Zum Mma - Kurs, Dr. W. Quapp, FS 2013
Ableitungen und Differentialgleichungen

Defining Derivatives --- from Mma Help System ---

You can define the derivative in *Mathematica* of a function
f of one argument simply by an assignment like f'[x_] = fp[x].

This defines the derivative of $f(x)$ to be $fp(x)$. In this case, you could have used = instead of :=.

```
f'[x_] := fp[x]
```

The rule for f'[x_] is used to evaluate this derivative.

```
D[f[x^2], x]
```

$2\,x\,\mathrm{fp}[x^2]$

Differentiating again gives derivatives of *fp*.

```
D[%, x]
```

$2\,\mathrm{fp}[x^2] + 4\,x^2\,\mathrm{fp}'[x^2]$

This defines a value for the derivative of g at the origin.

```
g'[0] = g0
```

g0

The value for g'[0] is used.

```
D[g[x]^2, x] /. x -> 0
```

2 g0 g[0]

This defines the second derivative of g, with any argument.

```
g''[x_] = gpp[x]
```

gpp[x]

The value defined for the second derivative is used.

```
D[g[x]^2, {x, 2}]
```

$2\,g[x]\,\mathrm{gpp}[x] + 2\,g'[x]^2$

DSolve --- Aufgaben aus Mma Hilfe

```
DSolve[y'[x] + y[x] == a Sin[x], y[x], x]
```

$\left\{\left\{y[x] \to e^{-x}\,C[1] + \frac{1}{2}\,a\,(-\mathrm{Cos}[x] + \mathrm{Sin}[x])\right\}\right\}$

Include a boundary condition:

```
DSolve[{y'[x] + y[x] == a Sin[x], y[0] == 0}, y[x], x]
```

$$\left\{\left\{y[x] \to -\frac{1}{2} a e^{-x} (-1 + e^{x} \mathrm{Cos}[x] - e^{x} \mathrm{Sin}[x])\right\}\right\}$$

Get a "pure function" solution for y :

```
DSolve[{y'[x] + y[x] == a Sin[x], y[0] == 0}, y, x]
```

$$\left\{\left\{y \to \mathrm{Function}\left[\{x\}, -\frac{1}{2} a e^{-x} (-1 + e^{x} \mathrm{Cos}[x] - e^{x} \mathrm{Sin}[x])\right]\right\}\right\}$$

Substitute the solution into any expression :

```
FullSimplify[y''[x] + y[x]^2 /. %]
```

$$\left\{\frac{1}{4} a e^{-2x} \left(a - 2 e^{x} (-1 + a \mathrm{Cos}[x] - a \mathrm{Sin}[x]) + e^{2x} (a + 2 \mathrm{Cos}[x] - 2 \mathrm{Sin}[x] - a \mathrm{Sin}[2x])\right)\right\}$$

Diff.equ. of second order : no boundary condition gives two generated parameters :

```
DSolve[y''[x] + 4 y[x] == 0, y, x]
```

{{y → Function[{x}, C[1] Cos[2 x] + C[2] Sin[2 x]]}}

One boundary condition :

```
DSolve[{y''[x] + 4 y[x] == 0, y[0] == 1}, y, x]
```

{{y → Function[{x}, Cos[2 x] + C[2] Sin[2 x]]}}

Two boundary conditions (one value of the function, one of the derivative)

```
DSolve[{y''[x] + 4 y[x] == 0, y[0] == 1, y'[0] == 4}, y, x]
```

{{y → Function[{x}, Cos[2 x] + 2 Sin[2 x]]}}

or (two values of the function)

```
DSolve[{y''[x] + 4 y[x] == 0, y[0] == 1, y[1] == 2}, y, x]
```

{{y → Function[{x}, Cos[2 x] - Cot[2] Sin[2 x] + 2 Csc[2] Sin[2 x]]}}

Use differently named constants :

```
DSolve[y''[x] == y[x], y[x], x, GeneratedParameters → d]
```

$$\{\{y[x] \to e^{x} d[1] + e^{-x} d[2]\}\}$$

Use subscripted constants :

```
DSolve[y''[x] == y[x], y[x], x, GeneratedParameters → (Subscript[c, #] &)]
```

$$\{\{y[x] \to e^{x} c_1 + e^{-x} c_2\}\}$$

Solve a logistic equation :

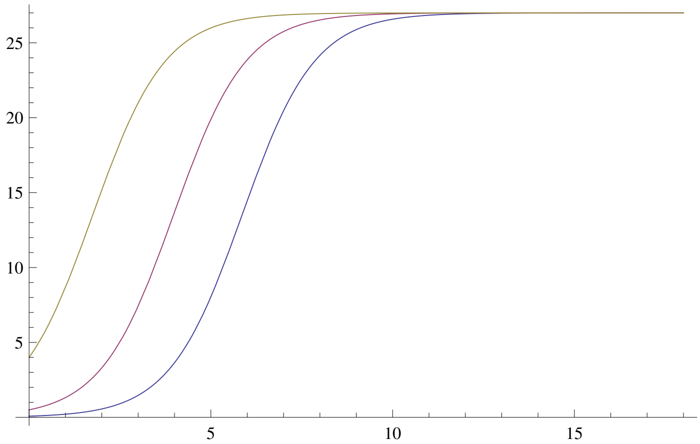
```
DSolve[{y'[x] == y[x] (1 - y[x] / 27), y[0] == a}, y, x]
```

Solve::ifun : Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information. ≫

$$\left\{\left\{y \to \text{Function}\left[\{x\}, \frac{27\, a\, e^{x}}{27 - a + a\, e^{x}}\right]\right\}\right\}$$

Plot the solution for different initial values :

```
Plot[Evaluate[y[x] /. % /. {{a → 1 / 13}, {a → 1 / 2}, {a → 4}}], {x, 0, 18}]
```

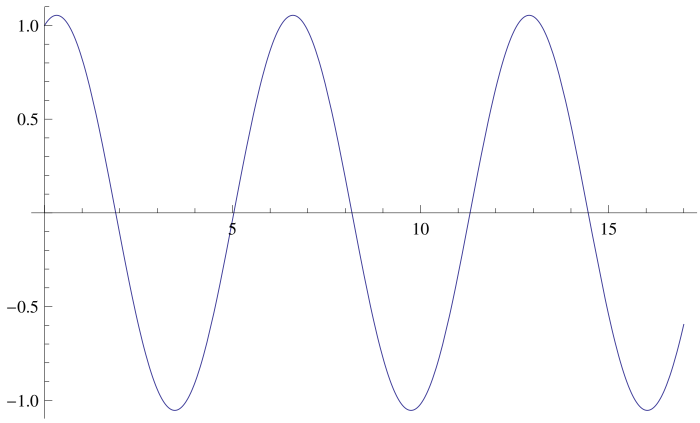


Solve a linear pendulum equation :

```
DSolve[{y''[x] + y[x] == 0, y[0] == 1, y'[0] == 1 / 3}, y , x]
```

$$\left\{\left\{y \to \text{Function}\left[\{x\}, \frac{1}{3}(3\,\text{Cos}[x] + \text{Sin}[x])\right]\right\}\right\}$$

```
Plot[y[x] /. %, {x, 0, 17}]
```

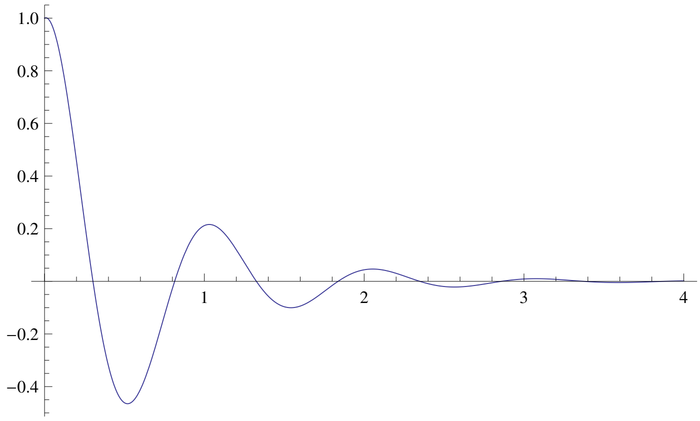


Displacement of a linear, damped pendulum :

```
DSolve[{y''[x] + 3 y'[x] + 40 y[x] == 0, y[0] == 1, y'[0] == 1 / 3}, y, x]
```

$$\left\{\left\{y \to \text{Function}\left[\{x\}, \frac{1}{453} e^{-3x/2} \left(453\,\text{Cos}\left[\frac{\sqrt{151}\, x}{2}\right] + 11\sqrt{151}\,\text{Sin}\left[\frac{\sqrt{151}\, x}{2}\right]\right)\right]\right\}\right\}$$

```
Plot[y[x] /. %, {x, 0, 4}, PlotRange → All]
```



Find a power series solution when the exact solution is known :

```
DSolve[{y'[x] + Exp[x] y[x] == 1, y[0] == 3}, y , x]
```

$$\left\{\left\{y \to \text{Function}\left[\{x\},\ e^{-e^x}\ (3\ e - \text{ExpIntegralEi}[1] + \text{ExpIntegralEi}[e^x])\right]\right\}\right\}$$

```
Series[y[x] /. %[[1]], {x, 0, 7}]
```

$$3 - 2x - \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{6} - \frac{x^5}{120} - \frac{11x^6}{360} - \frac{x^7}{105} + O[x]^8$$

Recover a function from its gradient vector :

```
DSolve[{D[f[x, y], x] == 2 x y^3 + y Cos[x y],
       D[f[x, y], y] == 3 x^2 y^2 + x Cos[x y]}, f[x, y], {x, y}]
```

$$\{\{f[x, y] \to x^2 y^3 + C[1] + \text{Sin}[x\,y]\}\}$$

Solutions satisfy the differential equation and boundary conditions :

```
DSolve[{y''[x] - y[x] == 0, y[0] == 1, y'[0] == 4}, y, x]
```

$$\left\{\left\{y \to \text{Function}\left[\{x\},\ \frac{1}{2}\, e^{-x} \left(-3 + 5\, e^{2x}\right)\right]\right\}\right\}$$

```
Simplify[{y''[x] - y[x] == 0, y[0] == 1, y'[0] == 4} /. %]
```

{{True, True, True}}

Differential equation correspoding to Integrate

```
DSolve[y'[x] == Exp[-x^2], y, x]
```

$$\left\{\left\{y \to \text{Function}\left[\{x\},\ C[1] + \frac{1}{2}\sqrt{\pi}\ \text{Erf}[x]\right]\right\}\right\}$$

```
Integrate[Exp[-x^2], x]
```

$$\frac{1}{2}\sqrt{\pi}\ \text{Erf}[x]$$

Use NDSolve to find a numerical solution :

```
exactsol = DSolve[{y''[x] + y[x] == 0, y[0] == 1, y'[0] == 0}, y, x]
```

{{y → Function[{x}, Cos[x]]}}

```
Table[y[x] /. exactsol[[1]], {x, -2., 2}]
```

```
numsol = NDSolve[{y''[x] + y[x] == 0, y[0] == 1, y'[0] == 0}, y, {x, -2, 2}]
```

{{y → InterpolatingFunction[{{-2., 2.}}, <>]}}

```
Table[y[x] /. numsol[[1]], {x, -2., 2}]
```

{-0.416147, 0.540302, 1., 0.540302, -0.416147}

Compute an impulse response

```
DSolve[y'''[x] - 5 y''[x] + 9 y'[x] - 5 y[x] ==
     DiracDelta''[x] + 2 DiracDelta'[x] + DiracDelta[x] &&
   y[-1] == 0 && y'[-1] == 0 && y''[-1] == 0, y[x], x] // FullSimplify
```

{{y[x] → $-e^x$ HeavisideTheta[x] (-2 + e^x (Cos[x] - 7 Sin[x]))}}

The same computation using InverseLaplaceTransform

$$\text{InverseLaplaceTransform}\left[\frac{s\hat{}2 + 2\,s + 1}{s\hat{}3 - 5\,s\hat{}2 + 9\,s - 5}, s, x\right] \text{ // FullSimplify}$$

e^x (2 - e^x (Cos[x] - 7 Sin[x]))

Results may contain symbolic integrals :

```
DSolve[y'[x] == f[x], y, x]
```

$$\left\{\left\{y \to \text{Function}\left[\{x\}, C[1] + \int_1^x f[K[1]]\, dK[1]\right]\right\}\right\}$$

Inverse functions may be required to find the solution :

```
DSolve[{y'[x]^2 == (1 - y[x]^2) (1 - (1 / 2) y[x]^2), y[0] == 0}, y, x]
```

Solve::ifun : Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information. ≫

$$\left\{\left\{y \to \text{Function}\left[\{x\}, -\text{JacobiSN}\left[x, \frac{1}{2}\right]\right]\right\}, \left\{y \to \text{Function}\left[\{x\}, \text{JacobiSN}\left[x, \frac{1}{2}\right]\right]\right\}\right\}$$

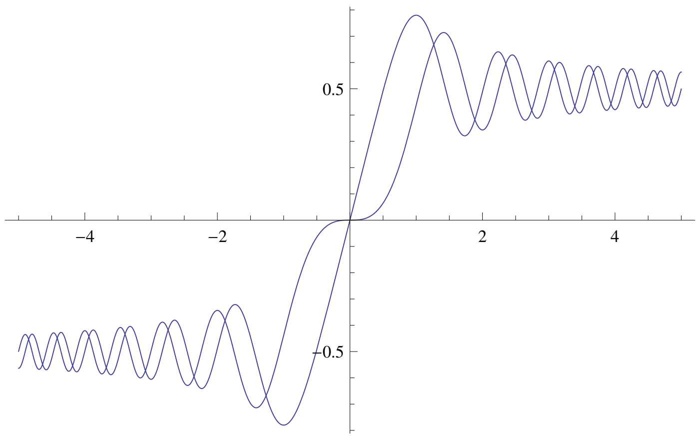
Generate a Cornu spiral :

```
DSolve[{x'[s] == Cos[t[s]], y'[s] == Sin[t[s]],
   t'[s] == s, x[0] == 0, y[0] == 0, t[0] == 0}, {x, y, t}, s]
```

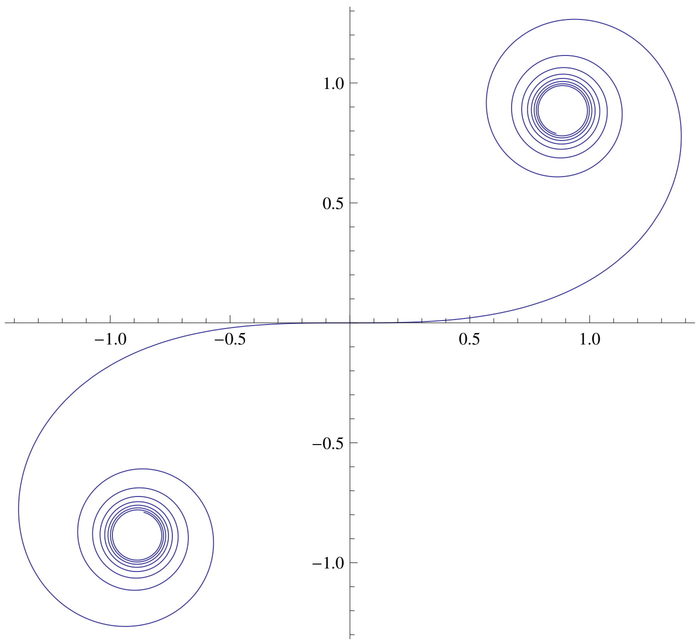
$$\left\{\left\{t \to \text{Function}\left[\{s\}, \frac{s^2}{2}\right], x \to \text{Function}\left[\{s\}, \sqrt{\pi}\ \text{FresnelC}\left[\frac{s}{\sqrt{\pi}}\right]\right],\right.\right.$$
$$\left.\left. y \to \text{Function}\left[\{s\}, \sqrt{\pi}\ \text{FresnelS}\left[\frac{s}{\sqrt{\pi}}\right]\right]\right\}\right\}$$

Die Loesung enthaelt die Fresnel Funktionen

```
fc = Plot[FresnelC[x], {x, -5, 5}];
fs = Plot[FresnelS[x], {x, -5, 5}];
Show[fc, fs]
```




```
ParametricPlot[Evaluate[{x[s], y[s]} /. %], {s, -10, 10}]
```




```
(* ################################################################ *)

(*  Weitere Beispiele: DGl mit gewoehnlicher Funktionsdefinition *)

lsg = DSolve[y'[x] == -Cos[x] * y[x], y[x], x]
```

$\{\{y[x] \to e^{-\text{Sin}[x]}\, C[1]\}\}$

```
(* Versionen der Behandlung der Ausgabe *)
lsg[[1]]
lsg[[1, 1]]
lsg[[1, 1, 2]]
```

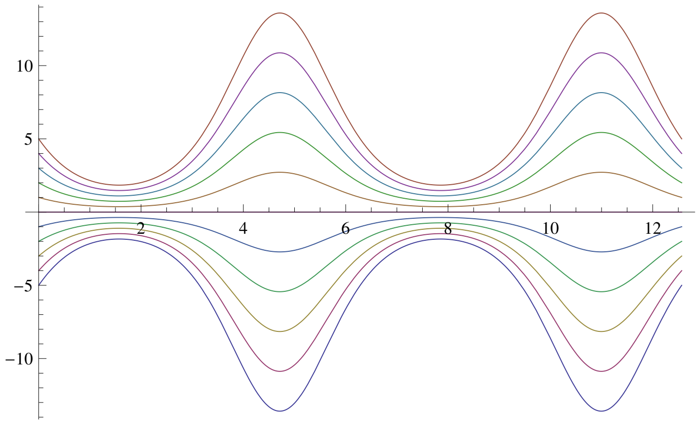
{y[x] → $e^{-Sin[x]}$ C[1]}

y[x] → $e^{-Sin[x]}$ C[1]

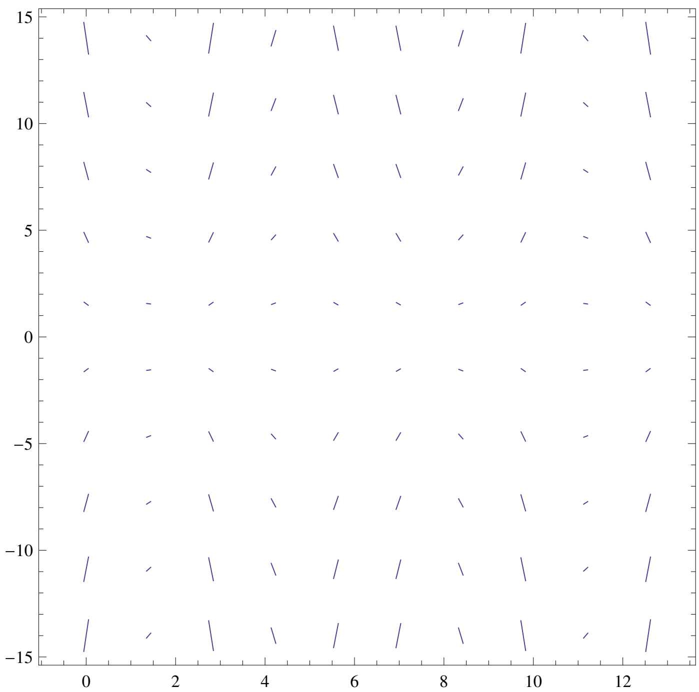
$e^{-Sin[x]}$ C[1]

```
zumbild = Table[lsg[[1, 1, 2]] /. C[1] → j, {j, -5, 5}];
```

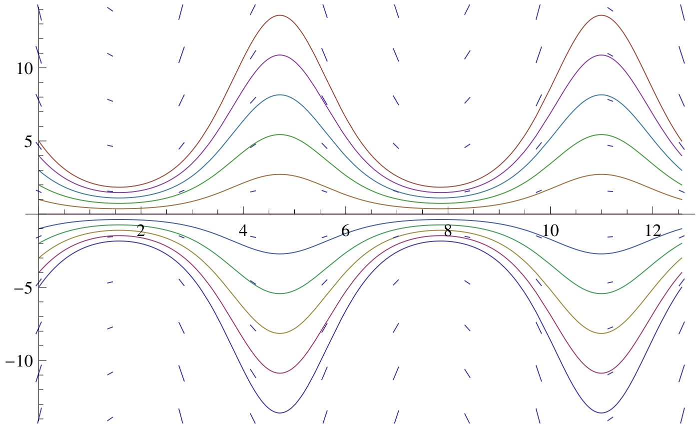
```
p1 = Plot[Evaluate[zumbild], {x, 0, 4 Pi}]
```




```
(* Veranschaulichung des Vektorfeldes der Differentialgleichung *)
p2 = VectorPlot[{1, -Cos[x] * y}, {x, 0, 4 Pi}, {y, -14, 14},
  VectorPoints → 10, VectorScale → {0.05}, VectorStyle → Arrowheads[0]]
```

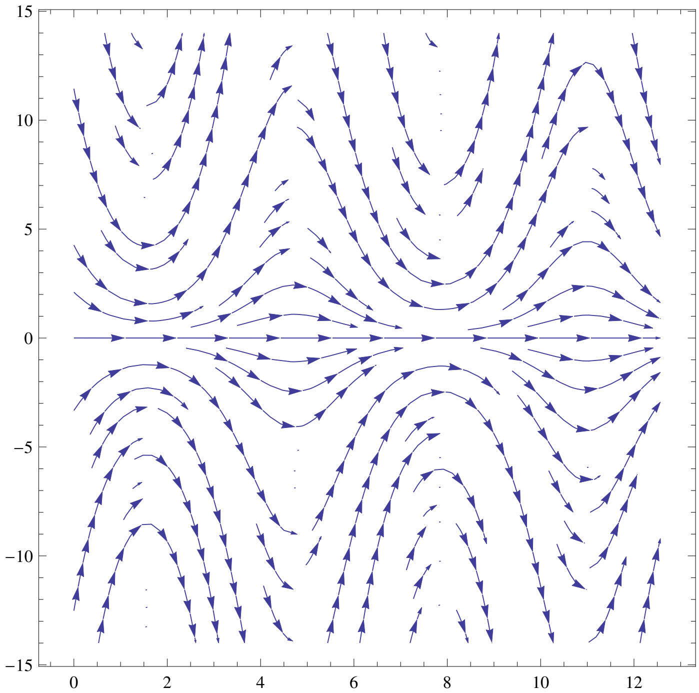


```
Show[p1, p2]
```



Bilder in einem Mma - Befehl kombiniert :

```
StreamPlot[{1, -Cos[x] * y}, {x, 0, 4 Pi}, {y, -14, 14}]
```



Welche Darstellung "besser" ist,
haengt sicher auch vom persoenlichen Geschmack ab

```
(*  Es geht nicht: deshalb besser mit "pure function" arbeiten *)
y'[x] /. lsg
```

{y′[x]}

```
lsg2 = DSolve[y'[x] == -Cos[x] * y[x], y, x]
```

$\{\{y \to \text{Function}[\{x\}, e^{-\text{Sin}[x]}\, C[1]]\}\}$

```
y'[x] /. lsg2
```

$\{-e^{-\text{Sin}[x]}\, C[1]\, \text{Cos}[x]\}$

```
(*  z.B. belibige Werte einsetzen  *)
y[t - 2] /. lsg2
```

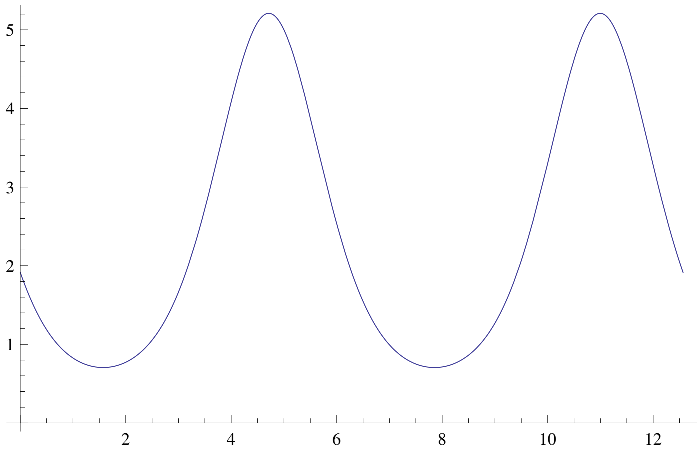
$\{e^{\text{Sin}[2-t]}\ C[1]\}$

Da volle Familie von Kurven die Ebene ueberdeckt,
kann jeder AW kann getroffen werden z.B.

```
(* Hinweis zur Fehlerbehandlung = >>  Unset[y[5]]  *)

lsg3 = DSolve[{y'[x] == -Cos[x] * y[x], y[5] == 5}, y, x]
```

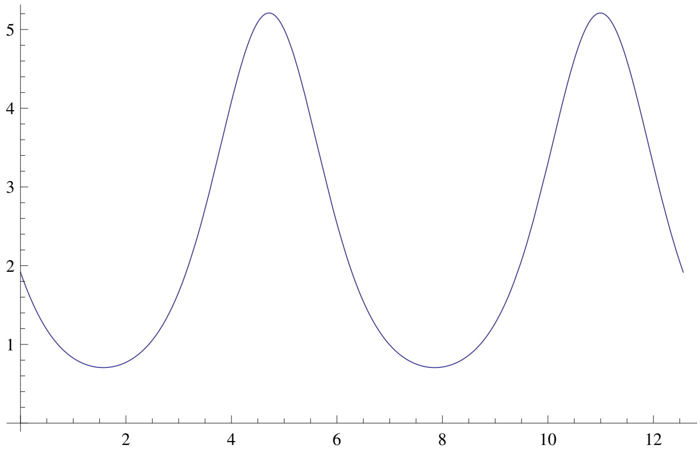
$\{\{y \to \text{Function}[\{x\},\ 5\ e^{\text{Sin}[5]-\text{Sin}[x]}]\}\}$

```
Plot[y[x] /. lsg3 , {x, 0, 4 Pi}]
```



Plot geht mit den {{ }} weil es automatisch Flatten anwendet .
Ohne AW in der DGl kann man auch eine entsprechende Kurve einzeln malen ,
wenn man C[1] richtig schaetzt ...,
oder wieder eine liste von Werteb einsetzen

```
Plot[y[x] /. lsg2 /. c[1] → 2 , {x, 0, 4 Pi}]
```



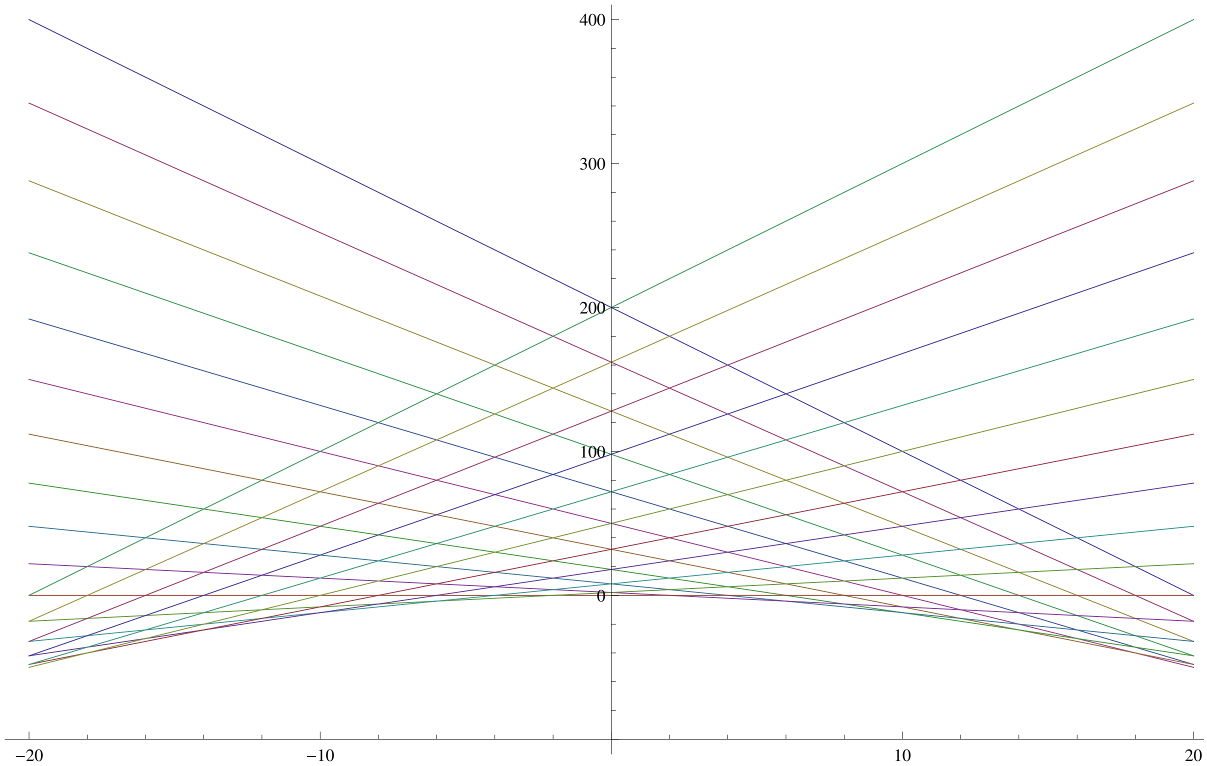

```
(* BSP: Mma findet Einhuellende bei einer nichtlinearen DGl nicht  *)

eqn = y'[x]^2 + 1 / 2 * x * y'[x] - y[x] / 2 == 0;
```

```
lsg4 = DSolve[eqn, y, x]
```

$\{\{y \to \text{Function}[\{x\}, x\,C[1] + 2\,C[1]^2]\}\}$

```
druck = Table[y[x] /. lsg4 /. C[1] → j, {j, -10, 10}];
```

```
Plot[druck, {x, -20, 20}, AxesOrigin → {0, -100}]
```



Hier raten, dass die Enveloppe eine Parabel
f ist (First[*expr*] gives the first element in *expr*,
siehe auch: Part, Last, Rest, Take, Select)

```
f = a2 x^2 + a1 x + a0;
sys = First[eqn] /. {y[x] → f, y'[x] → D[f, x]}
```

$$\frac{1}{2} x\,(a1 + 2\,a2\,x) + (a1 + 2\,a2\,x)^2 + \frac{1}{2}\left(-a0 - a1\,x - a2\,x^2\right)$$

```
sol = Solve[CoefficientList[sys, x] == 0 && a2 ≠ 0]
```

$$\left\{\left\{a0 \to 0,\ a2 \to -\frac{1}{8},\ a1 \to 0\right\}\right\}$$

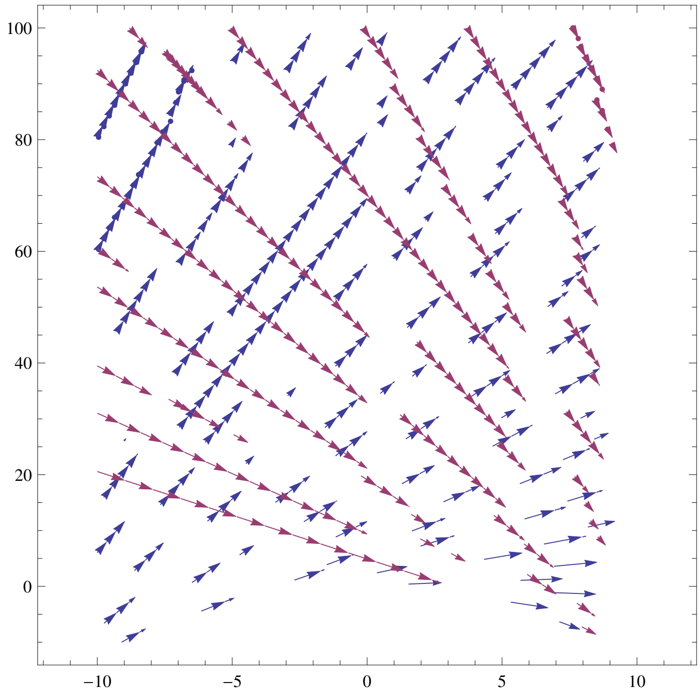
Diese spezielle Loesung ist also y[x] =

```
f /. sol
```

$$\left\{-\frac{x^2}{8}\right\}$$

StreamPlot ist hier etwas ueberfordert,
obwohl man die Loesungen auch sehen kann, wenn man sie schon kennt :

```
StreamPlot[{{1, 1 / 4 (-x + Sqrt[x^2 + 8 y])}, {1, 1 / 4 (-x - Sqrt[x^2 + 8 y])}} ,
 {x, -10, 10}, {y, -10, 100}]
```



Das Aufloesen der DGl "mit Hand" bringt auch keine bessere Einsicht, und auch keine Enveloppe :

```
lsgauf = DSolve[y'[x] == 1 / 4 (-x - Sqrt[x^2 + 8 y[x]]), y, x]
```

$$\left\{\left\{y \to \text{Function}\left[\{x\}, \frac{1}{64}\left(8 + e^{2\,C[1]} + 16\,x - 4\,\sqrt{2}\,\sqrt{e^{2\,C[1]} + 2\,e^{2\,C[1]}\,x + e^{2\,C[1]}\,x^2}\right)\right]\right\},\right.$$
$$\left.\left\{y \to \text{Function}\left[\{x\}, \frac{1}{64}\left(8 + e^{2\,C[1]} + 16\,x + 4\,\sqrt{2}\,\sqrt{e^{2\,C[1]} + 2\,e^{2\,C[1]}\,x + e^{2\,C[1]}\,x^2}\right)\right]\right\}\right\}$$

```
lsgaufaw = DSolve[{y'[x] == 1 / 4 (-x + Sqrt[x^2 + 8 y[x]]), y[0] == 100}, y, x]
```

Solve::ifun : Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information. ≫

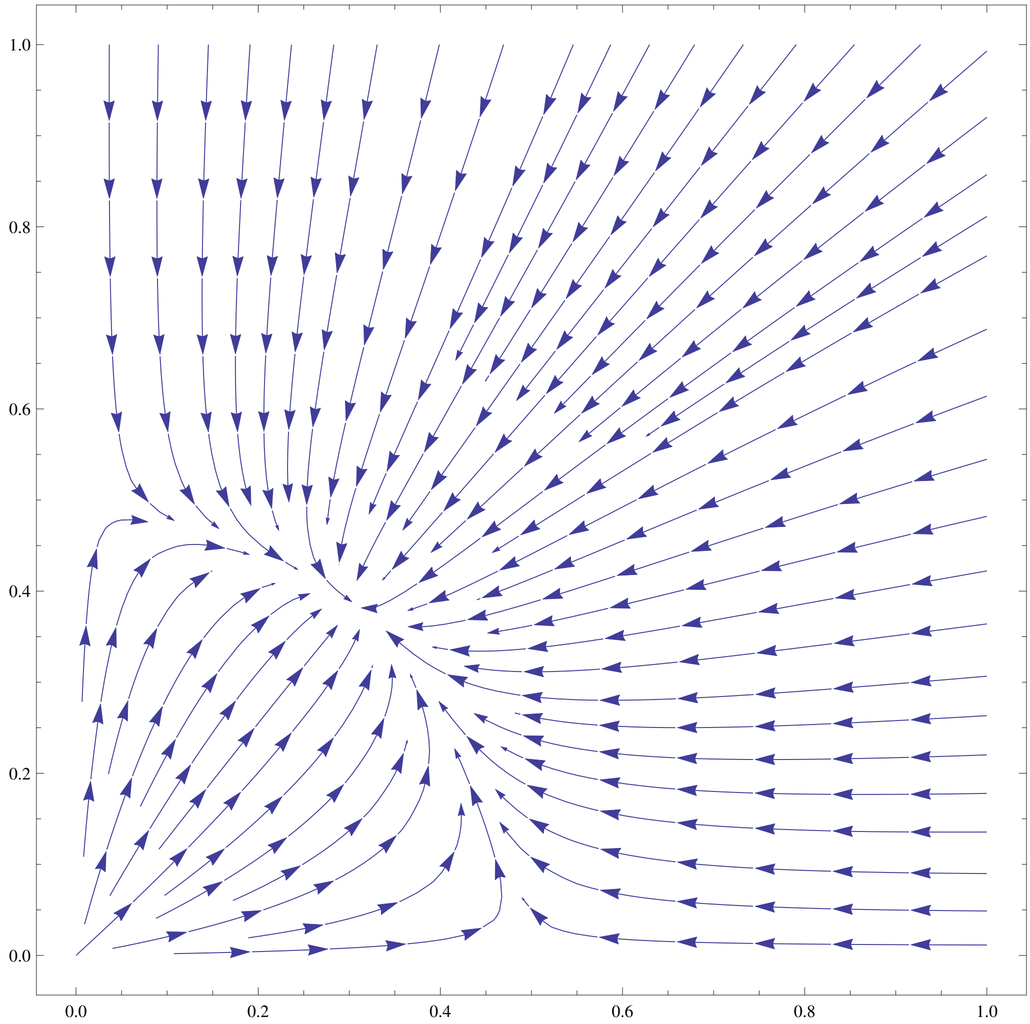
$$\left\{\left\{y \to \text{Function}\left[\{x\}, \frac{1}{4}\left(401 + 20\,\sqrt{2} + x - \sqrt{801 + 40\,\sqrt{2}}\,\sqrt{(1 + x)^2}\right)\right]\right\},\right.$$
$$\left.\left\{y \to \text{Function}\left[\{x\}, \frac{1}{4}\left(401 - 20\,\sqrt{2} + x + \sqrt{-\left(-801 + 40\,\sqrt{2}\right)\,(1 + x)^2}\right)\right]\right\}\right\}$$

```
(* Bsp: System von 2 Gleichungen: Konkurrierende Spezies,
vergleiche die logistische Gleichung fuer eine Dimension *)
```

```
f[x_, y_] := x (a - b1 x - b2 y)
g[x_, y_] := y (c - d1 x - d2 y)
(* Bsp Belegung der Parameter *)
a = 1;  b1 = 2;  b2 = 1;  c = 1;  d1 = 0.75;  d2 = 2;
sys = {f[x, y], g[x, y]}
```

{x (1 - 2 x - y), (1 - 0.75 x - 2 y) y}

```
StreamPlot[sys, {x, 0, 1}, {y, 0, 1}]
```




```
Clear[x, y]
Solve[{a - b1 x - b2 y == 0, c - d1 x - d2 y == 0}, {x, y}]
```

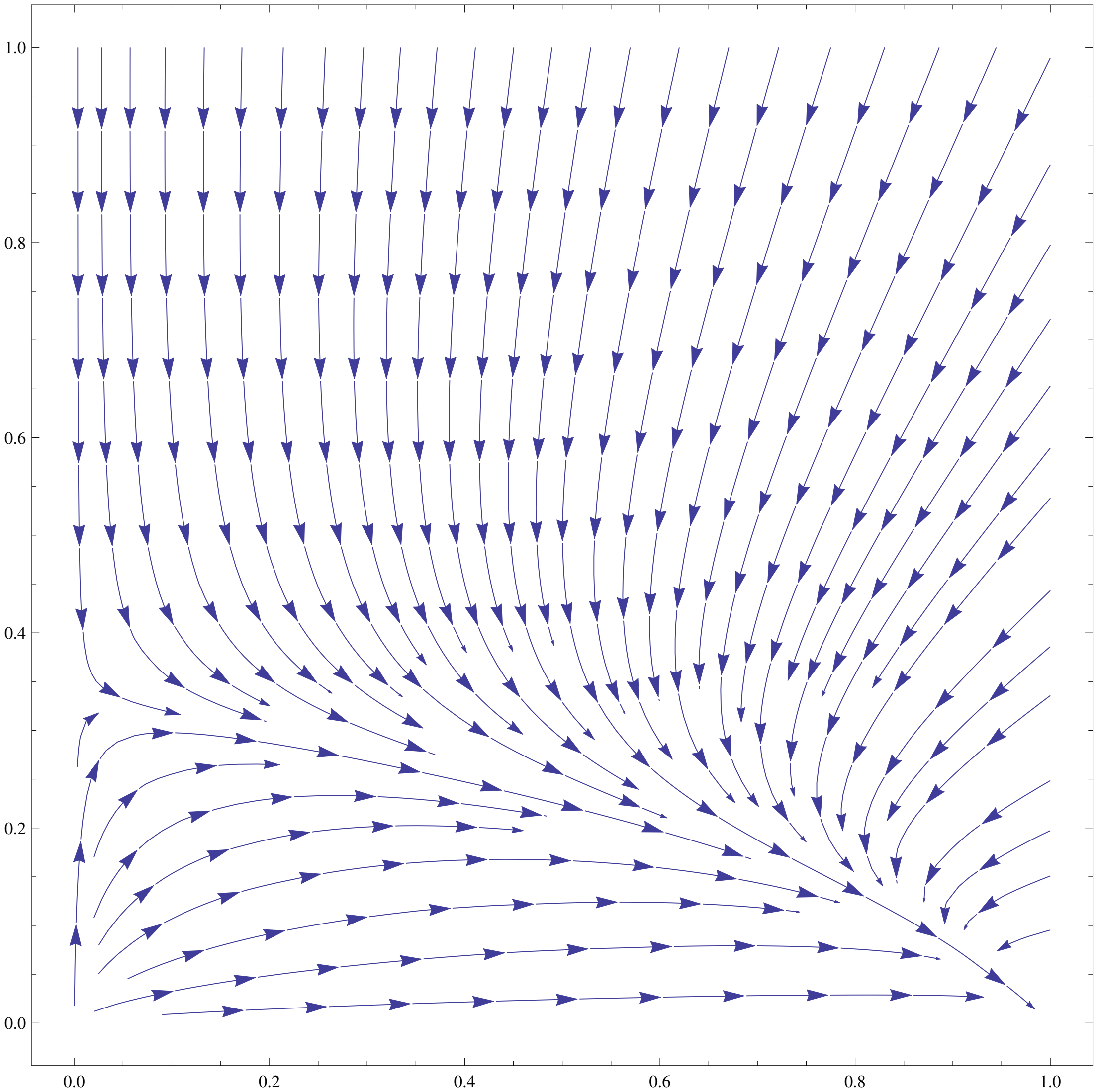
{{x → 0.307692, y → 0.384615}}

Es gibt einen LimesPunkt des Systems, den man direkt ausrechnen kann.

```
(* Bsp2 der Parameter *)
a = 1; b1 = 1; b2 = 1; c = 2 / 3; d1 = 0.75; d2 = 2;
sys = {f[x, y], g[x, y]}
```

$\left\{x\,(1 - x - y),\ \left(\frac{2}{3} - 0.75\,x - 2\,y\right) y\right\}$

```
StreamPlot[sys, {x, 0, 1}, {y, 0, 1}]
```



Die y - Spezies stirbt aus, x bleibt uebrig, im Limes wird die Loesung angenommen, die sich aus dem System bei y = 0 ergibt :
Solve a logistic equation :

```
restx = DSolve[x'[t] == a x[t] - b1 x[t]^2, x, t]
```

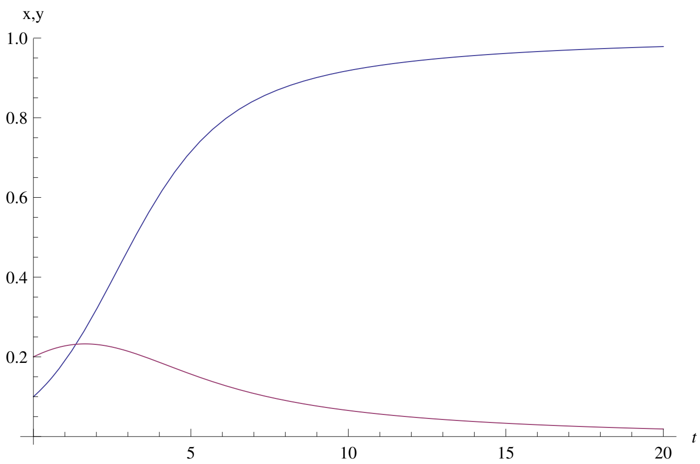
$$\left\{\left\{x \to \text{Function}\left[\{t\}, \frac{e^{t}}{e^{t} + e^{C[1]}}\right]\right\}\right\}$$

$$\text{Limit}\left[\frac{e^{t}}{e^{t} + e^{C[1]}}, t \text{ -> Infinity}\right]$$

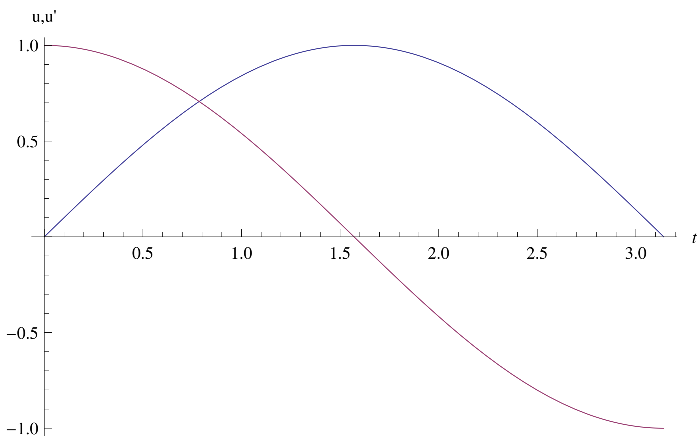
1

Bsp fuer eine konkrete Kurve mit NDSolve

```
(*  Clear[x,y,t,lsg] *)
a = 1.;  b1 = 1.;  b2 = 1;  c = 2. / 3;  d1 = 0.75;  d2 = 2;
lsg = NDSolve[{x'[t] == x[t] (a - b1 x[t] - b2 y[t]),
   y'[t] == y[t] (c - d1 x[t] - d2 y[t]), x[0] == 0.1, y[0] == 0.2}, {x, y}, {t, 0, 20}]
{{x → InterpolatingFunction[{{0., 20.}}, <>],
  y → InterpolatingFunction[{{0., 20.}}, <>]}}

Plot[{lsg[[1, 1, 2]][t], lsg[[1, 2, 2]][t]}, {t, 0, 20}, AxesLabel → {t, "x,y"}]
```




```
(* Anderes Bsp, get an InterpolatingFunction object
 approximating the solution of a differential equation: *)
ifun = First[
  u /. NDSolve[{u''[t] + u[t] == 0, u[0] == 0, u'[0] == 1}, u, {t, 0, π}]]
(* and Plot the function and its derivative: *)
Plot[{ifun[t], ifun'[t]}, {t, 0, π}, AxesLabel → {t, "u,u'"}]

InterpolatingFunction[{{0., 3.14159}}, <>]
```




```
(*  Bsp DGl-System mit Randwerten *)
```

```
dgl = DSolve[{x'[t] == y[t], y'[t] == -x[t], x[0] == 1, y'[1] == 2}, {x[t], y[t]}, t]
```

```
{{x[t] → Cos[t] - Cot[1] Sin[t] - 2 Csc[1] Sin[t],
  y[t] → -Cos[t] Cot[1] - 2 Cos[t] Csc[1] - Sin[t]}}
```

```
lsg = dgl[[1]] (* Beseitigung der Doppelklammer *)
```

```
{x[t] → Cos[t] - Cot[1] Sin[t] - 2 Csc[1] Sin[t],
 y[t] → -Cos[t] Cot[1] - 2 Cos[t] Csc[1] - Sin[t]}
```

```
(* Herausloesung beider Loesungsfunktionen *)
```

```
fx[t_] = x[t] /. lsg
fy[t_] = y[t] /. lsg
```

```
Cos[t] - Cot[1] Sin[t] - 2 Csc[1] Sin[t]
```

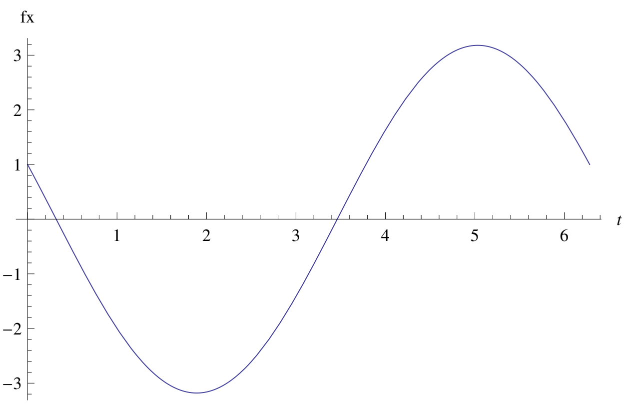
```
-Cos[t] Cot[1] - 2 Cos[t] Csc[1] - Sin[t]
```

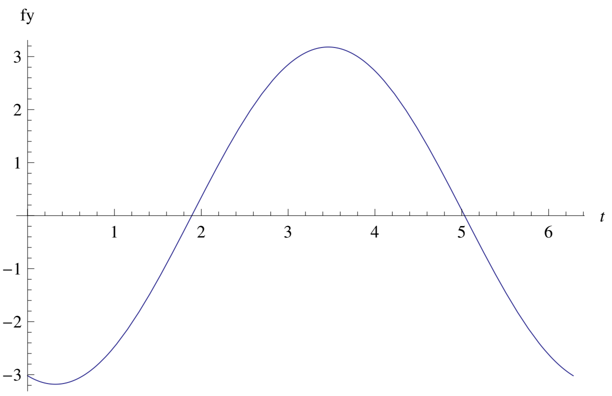
```
(* Vereinfachen *)
fx[t_] = ComplexExpand[fx[t]] // Simplify
fy[t_] = ComplexExpand[fy[t]] // Simplify
```

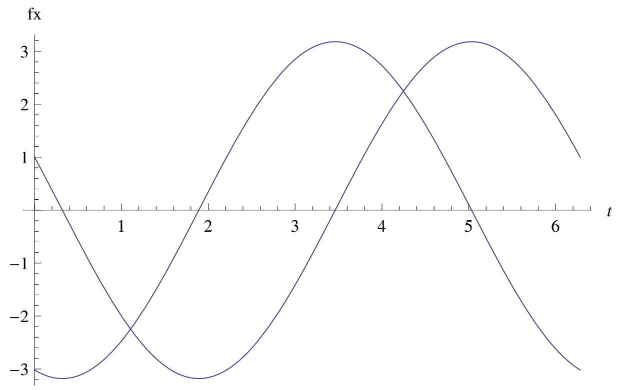
```
Csc[1] (Sin[1 - t] - 2 Sin[t])
```

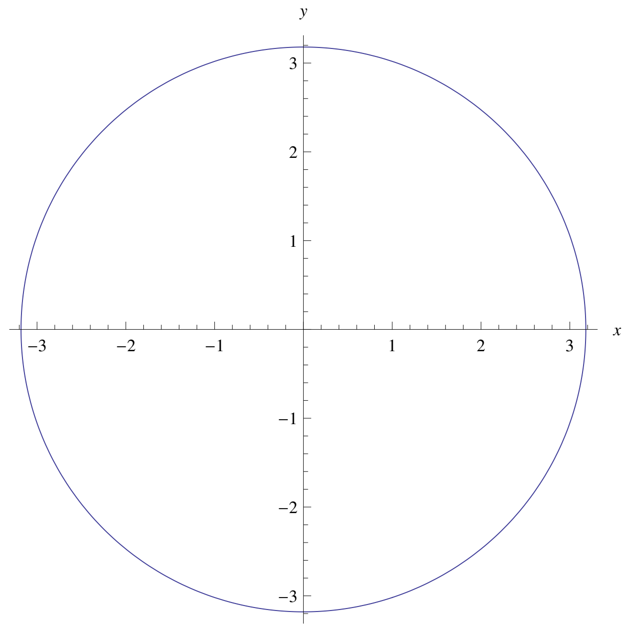
```
-(Cos[1 - t] + 2 Cos[t]) Csc[1]
```

```
f1 = Plot[fx[t], {t, 0, 2 Pi}, AxesLabel → {t, "fx"}]
f2 = Plot[fy[t], {t, 0, 2 Pi}, AxesLabel → {t, "fy"}]
Show[f1, f2]
ParametricPlot[{fx[t], fy[t]}, {t, 0, 2 Pi}, AspectRatio → 1, AxesLabel → {x, y}]
```










```
(* BSP: Michaelis-Menten DGl fuer emzymatische Reaktionen *)
mm = DSolve[{x'[t] == -V x[t] / (x[t] + Km), x[0] == 1}, x, t]
```

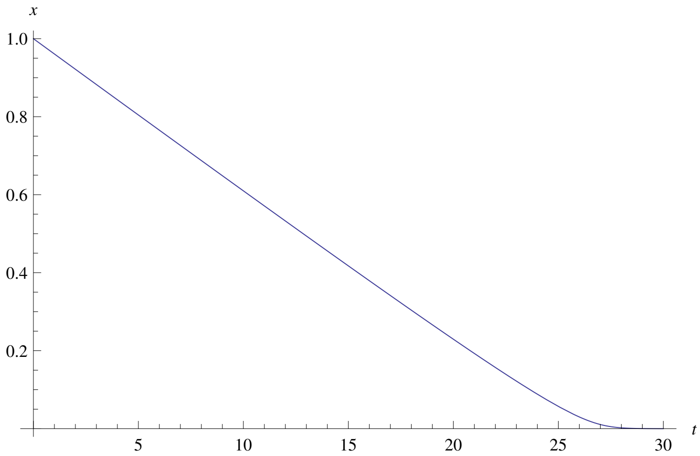
Solve::ifun : Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information. ≫

$$\left\{\left\{x \to \text{Function}\left[\{t\},\ \text{Km ProductLog}\left[\frac{e^{\frac{1}{\text{Km}}-\frac{t\,V}{\text{Km}}}}{\text{Km}}\right]\right]\right\}\right\}$$

```
xsp[t] = x[t] /. mm /.  V → 0.04 /. Km → 0.02
```

$$\left\{0.02\ \text{ProductLog}\left[50.\ e^{50.-2.\,t}\right]\right\}$$

```
Plot[0.02 ProductLog[50. e^(50 -2 t)], {t, 0, 30}, AxesLabel → {t, x}]
```



Hinweis : ProductLog ist die Loesung der Gleichung for w in $z = we^w$

```
Plot[ProductLog[x], {x, -1 / E, 1}]
```