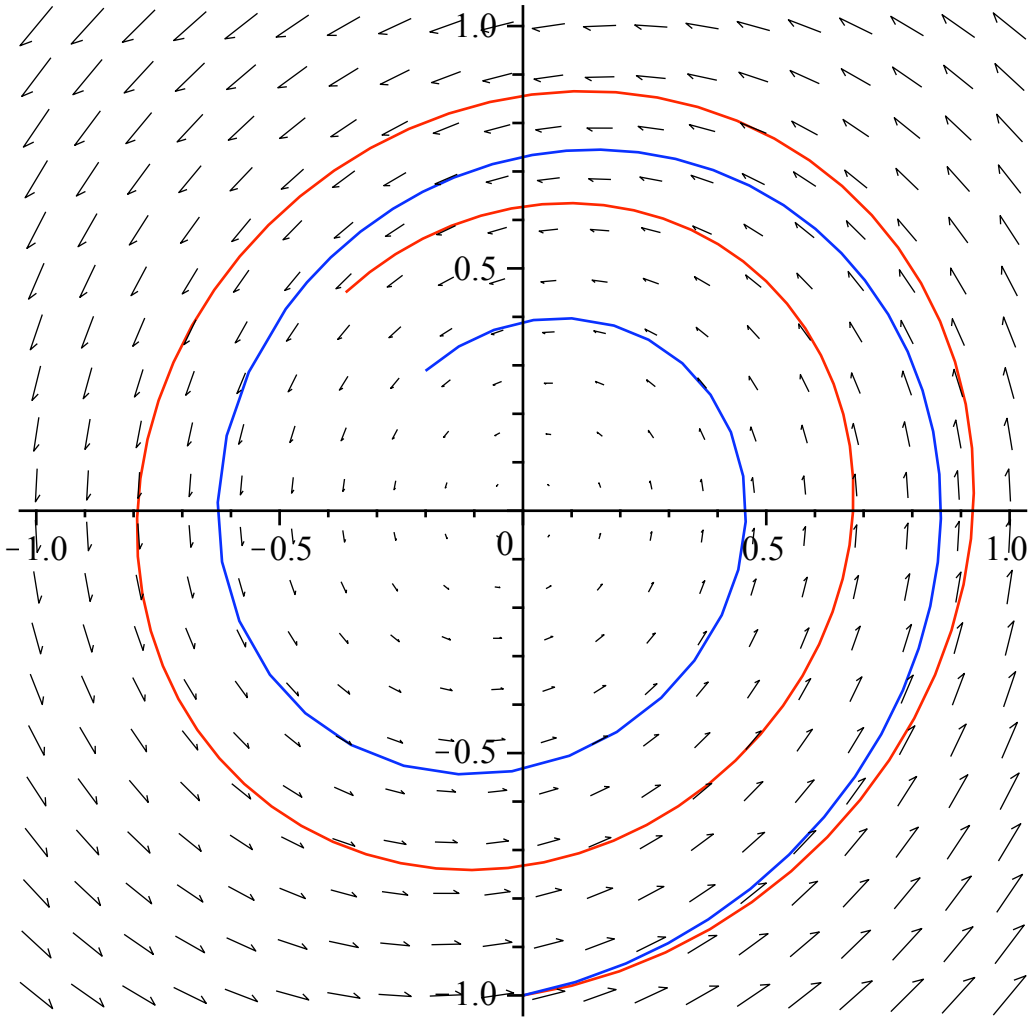
True solution

```
> xtrue:= t-> exp(-t/10)*Vector([sin(t),(sin(t)-10*cos(t))/10]);
```

$$xtrue := t \rightarrow e^{-\frac{1}{10}t}\, Vector\left(\left[\sin(t),\ \frac{1}{10}\sin(t) - \cos(t)\right]\right) \tag{3}$$

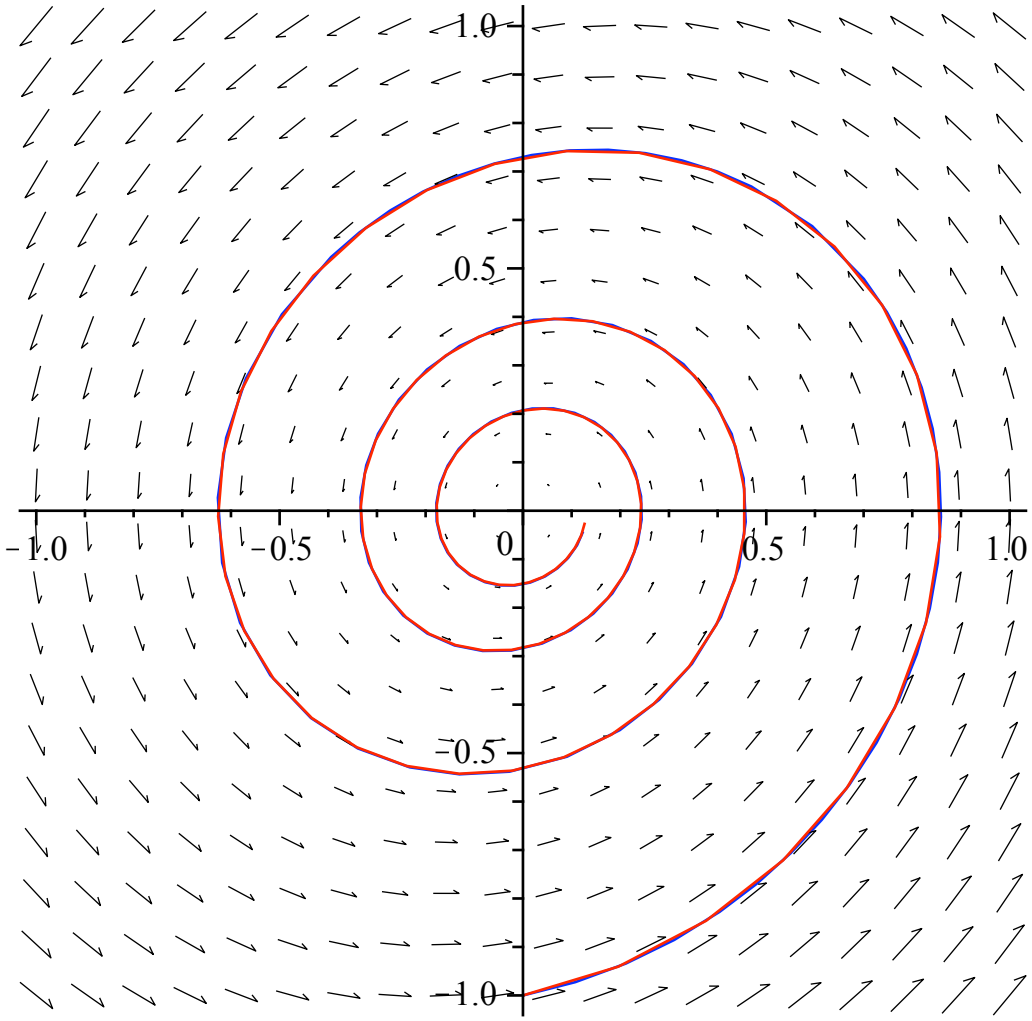
Euler's method

```
> n:=100: t0:=0: tn:=10: h:=(tn-t0)/n:
  tvals:=Vector(n+1): xvals:=Vector(n+1): xvals[1]:=x0: tvals[1]:=
  t0:
> for i from 1 to n do
    x := xvals[i]: t := tvals[i]:
    k := f(t,x):
    xvals[i+1] := x + h*k:
    tvals[i+1] := t:
  end do:
> p1:=plot([seq([xvals[j][1],xvals[j][2]],j=1..n+1)]):
  unassign('x'): unassign('y'): unassign('t'):
  p2:=fieldplot(f(t,Vector([x,y])),x=-1..1,y=-1..1):
  p3:=plot([xtrue(t)[1],xtrue(t)[2],t=t0..tn],color=blue):
  display({p1,p2,p3});
```



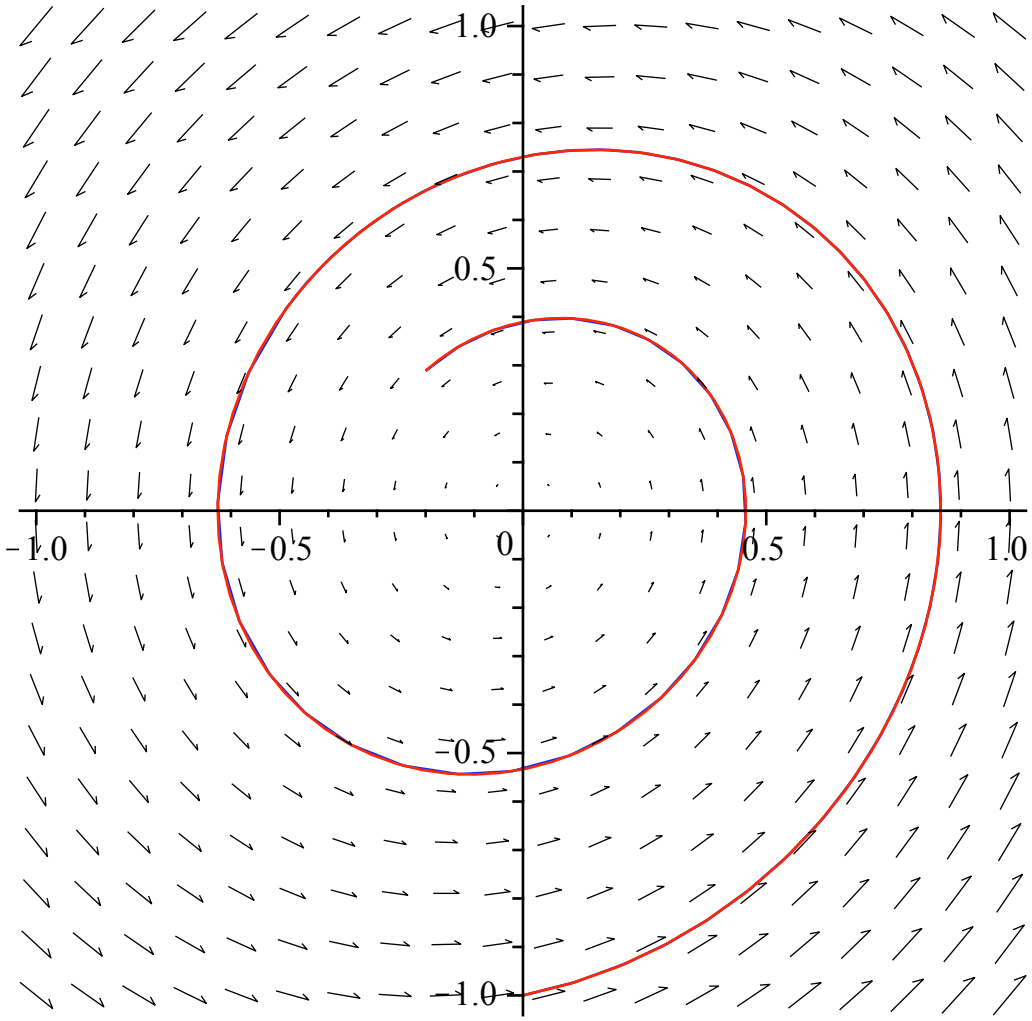
Improved Euler's method

```
> n:=100: t0:=0: tn:=20: h:=(tn-t0)/n:
  tvals:=Vector(n+1): xvals:=Vector(n+1): xvals[1]:=x0: tvals[1]:=
  t0:
> for i from 1 to n do
    x := xvals[i]: t := tvals[i]:
    k1:= f(t,x):          # left hand slope
    k2:= f(t+h,x+h*k1): # approximation to right hand slope
    k:=(k1+k2)/2:         # averaged slope
    xvals[i+1]:= x + h*k:          # improved Euler update
    tvals[i+1]:= t+h:              # increase x
  end do:
> p1:=plot([seq([xvals[j][1],xvals[j][2]],j=1..n+1)]):
  unassign('x'): unassign('y'): unassign('t'):
  p2:=fieldplot(f(t,Vector([x,y])),x=-1..1,y=-1..1):
  p3:=plot([xtrue(t)[1],xtrue(t)[2],t=t0..tn],color=blue):
  display({p1,p2,p3});
```



Runge Kutta method

```
> n:=100: t0:=0: tn:=10: h:=(tn-t0)/n:
  tvals:=Vector(n+1): xvals:=Vector(n+1): xvals[1]:=x0: tvals[1]:=
  t0:
> for i from 1 to n do
    x := xvals[i]: t := tvals[i]:
    k1:= f(t,x):                    # left hand slope
    k2:= f(t+h/2,x+h*k1/2):         # midpoint slope: first approx.
    k3:= f(t+h/2,x+h*k2/2):         # midpoint slope: second approx.
    k4:= f(t+h,x+h*k3):             # right hand slope approx.
    k:=(k1+2*k2+2*k3+k4)/6:         # Simpson's integration rule
    xvals[i+1]:= x + h*k:           # improved Euler update
    tvals[i+1]:= t+h:               # increase x
  end do:
> p1:=plot([seq([xvals[j][1],xvals[j][2]],j=1..n+1)]):
  unassign('x'): unassign('y'): unassign('t'):
  p2:=fieldplot(f(t,Vector([x,y])),x=-1..1,y=-1..1):
  p3:=plot([xtrue(t)[1],xtrue(t)[2],t=t0..tn],color=blue):
  display({p1,p2,p3});
```


1.0
0.5
-1.0
-0.5
0
0.5
1.0
-0.5
-1.0

>