Differentialgleichungen lösen mit Maple

Markus Antoni
Karlsruher Institut für Technologie (KIT), Institut für Analysis

Diese kleine Einführung entstand parallel zu der Veranstaltung "Übung zur Höheren Mathematik III für die Fachrichtung Physik" im WS 2012/2013. Sie dient vorrangig als Ergänzung zu den dort präsentierten Maple-Beispielen und erhebt keinen Anspruch auf Vollständigkeit oder Fehlerfreiheit.

Ich bedanke mich herzlich bei allen Teilnehmern der oben genannten Übung, die mich stets aufs Neue dazu motivierten, Maple als festen Bestandteil in die Übung zu integrieren.

Grundlagen

Der Anfang

Ein Maple-Worksheet starten wir stets mit der folgenden Zeile:

```
> restart;
```

Dieser Befehl löscht alle Zuordnungen und beseitigt jegliche Daten vorheriger Rechnungen. Sollte während einer Berechnung irgendetwas nicht klappen, obwohl alles irgendwie richtig erscheint, so kann ein nochmaliges Ausführen des ***restart***-Befehls hilfreich sein.

Ebenfalls am Anfang sollten mit dem ***with(...)***-Befehl spezielle Pakete geladen werden, die Maple zur Berechnung bestimmter Dinge benötigt. Beispiele hierfür sind die Pakete ***plots*** oder ***DEtools*** bzw. ***PDEtools***, wobei ersteres erweiterte Plot-Funktionen und letztere Befehle zum Lösen von Differentialgleichungen bereitstellt. Für unsere Zwecke wird daher folgende erste Zeile meist vollkommen ausreichen:

```
> restart; with(plots): with(DEtools): with(PDEtools):
```

Hierbei weisen wir noch darauf hin, dass mit einem **";"** oder einem **":"** stets Befehle ausgeführt werden. Eines dieser beiden Zeichen muss am Ende einer Anweisung **immer** stehen. Der Unterschied zwischen ihnen besteht lediglich darin, dass Maple bei einem **":"** keine Ausgabe erzeugt. Zum Vergleich:

```
> with(plots);
```

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *densityplot*, *display*, *dualaxisplot*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *interactiveparams*, *intersectplot*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *multiple*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *rootlocus*, *semilogplot*, *setcolors*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*]

Möchte man nähere Informationen zu speziellen Befehlen erhalten, so hilft einem die Hilfe-Funktion von Maple weiter, die man mit ***?Befehl;*** aufrufen kann. Zum Beispiel:

```
> ?plot;
```

Einfache Rechenoperationen

Maple beherrscht alle gängigen Rechenoperationen, deren Befehle anhand einiger Beispiele erklärt werden sollen:

```
> a + b; # Addition
```

$$a + b$$

```
> a - b; # Subtraktion
```

$$a - b$$

```
> a*b; # Multiplikation
```

$$a\,b$$

```
> a/b; # Division
```

$$\frac{a}{b}$$

```
> a^b; # Exponieren
```

$$a^b$$

Darüber hinaus sind natürlich auch zahlreiche Funktionen in Maple integriert, z.B.:

```
> exp(x); # Exponentialfunktion
```

$$e^x$$

```
> sin(x); cos(x); tan(x); # trigonometrische Funktionen
```

$$\sin(x)$$
$$\cos(x)$$
$$\tan(x)$$

```
> sqrt(x); surd(x,n); # Wurzeln
```

$$\sqrt{x}$$
$$\sqrt[n]{x}$$

Ebenfalls vorhanden sind Befehle für Summen und Integrale:

```
> sum(a[k], k = 1 .. n); # endliche Summe
```

$$\sum_{k=1}^{n} a_k$$

```
> sum(a[k], k = 1 .. infinity); # Reihe
```

$$\sum_{k=1}^{\infty} a_k$$

```
> int(f(t), t = a .. b); # Integration
```

$$\int_{a}^{b} f(t)\,\mathrm{d}t$$

Stehen in diesen Befehlen Werte oder Funktionen, die Maple auswerten kann, so wird direkt das Ergebnis dieser Operation ausgegeben.

```
> 2*2;
```

$$4$$

```
> exp(1);
```

$$\mathrm{e}$$

```
> sum(k, k = 1 .. 10);
```

$$55$$

```
> sum(1/(k^2), k = 1 .. infinity);
```

$$\frac{1}{6}\pi^2$$

```
> int(sin(t), t = 0 .. x);
```

$$1 - \cos(x)$$

Das Rechnen mit komplexen Zahlen ist in Maple natürlich auch möglich. Dabei ist allerdings zu beachten, dass die imaginäre Einheit ***i*** entgegen der üblichen Schreibweise mit einem großen ***I*** bezeichnet wird.

```
> (5+I*10)/(1-I*2);
```

$$-3 + 4\,\mathrm{I}$$

```
> I^2;
```

$$-1$$

Auch Gleichungen oder Gleichungssysteme kann Maple mit Hilfe des ***solve***-Befehls lösen. Die Syntax hierführ ist:

```
> # solve({Gleichung_1, Gleichung_2, ... , Gleichung_n},
  {Variable_1, Variable_2, ... , Variable_n}); # beachte die
  geschweifte Klammern!
```

Angewendet kann dies dann so aussehen:

```
> solve(a*x^2 + b*x + c = 0, x); # quadratische Gleichung
```

$$\frac{1}{2}\ \frac{-b+\sqrt{b^2-4\,a\,c}}{a},\ -\frac{1}{2}\ \frac{b+\sqrt{b^2-4\,a\,c}}{a}$$

```
> solve({2*a - b + c = 1, a + 2*b + d = 0, -a + c - d = 0, -b +
  c + d = 0}, {a, b, c, d}); # LGS
```

$$\left\{a=\frac{5}{13}, b=-\frac{1}{13}, c=\frac{2}{13}, d=-\frac{3}{13}\right\}$$

Funktionen

Die korrekte Definition einer Funktion ist wichtig, da nur bei richtiger Anwendung auf die Funktionswerte zugegriffen werden kann. Sie lautet allgemein:

```
> f := x -> f(x);
```

$$f := x \to f(x)$$

Hier ein paar Beispiele:

```
> f[1] := x -> x^2;
```

$$f_1 := x \to x^2$$

```
> f[2] := t -> exp(5*t); # die Funktionsvariable muss nicht
  unbedingt x sein
```

$$f_2 := t \to \mathrm{e}^{5\,t}$$

```
> f[3] := (x,y,z) -> sin(x*y) + cos(z)/arctan(y + z); #
  Funktion mit mehreren Veränderlichen
```

$$f_3 := (x, y, z) \to \sin(x\,y) + \frac{\cos(z)}{\arctan(y+z)}$$

```
> f[4] := x -> <cos(x), sin(x)>; # Vektor-wertige Funktion
```

$$f_4 := x \to \langle \cos(x), \sin(x) \rangle$$

```
> f[5] := x -> [cos(x), sin(x)]; # ersetzt man "< >" durch "[ ]
  ", wird Maple die Funktionswerte als Punkte auffassen
```

$$f_5 := x \to [\cos(x), \sin(x)]$$

An dieser Stelle sei noch angemerkt, dass wir mit ***":="*** Zuweisungen ausführen. Schreiben wir also zum Beispiel ***a := 3***, so hat ***a*** den festen Wert 3 und kann erst nach Ausführen eines ***restart*** wieder als Funktionsvariable benutzt werden.

Ein wenig schwieriger ist die Notation von abschnittsweise definierten Funktionen. Dies geschieht mit dem ***piecewise***-Befehl, den wir wie folgt aufrufen können:

```
> # piecewise(Bedingung_1, Funktion_1, Bedingung_2, Funktion_2,
  ... , Bedingung_n, Funktion_n, Funktion_otherwise);
```

Zum Beispiel:

```
> f := (x,y) -> piecewise(0 < abs(x) and abs(x) < 1 and -1 < y
  and  y < 0, 2*x, 0 < abs(x) and abs(x) < 1 and  y>=0 and y <
  x^2, 2*x-4*y/x, 0 < abs(x) and abs(x) < 1 and y >= x^2 and y
  < 1, -2*x,0); f(x,y);
```

$$f := (x, y) \rightarrow piecewise\left(0 < |x| \text{ and } |x| < 1 \text{ and } -1 < y \text{ and } y < 0, 2\,x, 0 < |x| \text{ and } |x| < 1 \text{ and } 0 \le y \text{ and } y < x^2, 2\,x - \frac{4\,y}{x}, 0 < |x| \text{ and } |x| < 1 \text{ and } x^2 \le y \text{ and } y < 1, -2\,x, 0\right)$$

$$\begin{cases} 2\,x & 0 < |x| \text{ and } |x| < 1 \text{ and } -1 < y \text{ and } y < 0 \\ 2\,x - \frac{4\,y}{x} & 0 < |x| \text{ and } |x| < 1 \text{ and } 0 \le y \text{ and } y < x^2 \\ -2\,x & 0 < |x| \text{ and } |x| < 1 \text{ and } x^2 \le y \text{ and } y < 1 \\ 0 & otherwise \end{cases}$$

Beachte hierbei, dass Maple Ausdrücke wie **"0 < x < 1"** nicht auswerten kann. Stattdessen müssen solche Ausdrücke entkoppelt und mit den logischen Befehlen ***and*** und ***or*** verknüpft werden.

Grafiken

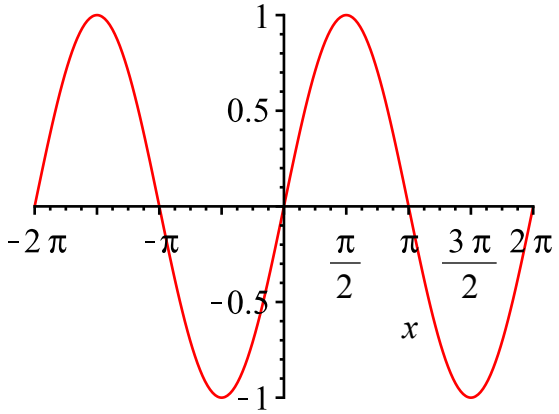
Zweidimensionale Zeichnungen

Zum Plotten von Funktionen stehen verschiedene Befehle zur Verfügung. Der Befehl für eine zweidimensionale Zeichnung einer explizit definierten Funktion ist der ***plot***-Befehl, mit der Notation

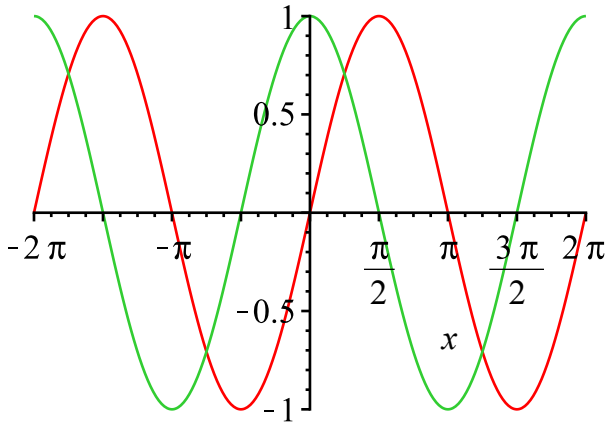
```
> # plot(f(x), x = a .. b, Optionen);
```

Bevor wir näher auf die zahlreichen Optionen eingehen, schauen wir uns erst einmal ein paar einfache Beispiele an:

```
> plot(sin(x), x = -2*Pi .. 2*Pi);
```

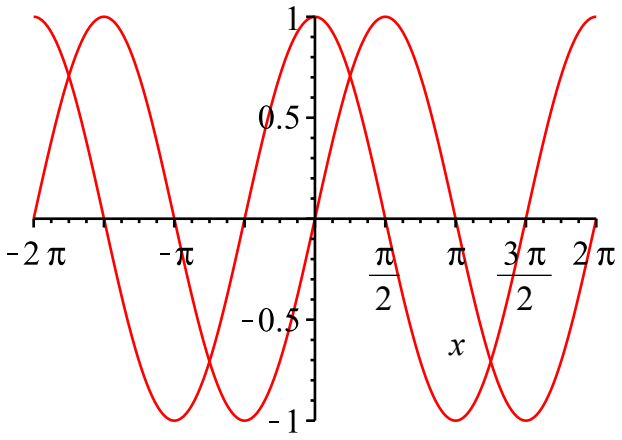


```
> plot([sin(x),cos(x)], x = -2*Pi .. 2*Pi); # mehrere Plots
  können mit "[ ]" oder "{ }" zusammengefasst werden.
```



Alternativ zu obiger Schreibweise, können verschiedene Plots mit dem ***display***-Befehl zusammen in einer Grafik angezeigt werden:

```
> Plot1 := plot(sin(x), x = -2*Pi .. 2*Pi): Plot2 := plot(cos
  (x), x = -2*Pi .. 2*Pi): display(Plot1, Plot2);
```



Hierbei wird allerdings bei allen möglichen Optionseinstellungen auf den Default-Wert zurückgegriffen (deshalb auch die gleiche Farbe). Dies beinhaltet unter vielen anderen Optionen:

axes: Dient der Einstellung, wie die Achsen des Koordinatensystems dargestellt werden sollen.
- mögliche Werte: **frame, boxed, normal, none**
- Default-Wert: **normal** (2D), **none** (3D)

color: Bestimmt die Farbe des Funktionsgraphen.
- mögliche Werte: **aquamarine, black, blue, brown, coral, cyan, gold, green, gray, magenta, maroon, navy, orange, pink, plum, red, sienna, turquoise, violet, wheat, white, yellow, etc.**
- Default-Wert: **red** (bis Maple 15, ab Maple 16 komplett neue Farbeinstellungen)

coords: Definiert das zugrunde liegende Koordinatensystem.
- mögliche Werte in 2D: **cartesian, logarithmic, parabolic, polar, etc.**
- mögliche Werte in 3D: **cylindrical, paraboloidal, rectangular, spherical, etc.**
- Default-Wert: **cartesian** (2D)**, rectangular** (3D)

discont: Gibt an, ob Unstetigkeiten der Funktion beim Zeichnen berücksichtigt werden sollen, oder nicht.
- mögliche Werte: **true, false**
- Default-Wert: **false**

filled: Bestimmt, ob die Fläche zwischen Schaubild und x-Achse ausgefüllt ist oder nicht.
- mögliche Werte: **true, false**
- Default-Wert: **false**

font: Dient der Einstellung der Schriftart, des Schriftstils und der Schriftgröße für Textobjekte innerhalb der Zeichnung. Es wird dabei unterschieden zwischen **axesfont, captionfont, labelfont** bzw. **titlefont**.
- mögliche Werte: **["Schriftart", "Schriftstil", "Schriftgröße"]**, wobei als Schriftart **Times, Courier, Helvetica** oder **Symbol** möglich ist, sowie jeder Font, der vom Betriebssystem unterstützt wird. Als Schriftstile stehen **roman, bold, italic, bolditalic, oblique** oder **boldoblique** zur Verfügung.

labels: Bestimmt die Bezeichnung der Koordinatenachsen.
- mögliche Werte: **["", ""]** (keine Beschriftung), **["a","b"]** (Achsen werden mit a und b bezeichnet)
- Default-Wert: keine Beschriftung

linestyle: Bestimmt den Zeichenstils des Funktionsgraphen.
- mögliche Werte: **solid, dot, dash, dashdot, longdash, spacedash, spacedot**
- Default-Wert: **solid**

numpoints: Gibt die minimale Anzahl an Punkten an, die Maple für die Zeichung verwendet.
- mögliche Werte: irgendeine natürliche Zahl
- Default-Wert: **200**

scaling: Bestimmt, ob die Maßstäbe der Achsen gleich sind oder nicht.
- mögliche Werte: **constrained** (gleich), **unconstrained** (nicht gleich)
- Default-Wert: **unconstrained**

style: Gibt den Stil der Oberfläche der zu zeichnenden Objekte an.
- mögliche Werte: **line, point, patch, patchnogrid, contour, patchcontour, wireframe**
- Default-Wert: **line** (2D), **patch** (3D)

symbol: Bestimmt den Zeichenstil für zu zeichnende Symbole.
- mögliche Werte: **asterisk, box, circle, cross, diagonalcross, diamond, point, solidbox, solidcircle, soliddiamond**
- Default-Wert: **diamond**

symbolsize: Definiert die Größe der zu zeichnenden Symbole.
- mögliche Werte: irgendeine natürliche Zahl
- Default-Wert: **10**

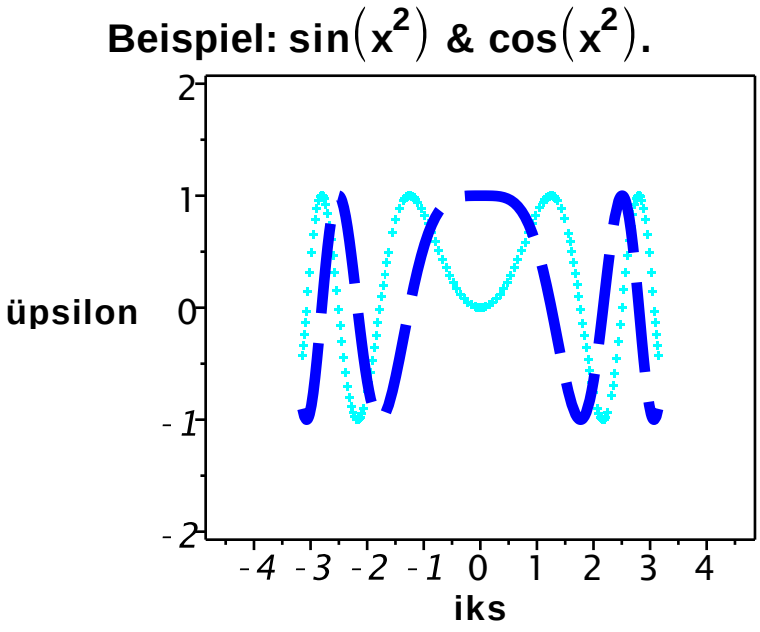
thickness: Definiert die Dicke der zu zeichnenden Linien.
- mögliche Werte: irgendeine nichtnegative ganze Zahl
- Default-Wert: **0**

title: Gibt die Überschrift der Zeichnung an. Sollen Formeln mit Zeichen kombiniert werden, so funktioniert dies mit dem ***typeset***-Befehl (siehe ***?typeset;*** für weitere Informationen).
- mögliche Werte: Zeichenkette in ""
- Default-Wert: keine Überschrift

view: Definiert den zu zeichnenden Ausschnitt.
- mögliche Werte: **[x_min .. x_max, y_min .. y_max, z_min .. z_max]**
- Default-Wert: der Ausschnitt, der den gesamten Funktionsgraphen im angegebenen Intervall enthält

Die Optionen werden dabei stets durch den Ausdruck **"Option = Wert"** festgelegt. Im Maple-Code sieht dies dann z.B. so aus (die Optionen müssen dabei nicht in alphabetischer Reihenfolge sein):

```
> plot([sin(x^2),cos(x^2)], x = -Pi .. Pi, axes = boxed, color
  = [cyan, blue], coords = cartesian, discont = false, filled =
  false, axesfont = [Helvetica, italic, 10], labelfont =
  [Helvetica, bolditalic, 10], titlefont = [Helvetica, bold,
  12], labels =["iks","üpsilon"], linestyle = [solid, dash],
  numpoints = 100, scaling = unconstrained, style = [point,
  line], symbol = cross, thickness = [1,4], title = typeset
  ("Beispiel: %1 & %2.", sin(x^2),cos(x^2)), view = [-1.5*Pi ..
  1.5*Pi, -2 .. 2]);
```



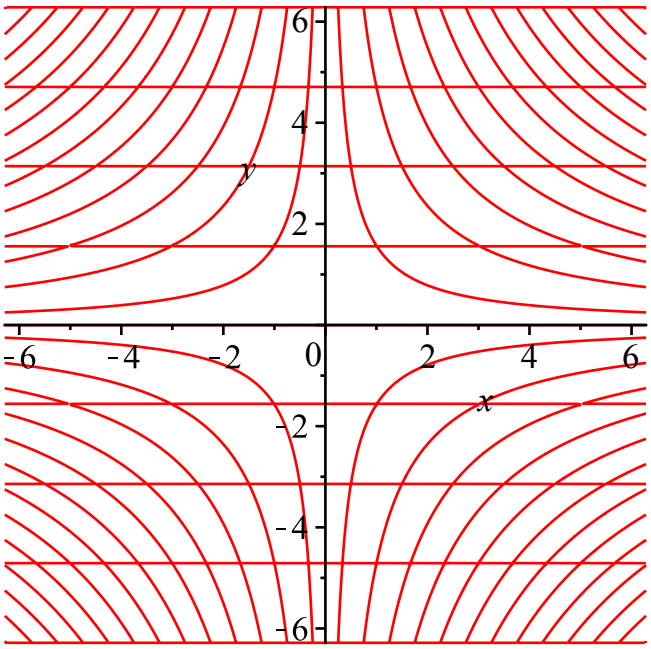
Insbesondere können Optionen, die nur einzelne Funktionsgraphen betreffen sollen, mit eckigen Klammern spezifiziert werden (siehe ***color***, ***linestyle***, ***thickness*** im obigen Beispiel).

Mittels ***implicitplot*** können implizit definierte Funktionen geplottet werden, d.h. es können z.B. Lösungsmengen von Gleichungen der Form **f(x,y) = c** dargestellt werden, der Befehl hierfür lautet allgemein

```
> # implicitplot(f(x,y) = c, x = a .. b, y = c(x) .. d(x),
  Optionen);
```

Wird in obiger Notation das **"= c"** weggelassen, so wird Maple automatisch den Ausdruck **"f(x,y) = 0"** plotten. Die meisten möglichen Optionen, die wir im vorherigen Abschnitt augelistet haben, können auch hier angewendet werden. Darüber hinaus gibt es allerdings noch spezielle Optionen, die der interessierte Leser in der Maple-Hilfe finden kann (z.B. via ***?imlicitplot;***). Hier noch ein Beispiel:

```
> implicitplot(tan(y)*cos(x*y), x = -2*Pi .. 2*Pi, y = -2*Pi ..
  2*Pi, numpoints = 500000);
```



Dreidimensionale Zeichnungen

Auch zum Zeichnen von dreidimensionalen Objekten stellt Maple Befehle bereit. Hierbei wollen wir zunächst den ***plot3d***-Befehl vorstellen. Er hat die Syntax:

```
> # plot3d(f(x,y), x = a .. b, y = c .. d, Optionen);
> # plot3d([f_1(x,y), f_2(x,y), f_3(x,y)], x = a .. b, y = c ..
  d, Optionen);
```

Wie auch im Falle von zweidimensionalen Zeichnungen gibt es wieder zahlreiche Optionen, die das Aussehen der Grafik nach Belieben verändern können. Neben den bereits im vorigen Abschnitt aufgeführten Optionen existieren darüber hinaus auch noch Einstellungen für Licht und Schatten, sowie verschiedene Farboptionen, die die Oberflächen der Grafiken in unterschiedlichen Farben strahlen lassen. Auch hier präsentieren wir nur eine kleine Auswahl und verweisen den interessierten Leser auf die Hilfe-Funktion von Maple:

color: Definiert die Farbe der Zeichnung.
- mögliche Werte: neben den oben angegebenen Farben ist es nun auch möglich der Farbe eine Funktion **f(x,y)** zuzuordenen.

light: Fügt der Zeichnung eine Lichtquelle hinzu.
- mögliche Werte: **[phi, theta, r, g, b]**, wobei die Winkel phi und theta die Einfallsrichtung des Lichtes bestimmen und r,g und b den Rot-, Gelb- und Blauanteil des Lichtes (müssen Werte zwischen 0 und 1 sein).
- Default-Wert: keine Lichtquelle

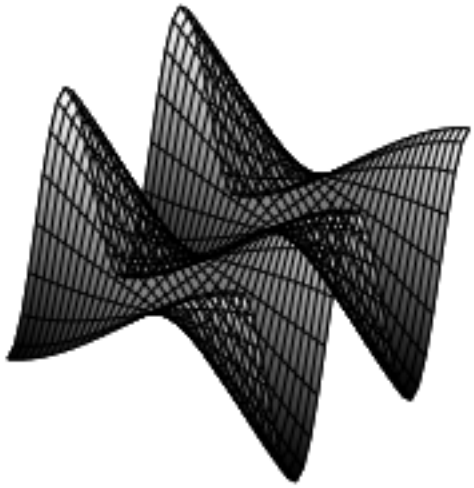
lightmodel: Verändert die Art des Lichts.
- mögliche Werte: **none, light1, light2, light3, light4**
- Default-Wert: **light3**

shading: Verändert die Färbung der Zeichung.
- mögliche Werte: **xy, xyz, z, zgrayscale, zhue, none**

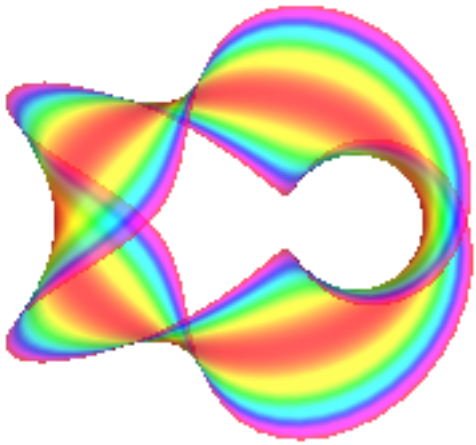
transparency: Definiert die Transparenz der Zeichung.
- mögliche Werte: Wert zwischen **0** (nicht transparent) und **1** (völlig transparent)
- Default-Wert: **0**

Nachfolgend zwei Beispiele mit Code:

```
> plot3d(cos(x)*sin(y), x = -2*Pi .. 2*Pi, y = -1 .. 1, shading
  = zgrayscale, transparency = 0.1, numpoints = 1000);
```



```
> plot3d([6+x*cos(2*y), y, x*sin(2*y)], x = -Pi .. Pi, y = 0 ..
  2*Pi, coords = cylindrical, color = x^2, style = patchnogrid,
  shading = zhue, scaling = constrained, lightmodel = light1,
  transparency = 0.5, numpoints= 5000);
```



Für implizit definierte Funktionen existiert ebenfalls eine Plot-Anweisung, der ***implicitplot3d***-Befehl. Völlig analog zum zweidimensionalen Fall heißt hier die Syntax:

```
> # implicitplot3d(f(x,y,z) = c, x = a .. b, y = c .. d, z = e
  .. f, Optionen);
```

Und auch die Optionen decken sich mit denen des gewöhnlichen ***plot3d***-Befehls. Ein Beispiel könnte so aussehen:

```
> implicitplot3d(x^3 + y^3 + z^3 + 3 = (x + y + z)^2, x = -5 ..
  5, y = -5 .. 5, z = -5 .. 5, numpoints = 10000, orientation=
  [-115,50,-140]);
```

Zu guter Letzt wollen wir noch den ***spacecurve***-Befehl betrachten. Wie der Name schon sagt, können wir damit Kurven im Raum skizzieren lassen. Die Anweisung lautet im Allgemeinen:

```
> # spacecurve([f_1(t), f_2(t), f_3(t)], t = a .. b, Optionen);
```

Geplottet wird hierbei die Spur der Kurve, siehe zum Beispiel:

```
> spacecurve([r*cos(3*r), r*sin(3*r), r], r = -5 .. 5,
  numpoints = 1000, thickness = 3, orientation = [40, 75, 0]);
```

Um die zeitliche Entwicklung einer Kurve im Raum darzustellen, benötigen wir weitere Befehle, denen wir im nächsten Abschnitt begegnen werden.

Animationen

Auch bewegte Bilder sind in Maple möglich. Wollen wir eine kontinuierliche Abfolge von Zeichungen sehen, so benutzen wir den folgenden Befehl:

```
> # animate(Plot-Befehl, [Plot-Anweisung (abh. von T)], T = a .
  . b, Optionen);
```

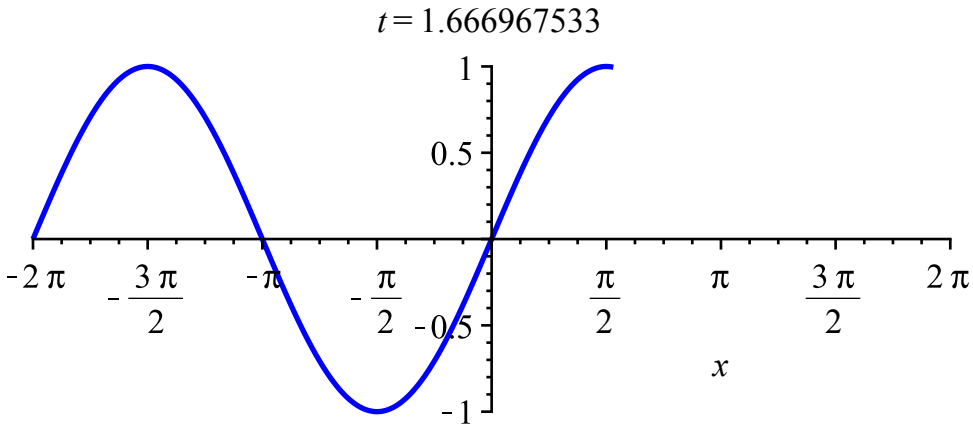
Dabei steht der Ausdruck "Plot-Befehl" für den jeweilig zugrundeliegenden Maple-Plot-Befehl, z.B. ***plot***, ***plot3d***, ***DEplot*** (siehe nächstes Kapitel), etc., aus dessen Bildern Maple eine Animation erstellt. Die darauf folgende Anweisung in **"[ ]"** enthält die Anweisungen der zu plottenden Bilder, wie sie bei den jeweiligen Plot-Befehlen aufgeführt wurden. Unter Optionen können Einstellungen für die einzelnen Plot-Bilder vorgenommen werden, sie können aber auch in den Plot-Anweisungen, d.h. in den eckigen Klammern davor stehen. Darüber hinaus können hier auch noch animationsspezifische Befehle aufgeführt werden. Die beiden wichtigsten sind:

frames: Gibt die Anzahl der zu erstellenden Bilder an.
- mögliche Werte: irgendeine natürliche Zahl
- Default-Wert: **25**

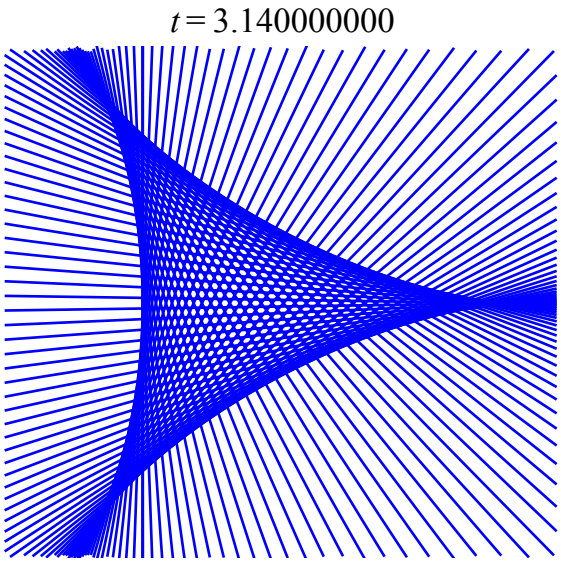
trace: Gibt die Anzahl der Spurbilder an, die bei der Animation stehen bleiben.
- mögliche Werte: eine natürliche Zahl n (dann bleiben n Bilder stehen, gleichverteilt auf das Intervall [a,b]) oder eine Liste von natürlichen Zahlen [n_1, ... , n_k] (dann bleiben die Frames n_1 bis n_k stehen)
- Default-Wert: **0**

Führen wir einen ***animate***-Befehl aus, so öffnet sich nach Klicken auf die Grafik am oberen Rand des Fensters eine Leiste, in der man u.a. die Animation starten, aber auch die FPS-Zahl ändern kann, um die Animationsgeschwindigkeit dem jeweiligen Zweck anzupassen. Dies wollen wir anhand einiger Beispiele näher betrachten:

```
> animate(plot, [sin(x), x = -2*Pi .. t, color = blue,
  thickness = 2], t = -2*Pi .. 2*Pi, frames = 50);
```

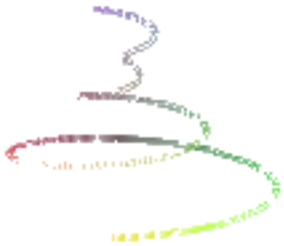



```
> animate(plot, [tan(-t/2)*(x-cos(t)) + sin(t), x = -2.5 ..
  3.5, axes = none, color = blue, thickness = 0, view = [-2.5 .
  . 3.5, -3 .. 3]], t = -3.14 .. 3.14, frames = 100, trace =
  100);
```



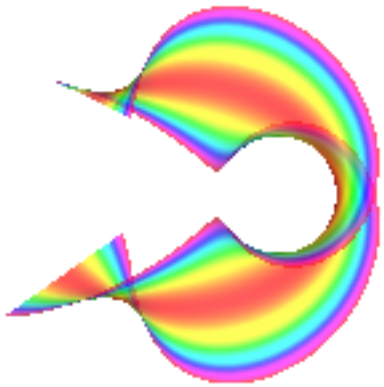
```
> animate(spacecurve, [[r*cos(3*r), r*sin(3*r), r], r = -5 ..
  T, numpoints = 1000, orientation = [40, 75, 0], thickness =
  3], T = -5 .. 5, frames = 50);
```

$$T = 1.530612245$$



```
> animate(plot3d,[[6+x*cos(2*y), y, x*sin(2*y)], x = -Pi .. Pi,
  y = 0 .. T, coords = cylindrical, color = x^2, style =
  patchnogrid, shading = zhue, scaling = constrained,
  lightmodel = light1, transparency = 0.5, numpoints= 5000], T
  = 0 .. 2*Pi, frames = 50);
```

$$T = 4.487989506$$

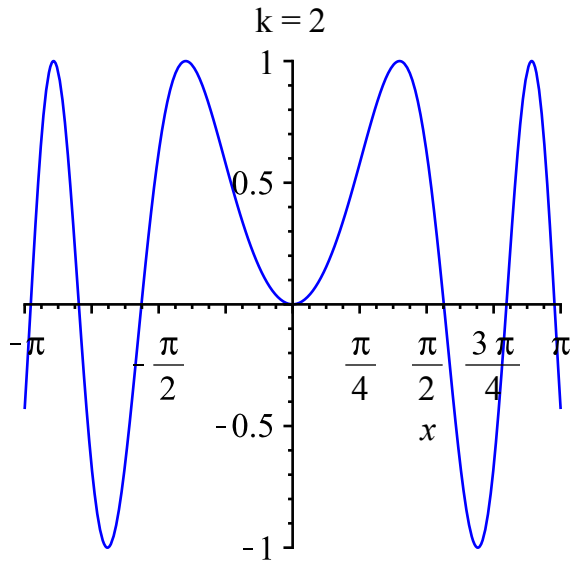


Neben den gewöhnlichen Animationen mit einem kontinuierlichen Parameter (der meist die Zeit darstellen soll), kann es natürlich auch sein, dass wir eine Abfolge von Bildern mit diskretem Parameter wünschen. Dies tritt zum Beispiel auch dann auf, wenn wir schlicht eine Folge von Grafiken hintereinander ausführen wollen. Hier können wir zu einer der beiden folgenden Varianten zurückgreifen:

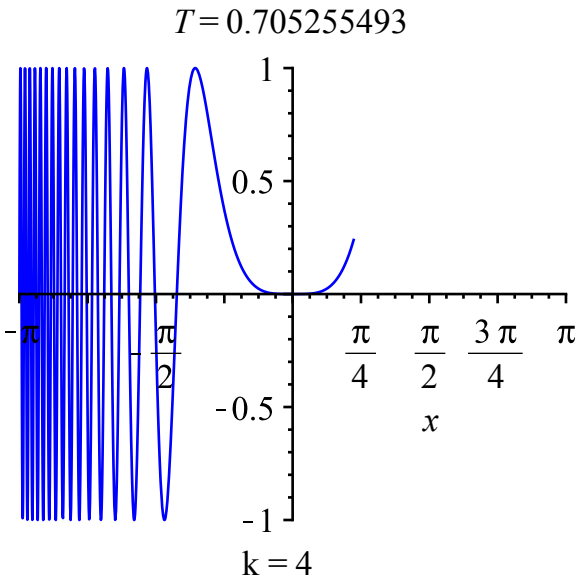
```
> # ani := k -> "Plot-Befehl":
  # display([seq(ani(k), k = 1 .. N)], insequence = true);
```

In der ersten Zeile sind sowohl stationäre als auch animierte Plotting-Befehle möglich, d. h. wir können damit auch eine Folge von Animationen darstellen:

```
> ani := k -> plot(sin(x^k), x = -Pi .. Pi, color = blue, title
  = typeset("k = ", k)):
  display([seq(ani(k), k = 0 .. 4)], insequence = true);
```




```
> ani := k -> animate(plot,[sin(x^k), x = -Pi .. T, color =
  blue, caption = typeset("k = ", k)], T = -Pi .. Pi, frames =
  50):
  display([seq(ani(k), k = 0 .. 4)], insequence = true);
```



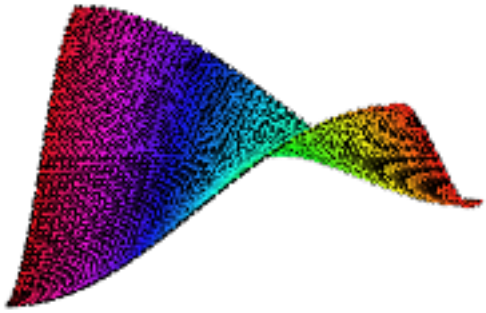
In der zweiten Möglichkeit benutzen wir eine for-Schleife, um zuerst die einzelnen Plots bzw. Animationen in einem Feld/Array zu speichern und sie anschließend als "Animation" zusammenzufügen:

```
> # for k from 1 to N do
  # ani[k] := "Plot-Befehl":
  # end do:
  # display([seq(ani[k], k = 1 .. N)], insequence = true); #
  auf eckige Klammern hier achten!
```

Auch hier zwei Beispiele:

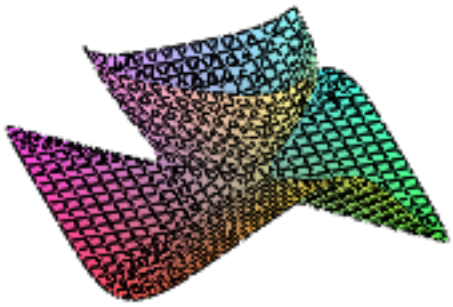
```
> for k from 0 to 12 do
  ani[k] := plot3d(cos(Pi/6*k)*sin(Pi/6*x)*sin(Pi/6*y), x = -3
  .. 3, y = -3.. 3, numpoints = 3000, title = typeset("k = ",
  k), color = x):
  end do:
  display([seq(ani[k], k = 0 .. 12)], insequence = true);
```

$k = 4$



```
> for k from 1 to 11 do
  ani[k] := implicitplot3d( x^k + y^k + z^k - 3 = (x + y + z)^
  (k-1), x = -5 .. 5, y = -5 .. 5, z = -5 .. 5, numpoints =
  10000, title = typeset("k = ", k), orientation= [-115,50,
  -140]):
  end do:
  display([seq(ani[k], k = 1 .. 11)], insequence = true);
```

$k = 5$



Diese Möglichkeiten können natürlich auch dazu benutzt werden, den ***animate***-Befehl komplett zu umgehen, da auch hier bei geeignetem ***N*** ein flüssiger Bildlauf entstehen kann (Maple macht ja auch nichts anderes, als das Parameterintervall zu diskretisieren).

Felder

Als letzten Punkt möchten wir hier noch zeigen, wie mehrere Grafiken nebeneinander dargestellt werden können. Dies geschieht mit Feldern und dem ***display***-Befehl. Unabhänig davon, ob Zeichnungen oder Animationen nebeneinander gestellt werden sollen, lautet die Syntax hierfür:

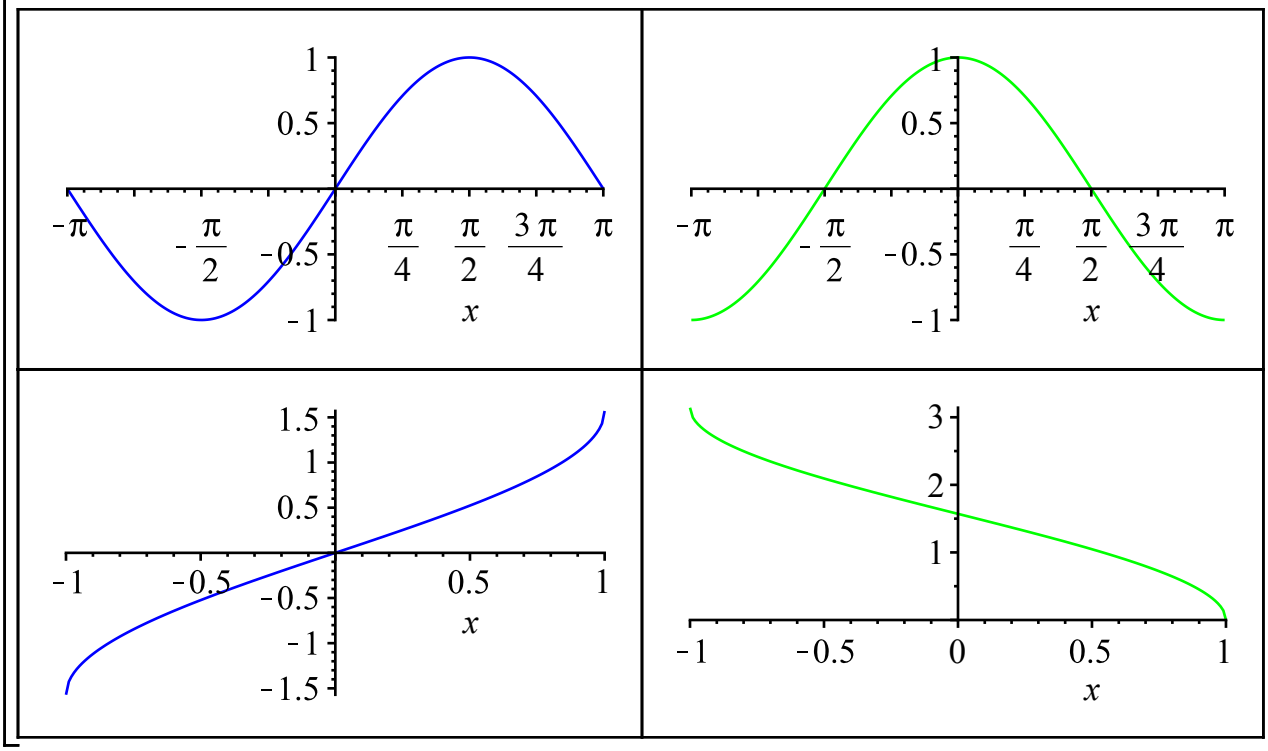
```
> # G := array(1 .. n, 1 .. m):
  # G[i,j] := "j. Zeichnung bzw. Animation in i. Zeile":
  # display(G);
```

Alternativ kann man auch folgendes schreiben:

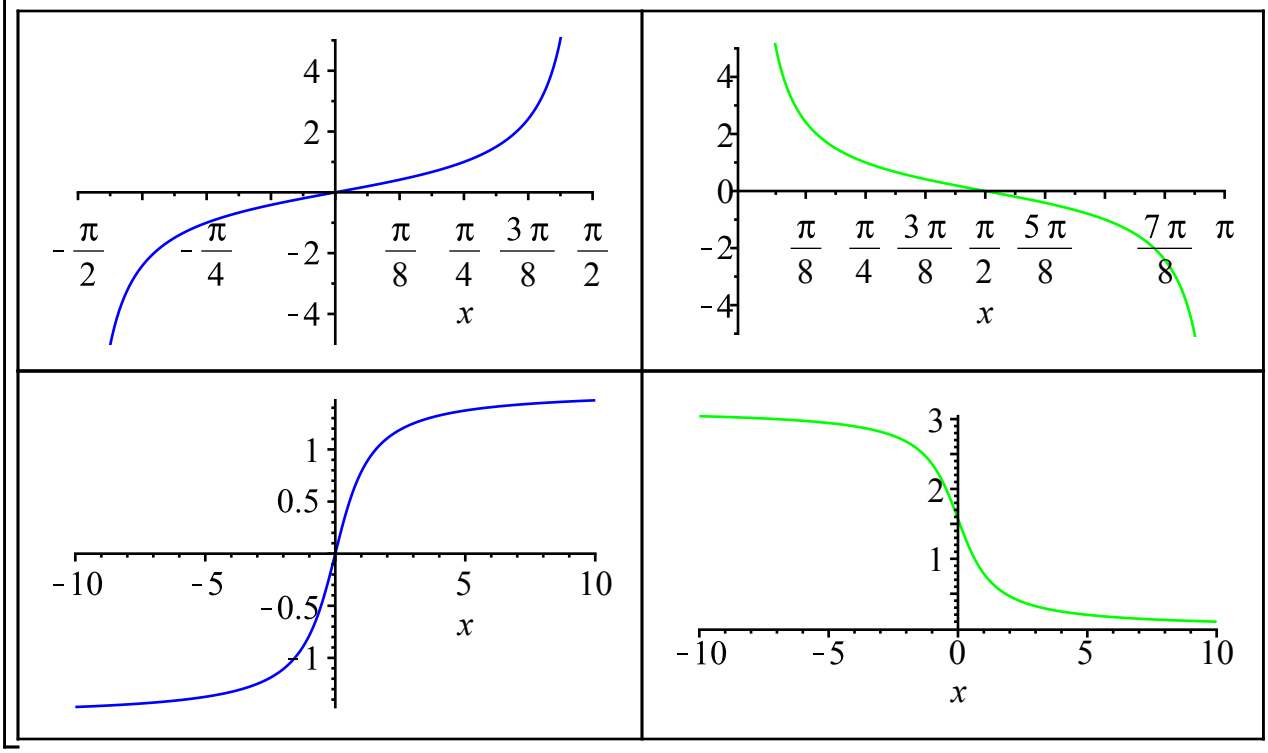
```
> # Grafik_1 := "1. Zeichnung bzw. Animation":
  # Grafik_2 := "2. Zeichnung bzw. Animation":
  # ...
  # Grafik_n := "n. Zeichnung bzw. Animation":
  # display(array([ [Grafik_1, ...], ... ,[ ..., Grafik_n] ]));
  # jede Zeile wird in eckigen Klammern zusammengefasst
```

In Beispielen kann dies dann so aussehen:

```
> B := array(1 .. 2, 1 .. 2):
  B[1,1] := plot(sin(x), x = -Pi .. Pi, color = blue):
  B[1,2] := plot(cos(x), x = -Pi .. Pi, color = green):
  B[2,1] := plot(arcsin(x), x = -1 .. 1, color = blue):
  B[2,2] := plot(arccos(x), x = -1 .. 1, color = green):
  display(B);
```

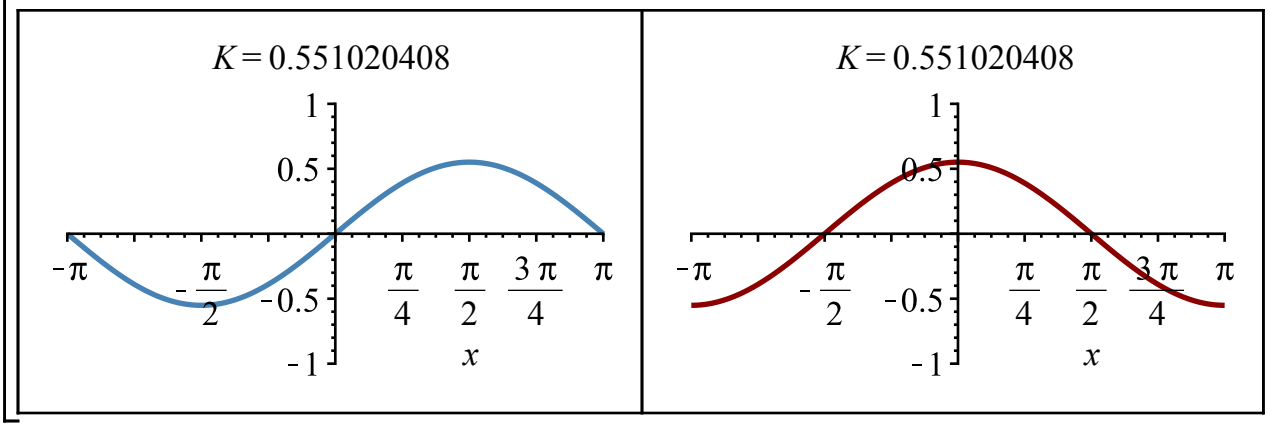


```
> B_1 := plot(tan(x), x = -Pi/2 .. Pi/2, color = blue, view =
  -5 .. 5):
  B_2 := plot(cot(x), x = 0 .. Pi, color = green, view = -5 ..
  5):
  B_3 := plot(arctan(x), x = -10 .. 10, color = blue):
  B_4 := plot(arccot(x), x = -10 .. 10, color = green):
  display(array([ [B_1, B_2] , [B_3, B_4] ]));
```



Ebenfalls möglich ist es, Animationen oder Folgen von Grafiken nebeneinander ablaufen zu lassen. Hier ein Beispiel:

```
> C_1 := animate(plot,[K*sin(x), x= -Pi .. Pi, color =
  "SteelBlue", thickness = 2], K = -1 .. 1, frames = 50):
  C_2 := animate(plot,[K*cos(x), x= -Pi .. Pi, color =
  "DarkRed", thickness = 2], K = -1 .. 1, frames = 50):
  display(array([C_1,C_2]));
```



Differentialgleichungen

Ableitungen

Bevor wir uns den Differentialgleichungen widmen, wollen wir kurz über die Ableitungsbefehle sprechen. Die Formulierung hierfür lautet:

```
> diff(g(x),x);
```

$$\frac{d}{dx} g(x)$$

Für die zweite bzw. n-te Ableitung schreiben wir:

```
> diff(g(x),x,x); # alternativ: diff(g(x),x$2);
```

$$\frac{d^2}{dx^2} g(x)$$

```
> diff(g(x),x$n);
```

$$\frac{d^n}{dx^n} g(x)$$

Wollen wir eine Funktion mit mehreren Veränderlichen partiell ableiten, so schreiben wir:

```
> diff(g(x,y),x);
```

$$\frac{\partial}{\partial x} g(x,y)$$

```
> diff(g(x,y),x,y); # hier wird zuerst nach x abgeleitet
```

$$\frac{\partial^2}{\partial y\, \partial x} g(x,y)$$

```
> diff(g(x,y),x$n,y$m);
```

$$\frac{\partial^m}{\partial y^m} \left(\frac{\partial^n}{\partial x^n} g(x,y) \right)$$

Geben wir eine Funktion explizit an, so kann Maple in den meisten Fällen die Ableitung angeben, z.B.:

```
> diff(sqrt(x),x);
```

$$\frac{1}{2\sqrt{x}}$$

```
> diff(exp(2*x),x$n);
```

$$2^n e^{2x}$$

```
> diff(x^y + y^x,x,y);
```

$$\frac{x^y \ln(x)\, y}{x} + \frac{x^y}{x} + \frac{y^x x \ln(y)}{y} + \frac{y^x}{y}$$

Analytische Lösungen

Maple kann für viele gewöhnliche Differentialgleichungen eine analytische Lösung angeben. Dazu gehören insbesondere lineare Differentialgleichungen und die meisten Differentialgleichungen erster Ordnung. Die Formulierung in Maple lautet hier dann zum Beispiel:

```
> # Dgl := diff(y(x),x) = f(x,y(x));
> # Dgl := p(x,y(x)) + diff(y(x),x)*q(x,y(x)) = 0; # exakte
  Differentialgleichung
```

Wollen wir Maple eine Lösung analytisch berechnen lassen, so erreichen wir das mit dem ***dsolve***-Befehl:

```
> # dsolve(Dgl, y(x), Optionen) # allgemeine Lösung
> # dsolve({Dgl, Anf}, y(x), Optionen) # Lösung mit
  Anfangswerten
```

Anf kann hier irgendein Anfangswert oder auch eine Liste von Anfangswerten sein. Im zweiten Fall wird Maple (sofern möglich) für jeden Anfangswert eine Lösung berechnen. Unter Optionen können wir Maple genauere Vorschriften machen, wie die Lösung angegeben (***explicit***, ***implicit***) und mit welchen Methoden die Differentialgleichung gelöst werden soll, z.B. Potenzreihenansatz, Laplace-Methode (siehe ***?dsolve;*** für weitere Informationen). Wir wollen noch anmerken, dass ***Dgl*** auch ein System von Differentialgleichungen, sowie ***y(x)*** eine Funktion mit mehreren Komponenten darstellen kann. Die beiden Ausdrücke müssen hierbei dann lediglich in **{ }** oder **[ ]** gesetzt werden.

Betrachten wir ein paar Beispiele:

```
> Dgl_1 := diff(y(x),x) = a*y(x);
```

$$Dgl\_1 := \frac{d}{dx} y(x) = a\, y(x)$$

```
> dsolve(Dgl_1, y(x));
```

$$y(x) = \_C1\, e^{a x}$$

```
> dsolve({Dgl_1, y(0) = a}, y(x));
```

$$y(x) = a\, e^{a x}$$

```
> Dgl_2 := 2*ln(x)*y(x)^3 + diff(y(x),x)*x^2 = 0;
```

$$Dgl\_2 := 2 \ln(x)\, y(x)^3 + \left(\frac{d}{dx} y(x) \right) x^2 = 0$$

```
> dsolve(Dgl_2, y(x), implicit);
```

$$\frac{1}{y(x)^2} + \frac{4 \ln(x)}{x} + \frac{4}{x} - \_C1 = 0$$

```
> dsolve(Dgl_2, y(x), explicit);
```

$$y(x) = \frac{\sqrt{-(4 \ln(x) + 4 - \_C1\, x)\, x}}{4 \ln(x) + 4 - \_C1\, x}, y(x) = -\frac{\sqrt{-(4 \ln(x) + 4 - \_C1\, x)\, x}}{4 \ln(x) + 4 - \_C1\, x}$$

```
> Dgl_3 := diff(y(x),x,x) + 5*diff(y(x),x) + 13*y(x) = 0;
```

$$Dgl\_3 := \frac{d^2}{dx^2} y(x) + 5 \left(\frac{d}{dx} y(x) \right) + 13\, y(x) = 0$$

```
> dsolve(Dgl_3, y(x));
```

$$y(x) = \_C1\, e^{-\frac{5}{2} x} \sin\left(\frac{3}{2} \sqrt{3}\, x \right) + \_C2\, e^{-\frac{5}{2} x} \cos\left(\frac{3}{2} \sqrt{3}\, x \right)$$

```
> dsolve({Dgl_3, y(0) = 1, D(y)(0) = 0}, y(x));
```

$$y(x) = \frac{5}{9} \sqrt{3}\, e^{-\frac{5}{2} x} \sin\left(\frac{3}{2} \sqrt{3}\, x \right) + e^{-\frac{5}{2} x} \cos\left(\frac{3}{2} \sqrt{3}\, x \right)$$

```
> Dgl_4 := diff(u(x),x) = u(x) + 2*w(x), diff(v(x),x) = 2*v(x)
  - w(x), diff(w(x),x) = 4*v(x) - 2*w(x);
```

$$Dgl\_4 := \frac{d}{dx} u(x) = u(x) + 2\, w(x),\ \frac{d}{dx} v(x) = 2\, v(x) - w(x),\ \frac{d}{dx} w(x) = 4\, v(x) - 2\, w(x)$$

```
> dsolve({Dgl_4},{u(x),v(x),w(x)});
```

$$\left\{ u(x) = -2\, \_C2 - 2\, \_C3 - 2\, \_C2\, x + e^{x}\, \_C1,\ v(x) = \frac{1}{4}\, \_C2 + \frac{1}{2}\, \_C2\, x + \frac{1}{2}\, \_C3,\ w(x) = \_C2\, x + \_C3 \right\}$$

```
> dsolve({Dgl_4, u(0) = 0, v(0) = 0, w(0) = 1},{u(x),v(x),w(x)}
  );
```

$$\{ u(x) = 2 + 4\, x - 2\, e^{x},\ v(x) = -x,\ w(x) = -2\, x + 1 \}$$

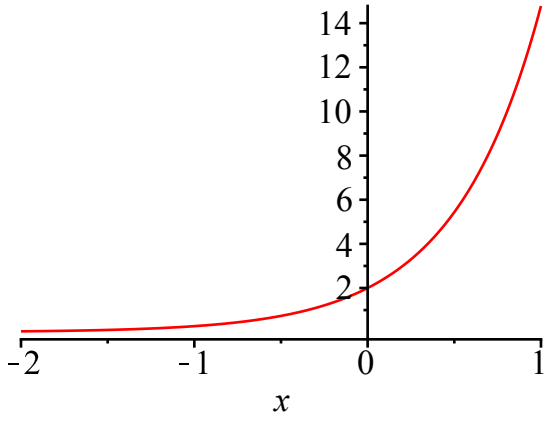
Möchte man die so berechneten Lösungen nun plotten, so funktioniert dies über folgende Befehlskette:

```
> # sol := dsolve({Dgl, Anf}, y(x), Optionen);
  # plot(rhs(sol), x = a .. b, Optionen); # statt "plot" gehen
  hier natürlich auch alle anderen Plot-Befehle
```

Für obige Beispiele kann dies dann so aussehen:

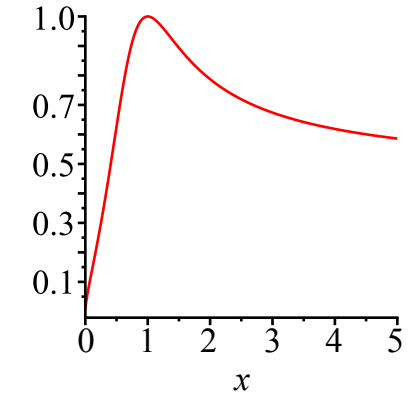
```
> a := 2: sol_1 := dsolve({Dgl_1, y(0) = a}, y(x));
  plot(rhs(sol_1), x = -2 .. 1);
```

$$sol\_1 := y(x) = 2\, e^{2x}$$



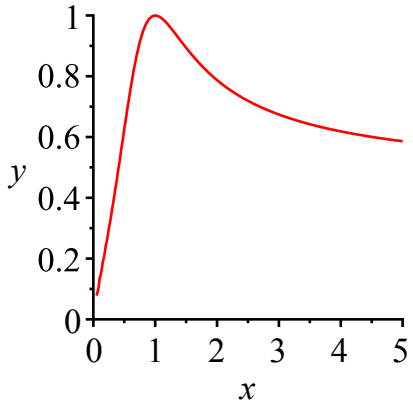
```
> sol_2a := dsolve({Dgl_2, y(1) = 1}, y(x));
plot(rhs(sol_2a), x = 0 .. 5);
```

$$sol\_2a := y(x) = -\frac{\sqrt{-(4\ln(x)+4-5x)\,x}}{4\ln(x)+4-5x}$$



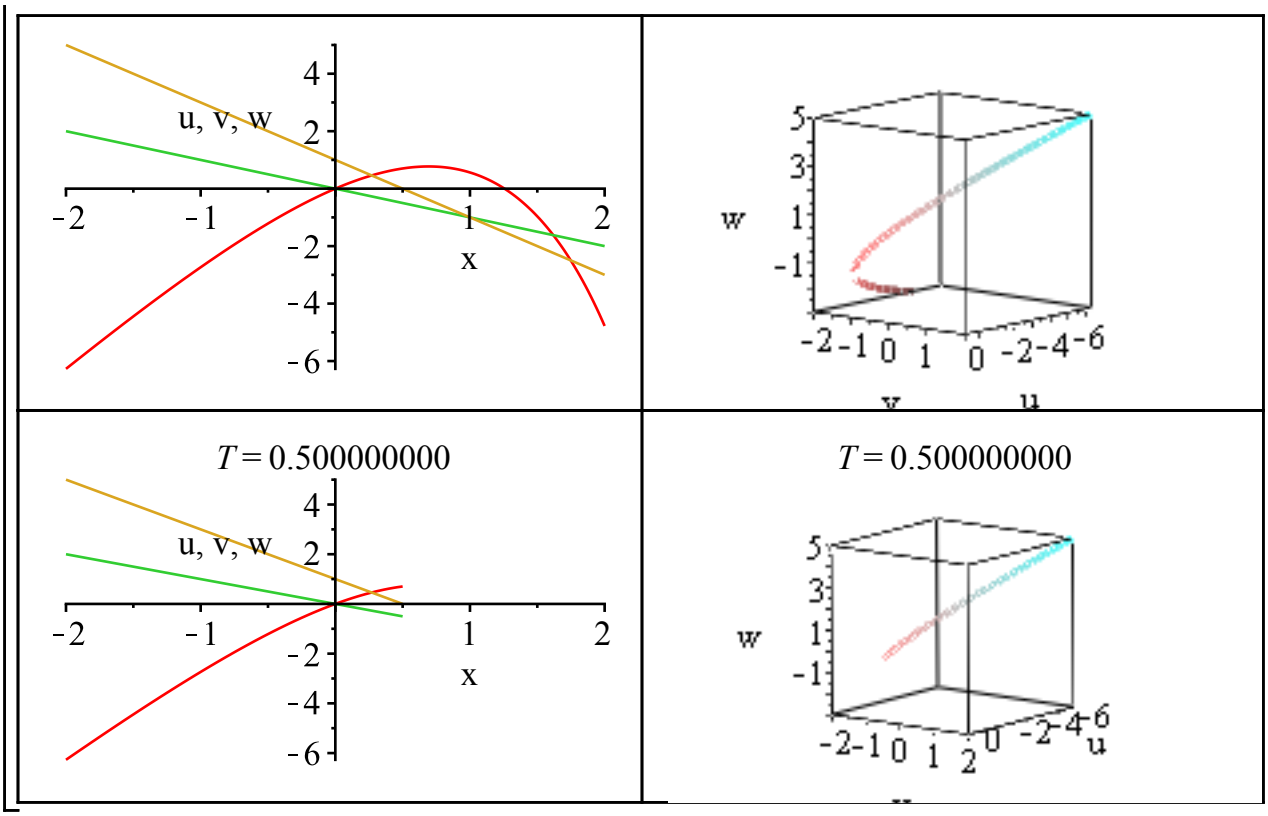

```
> sol_2b := dsolve({Dgl_2, y(1) = 1}, y(x), implicit);
implicitplot(sol_2b, x = 0 .. 5, y = 0 .. 1, numpoints =
10000, view = [0 .. 5, 0 .. 1]); # bei implicitplot ist "rhs"
nicht notwendig
```

$$sol\_2b := \frac{1}{y(x)^2} + \frac{4\ln(x)}{x} + \frac{4}{x} - 5 = 0$$




```
> sol_4 := dsolve({Dgl_4, u(0) = 0, v(0) = 0, w(0) = 1}, {u(x),
v(x),w(x)});
# sol_4 hat hier mehrere Komponenten, auf die wir mit "[ ]"
zugreifen
P1 := plot([rhs(sol_4[1]), rhs(sol_4[2]), rhs(sol_4[3])], x =
-2 .. 2, labels = ["x","u, v, w"]):
P2 := spacecurve([rhs(sol_4[1]), rhs(sol_4[2]), rhs(sol_4[3])
], x = -2 .. 2, axes = boxed, labels = ["u","v","w"],
thickness = 3, orientation = [40,80,0]):
A1 := animate(plot, [[rhs(sol_4[1]), rhs(sol_4[2]), rhs(sol_4
[3])], x = -2 .. T, labels = ["x","u, v, w"]], T = -2 .. 2):
A2 := animate(spacecurve, [[rhs(sol_4[1]), rhs(sol_4[2]), rhs
(sol_4[3])], x = -2 .. T, axes = boxed, labels = ["u","v",
"w"], thickness = 3, orientation = [40,80,0]], T = -2 .. 2):
display(array([ [P1, P2], [A1, A2] ]));
```

$$sol\_4 := \{u(x) = 2 + 4x - 2e^{x},\ v(x) = -x,\ w(x) = -2x + 1\}$$



Grafische Lösungen

In vielen Fällen ist eine analytische Lösung nicht möglich und wir müssen auf numerische Methoden zurückgreifen. Hier bietet Maple mit dem ***DEtools***-Paket Befehle, die eine Differentialgleichung gleichzeitig (numerisch) lösen und plotten. Näher betrachten werden wir hier den ***DEplot***- und den ***DEplot3d***- Befehl:

```
> # DEplot({Dgl}, {y(t)}, t = a .. b, [[Anfbed]], y_1 = c_1 ..
  d_1, y_2 = c_2 .. d_2, Optionen); # geschweifte Klammern nur
  bei zweidim Dgl. nötig
> # DEplot3d({Dgl}, {y(t)}, t = a .. b, [[Anfbed]], y_1 = c_1 .
  . d_1, y_2 = c_2 .. d_2, y_3 = c_3 .. d_3, Optionen);
```

Neben den Lösungen für die Anfangswerte, die in ***Anfbed*** aufgelistet sind, kann Maple auch ein Richtungsfeld plotten (sogar standardmäßig bei ***DEplot***) . Insbesondere hierfür stehen nun wieder verschiedene Optionen zur Verfügung:

animatecurves: Erzeugt eine Animation der Lösungen.
- mögliche Werte: **true, false**
- Default-Wert: **false**

animatefield: Erzeugt eine Animation des Richtungsfeldes (nur im autonomen Fall möglich!).
- mögliche Werte: **true, false**
- Default-Wert: **false**

animate: Erzeugt eine Animation der Lösung und des Richtungsfeldes (falls möglich).
- mögliche Werte: **true, false**
- Default-Wert: **false**

arrows: Definiert die Darstellung (Art der Pfeile) des Richtungsfeldes.
- mögliche Werte: **small, smalltwo, medium, mediumfill, large, curve, comet, line, none** (kein Richtungsfeld)
- Default-Wert: **small** (2D), **none** (3D)

color: Definiert die Farbe des Richtungsfeldes (nicht der Lösungen!).
- mögliche Werte: wie bei ***plot3d***
- Default-Wert: **red** (bis Maple 15)

linecolor: Gibt die Farbe der zu zeichnenden Lösungen an.
- mögliche Werte: wie bei ***plot***
- Default-Wert: **yellow** (bis Maple 15)

numframes: Gibt die Anzahl der Bilder an, die bei einer Animation erstellt werden.
- mögliche Werte: irgendeine natürliche Zahl
- Default-Wert: **25**

numpoints: Gibt die Anzahl der Punkte an, die Maple zum Zeichnen der Lösungen verwendet.
- mögliche Werte: irgendeine natürliche Zahl
- Default-Wert: **49**

obsrange: Gibt an, ob die Lösung abbricht oder fortläuft, wenn sie das angegebene Gebiet verlässt.
- mögliche Werte: **true** (bricht ab), **false** (läuft fort)
- Default-Wert: **true**

scene: Bestimmt die Achsen, die geplottet werden sollen (bei mehrdimensionalen Systemen).
- mögliche Werte: **[a,b]** (bei ***DEplot***) bzw. **[a,b,c]** (bei ***DEplot3d***) mit **a**, **b**, **c** aus {t, y_1(t), y_2(t), y_3(t)} beliebig.
- Default-Wert: **[t, y_1(t)]** (eindim. Dgl.), **[y_1(t), y_2(t)]** (zweidim. Dgl.), **[y_1(t), y_2(t), y_3(t)]** (dreidim. Dgl.)

size: Bestimmt die Größe der Pfeile im Richtungsfeld.
- mögliche Werte: irgendeine positive Zahl
- Default-Wert: **1.0**

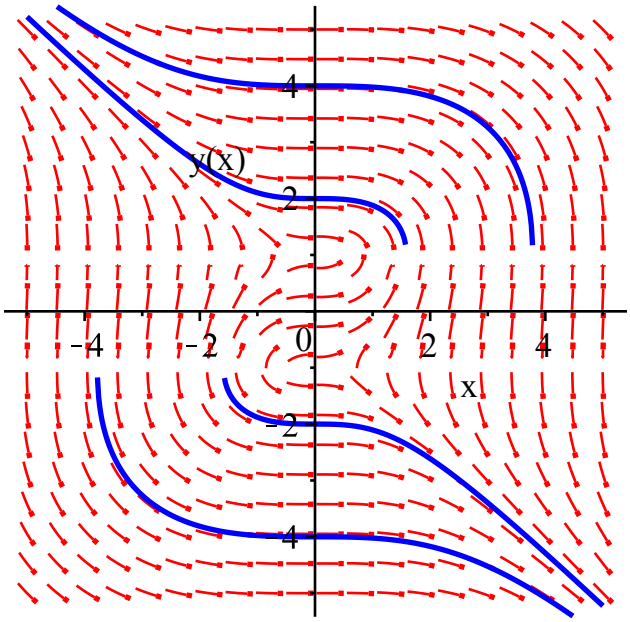
stepsize: Definiert die Schrittweite, die bei der Berechnung der Lösungen benutzt werden soll.
- mögliche Werte: irgendeine positive Zahl
- Default-Wert: 1/48 * (b - a) (in 2D), 1/20*(b - a) (in 3D)

Hier drei Beispiele:

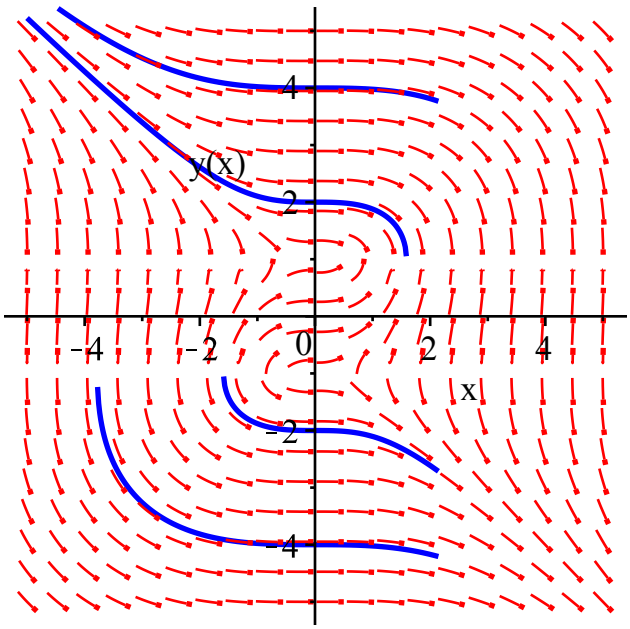
```
> Diffgl := diff(y(x),x) = x^2/(1-y(x)^2);
```

$$Diffgl := \frac{d}{dx} y(x) = \frac{x^2}{1 - y(x)^2}$$

```
> DEplot(Diffgl, y(x), x = -5 .. 5, [[y(0) = 4], [y(0) = 2], [y
  (0) = -4], [y(0) = -2]], y = -5 .. 5, thickness = 2, arrows =
  curve, color = red, linecolor = blue, obsrange = false,
  numpoints = 500); # ohne Animation
```



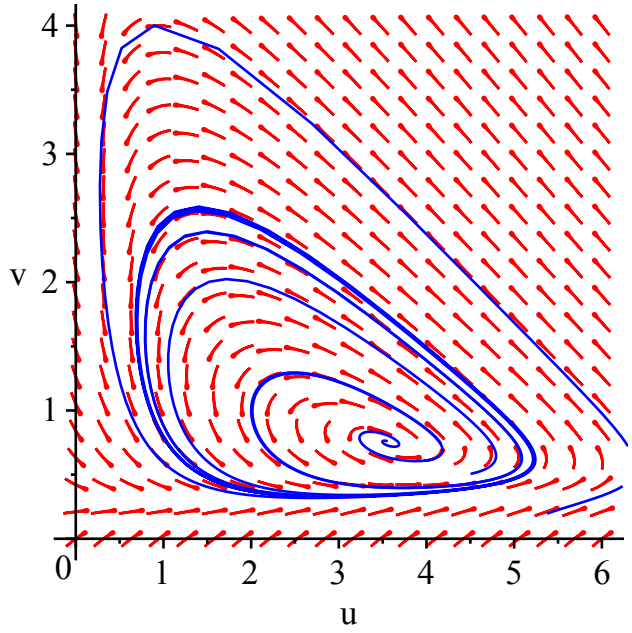

```
> DEplot(Diffgl, y(x), x = -5 .. 5, [[0,4], [0,2], [0,-2], [0,
  -4]], y = -5 .. 5, thickness = 2, arrows = curve, color =
  red, linecolor = blue, obsrange = false, numpoints = 540,
  animatecurves = true, numframes = 50); # mit Animation,
  verkürzte Schreibweise für Anfangswerte
```



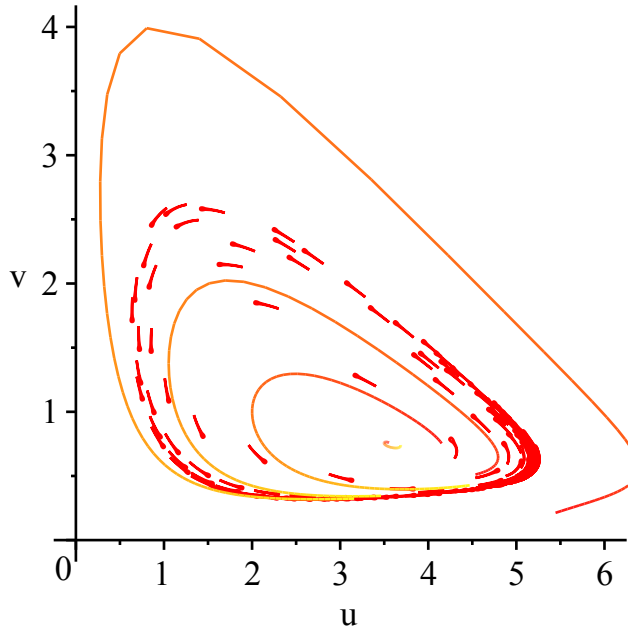
```
> Dglsys := diff(u(t),t) = 2-u(t)*v(t)^2, diff(v(t),t) = 1-4*v
  (t) + u(t)*v(t)^2;
```

$$Dglsys := \frac{d}{dt}u(t) = 2 - u(t)\,v(t)^2,\ \frac{d}{dt}v(t) = 1 - 4\,v(t) + u(t)\,v(t)^2$$

```
> DEplot({Dglsys}, {u(t),v(t)}, t = -1.5 .. 20, [[u(0) = 3.5, v
  (0) = 0.75],[u(0) = 2,v(0) = 2],[u(0) = 2, v(0) = 1],[u(0) =
  1, v(0) = 4]], u = 0 .. 6, v = 0 .. 4, thickness = 1, arrows
  = comet, color = red, linecolor = blue, obsrange = false,
  numpoints = 500); # ohne Animation
```



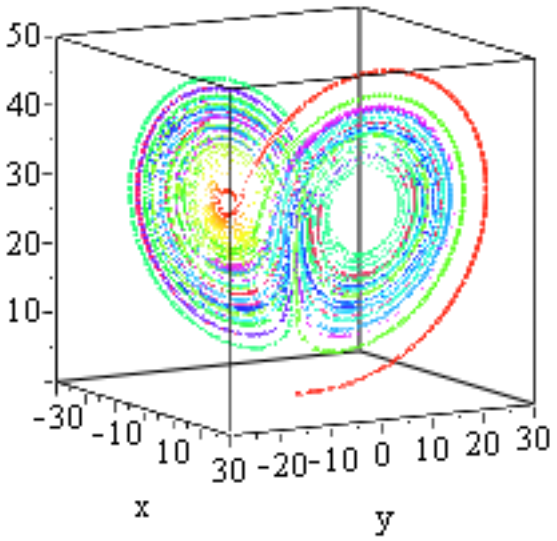

```
> DEplot({Dglsys}, {u(t),v(t)}, t = -1.5 .. 20, [[0,3.5,0.75],
  [0,2,2],[0,2,1],[0,1,4]], u = 0 .. 6, v = 0 .. 4, thickness =
  1, arrows = comet, color = red, linecolor = t/2, obsrange =
  false, numpoints = 540, animate = true, numframes = 50); #
  mit Animation, verkürzte Schreibweise für Anfangswerte
```



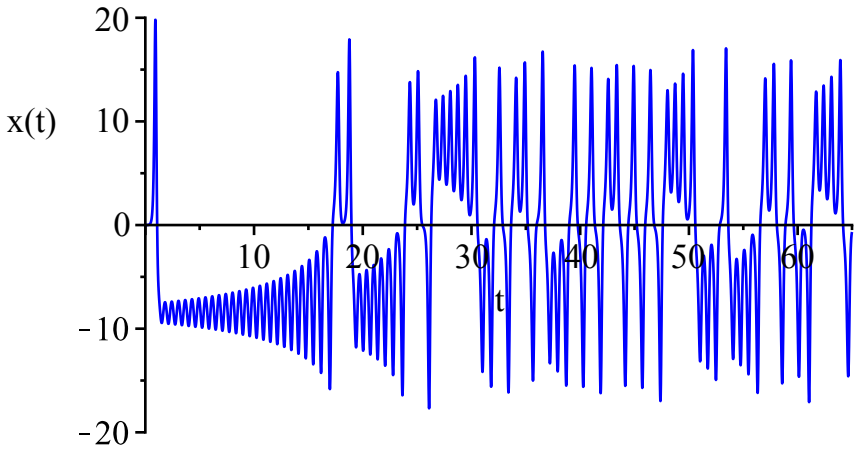
```
> ODEsys := diff(x(t),t) = -10*x(t) + 10*y(t), diff(y(t),t) =
  28*x(t) - y(t) - x(t)*z(t), diff(z(t),t) = -8/3*z(t) + x(t)*y
  (t);
```

$$ODEsys := \frac{d}{dt}x(t) = -10\,x(t) + 10\,y(t),\ \frac{d}{dt}y(t) = 28\,x(t) - y(t) - x(t)\,z(t),\ \frac{d}{dt}z(t) = -\frac{8}{3}\,z(t) + x(t)\,y(t)$$

```
> DEplot3d({ODEsys}, {x(t),y(t),z(t)}, t = 0 .. 65, [[x(0) =
  0.001, y(0) = 0.001, z(0) = 0.001]], x = -30 .. 30, y = -30 .
  . 30, z = 0 .. 50, thickness = 2, color = red, linecolor =
  t/2, obsrange = false, numpoints = 20000, orientation = [-30,
  80,0]);
```




```
> DEplot({ODEsys}, {x(t),y(t),z(t)}, t = 0 .. 65, [[x(0) =
  0.001, y(0) = 0.001, z(0) = 0.001]], x = -20 .. 20, thickness
  = 1, color = red, linecolor = blue, obsrange = false,
  numpoints = 10000, scene = [t,x(t)]); # nur Lösung x(t) mit
  "scene", beachte, dass hier "DEplot" und nicht "DEplot3d"
  stehen muss
```



Ausgewählte Beispiele aus Übung und Tutorium

1. Tutorium, Beispiel 4, Thema: Riccati-Differentialgleichung

```
> restart; with(DEtools): with(plots):
> Dgl:= diff(y(x),x) = f(x,y(x));
```

$$Dgl := \frac{d}{dx} y(x) = f(x, y(x))$$

```
> f := (x,y) -> -3*x*y + x*y^2 + 2*x;
```

$$f := (x, y) \to -3\,x\,y + x\,y^2 + 2\,x$$

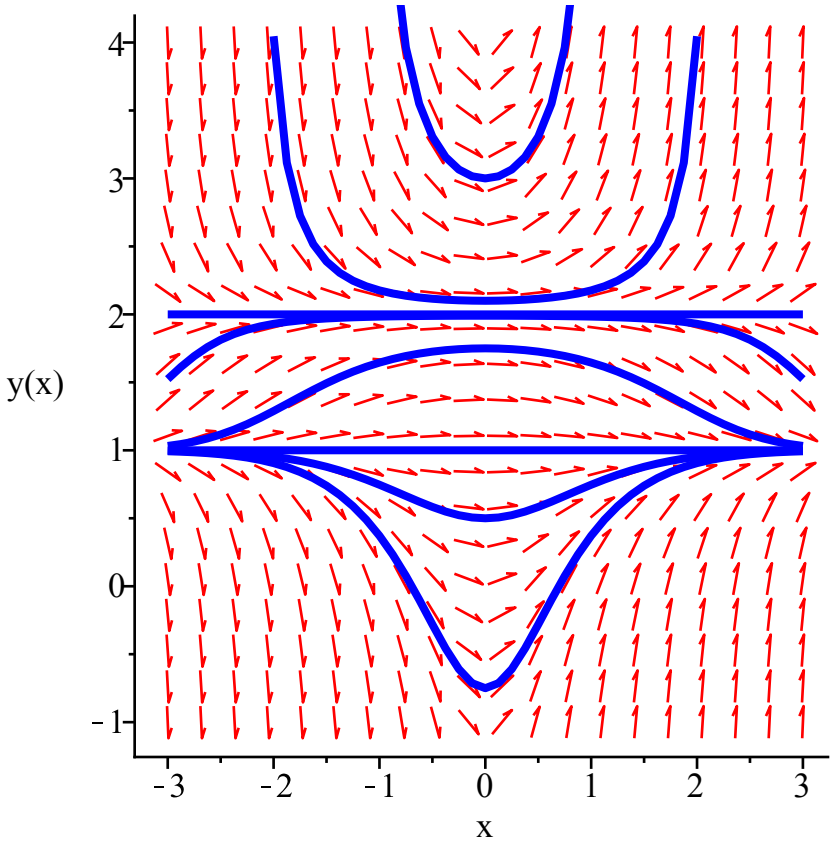
```
> dsolve(Dgl, y(x));
```

$$y(x) = \frac{-2 + e^{\frac{1}{2}x^2}\_C1}{e^{\frac{1}{2}x^2}\_C1 - 1}$$

```
> AnfBed := [y(0) = 1], [y(0) = 2], [y(0) = 1.75], [y(0) =
  1.99], [y(0) = 2.1], [y(0) = 3], [y(0) = 0.5], [y(0) = -0.75]
  ;
```

$$AnfBed := [y(0) = 1], [y(0) = 2], [y(0) = 1.75], [y(0) = 1.99], [y(0) = 2.1], [y(0) = 3], [y(0) = 0.5], [y(0) = -0.75]$$

```
> DEplot(Dgl, y(x), x = -3 .. 3, y = -1 .. 4, [AnfBed],
  linecolor = blue, color = red, arrows = small, axes = frame);
```



1. Übung, Aufgabe 5, Thema: Bewegung mit Luftwiderstand

Horizontale Bewegung (am Beispiel eines Menschen):

```
> restart; with(DEtools): with(plots):
> Dgl := diff(v(t),t) = f(t,v(t));
```

$$Dgl := \frac{d}{dt} v(t) = f(t, v(t))$$

```
> f := (t,v) -> -k/m*v^2;
```

$$f := (t, v) \rightarrow -\frac{k\, v^2}{m}$$

```
> k := 1/2*c_W*rho*A;
```

$$k := \frac{1}{2} c\_W \rho A$$

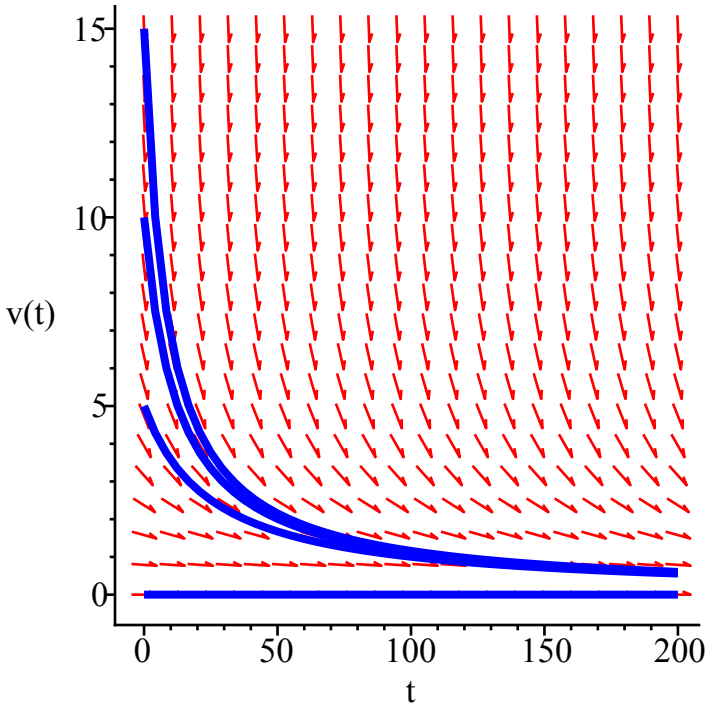
```
> c_W := 0.78; A := 1.7; rho := 1.2; m := 100;
```

$$c\_W := 0.78$$
$$A := 1.7$$
$$\rho := 1.2$$
$$m := 100$$

```
> AnfBed := [v(0) = 0], [v(0) = 5], [v(0) = 10], [v(0) = 15];
```

$$AnfBed := [v(0) = 0], [v(0) = 5], [v(0) = 10], [v(0) = 15]$$

```
> DEplot(Dgl, v(t), t = 0 .. 200, v = 0 .. 15, [AnfBed],
  linecolor = blue, color = red, arrows = small, axes = frame);
```



Vertikale Bewegung:

a) Freier Fall eines Menschen:

```
> restart; with(DEtools): with(plots):
> Dgl_1 := diff(v(t),t) = f_1(t,v(t));
```

$$Dgl\_1 := \frac{d}{dt} v(t) = f\_1(t, v(t))$$

```
> f_1 := (t,v) -> -k_1/m*v^2 + g;
```

$$f\_1 := (t, v) \rightarrow -\frac{k\_1 v^2}{m} + g$$

```
> k_1 := 1/2*c_W1*rho*A1;
```

$$k\_1 := \frac{1}{2} c\_W1 \rho A1$$

```
> c_W1 := 0.78; A1 := 1.7; rho := 1.2; m := 100; g := 9.81;
  v_inf1 := sqrt(m*g/k_1);
```

$$c\_W1 := 0.78$$

$$A1 := 1.7$$

$$\rho := 1.2$$

$$m := 100$$

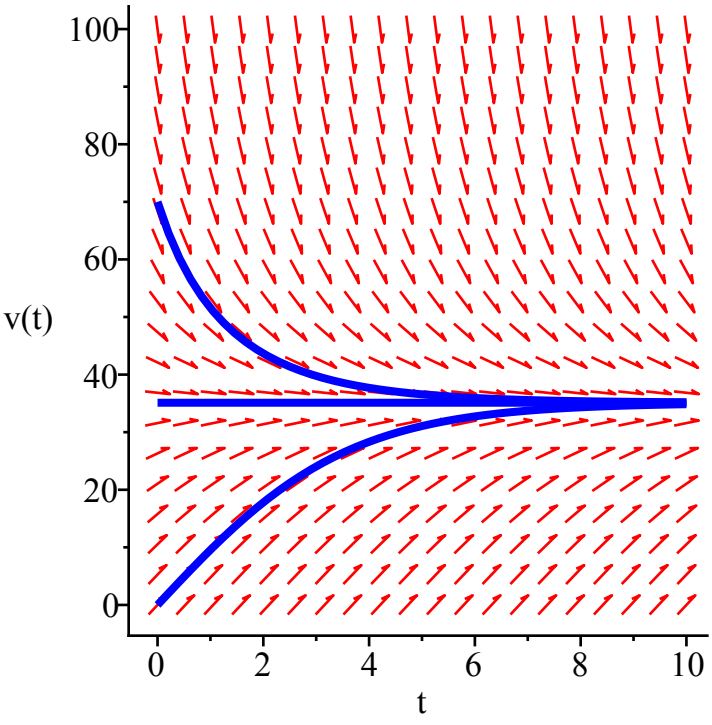
$$g := 9.81$$

$$v\_inf1 := 35.11455074$$

```
> AnfBed := [v(0) = 0], [v(0) = v_inf1], [v(0) = 70];
```

$$AnfBed := [v(0) = 0], [v(0) = 35.11455074], [v(0) = 70]$$

```
> DEplot(Dgl_1, v(t), t = 0 .. 10, v = 0 .. 100, [AnfBed],
  linecolor = blue, color = red, arrows = small, axes = frame);
```



b) Fallschirmsprung

```
> Dgl_2 := diff(v(t),t) = f_2(t,v(t)); f_2 := (t,v) -> -k_2/m*
  v^2 + g;
```

$$Dgl\_2 := \frac{d}{dt} v(t) = f\_2(t, v(t))$$

$$f\_2 := (t, v) \rightarrow -\frac{k\_2\, v^2}{m} + g$$

```
> k_2 := 1/2*c_W2*rho*A2;
```

$$k\_2 := 0.6000000000\, c\_W2\, A2$$

```
> c_W2 := 1.33; A2 := 40; rho := 1.2; m := 100; g := 9.81;
  v_inf2 := sqrt(m*g/k_2);
```

$$c\_W2 := 1.33$$

$$A2 := 40$$

$$\rho := 1.2$$

$$m := 100$$

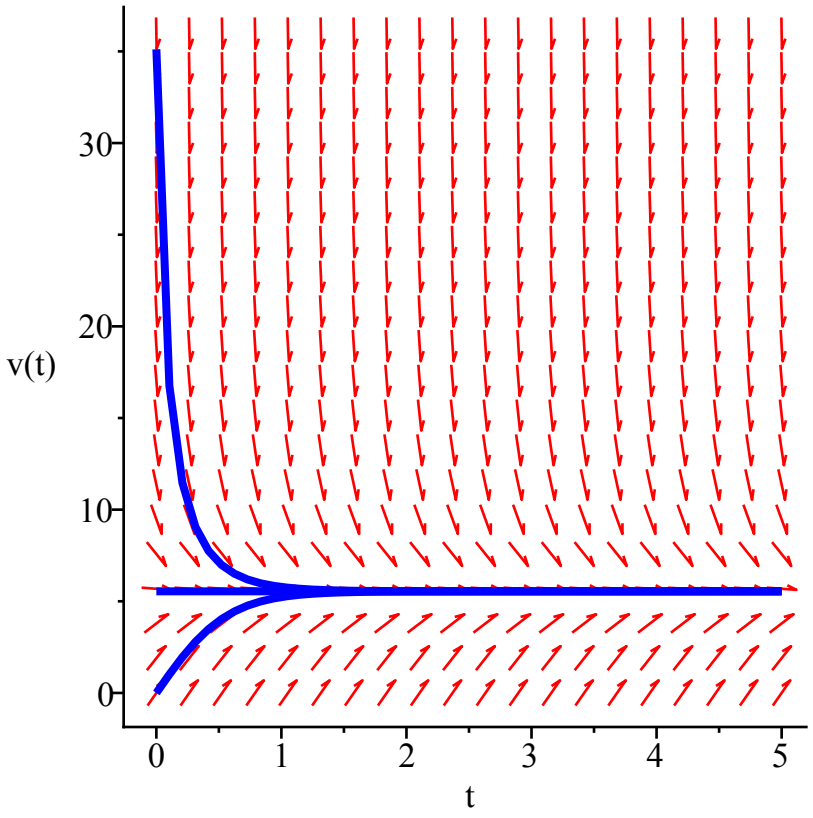
$$g := 9.81$$

$$v\_inf2 := 5.543742663$$

```
> AnfBed := [v(0) = v_inf1], [v(0) = v_inf2], [v(0) = 0];
```

$$AnfBed := [v(0) = 35.11455074], [v(0) = 5.543742663], [v(0) = 0]$$

```
> DEplot(Dgl_2, v(t), t = 0 .. 5, v = 0 .. 36, [AnfBed],
  linecolor = blue, color = red, arrows = small, axes = frame);
```



2. Tutorium, Beispiel 4, Thema: Exakte Differentialgleichung

```
> restart; with(DEtools): with(plots):
> Dgl := p(x,y(x)) + q(x,y(x))*diff(y(x),x) = 0;
```

$$Dgl := p(x, y(x)) + q(x, y(x)) \left(\frac{d}{dx} y(x) \right) = 0$$

```
> p := (x,y) -> cos(x) + sin(x); q := (x,y) -> 2*y*sin(x);
```

$$p := (x, y) \rightarrow \cos(x) + \sin(x)$$

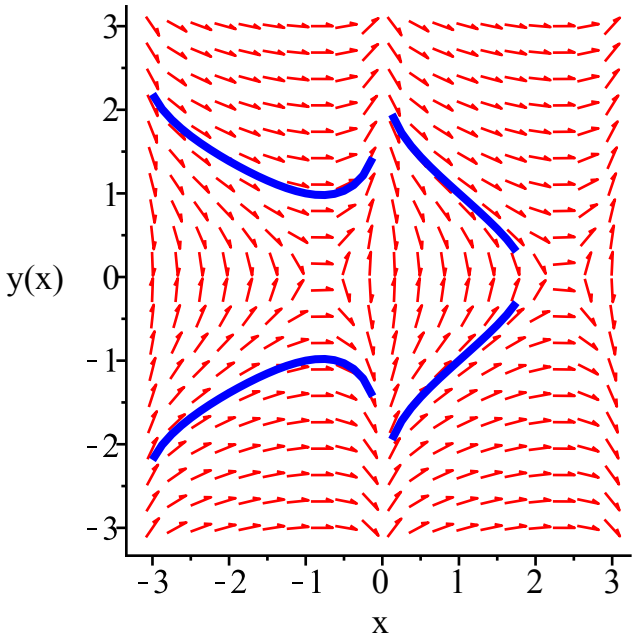
$$q := (x, y) \rightarrow 2\, y \sin(x)$$

```
> dsolve(Dgl, y(x), implicit); dsolve(Dgl, y(x), explicit);
```

$$y(x)^2 + \ln(\sin(x)) + x - \_C1 = 0$$

$$y(x) = \sqrt{-\ln(\sin(x)) - x + \_C1}\,,\ y(x) = -\sqrt{-\ln(\sin(x)) - x + \_C1}$$

```
> DEplot(Dgl, y(x), x = -3 .. 3, y = -3 .. 3, [[-1,1],[-1,-1],
  [1,-1],[1,1]], linecolor = blue, color = red, arrows = small,
  axes = frame);
```



2. Übung, Aufgabe 9, Thema: Exakte Differentialgleichung

```
> restart; with(DEtools): with(plots):
> Dgl:= p(x,y(x)) + q(x,y(x))*diff(y(x),x) = 0;
```

$$Dgl := p(x, y(x)) + q(x, y(x)) \left(\frac{d}{dx} y(x) \right) = 0$$

```
> p := (x,y) -> tan(x*y) + x*y; q := (x,y) -> x^2;
```

$$p := (x, y) \rightarrow \tan(x\, y) + x\, y$$

$$q := (x, y) \rightarrow x^2$$

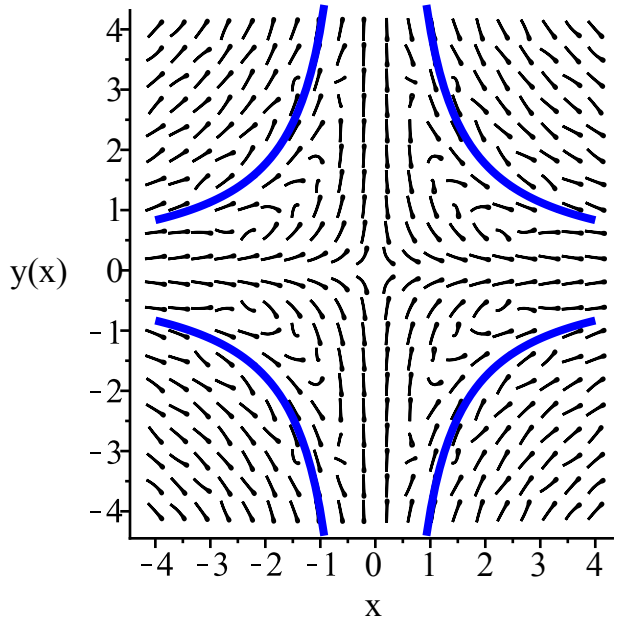
```
> dsolve(Dgl, y(x), implicit);
```

$$y(x)^2 + \ln(\sin(x)) + x - \_C1 = 0$$

```
> AnfBed := [y(1) = 4], [y(-1) = 4], [y(-1) = -4], [y(1) = -4];
```

$$AnfBed := [y(1) = 4], [y(-1) = 4], [y(-1) = -4], [y(1) = -4]$$

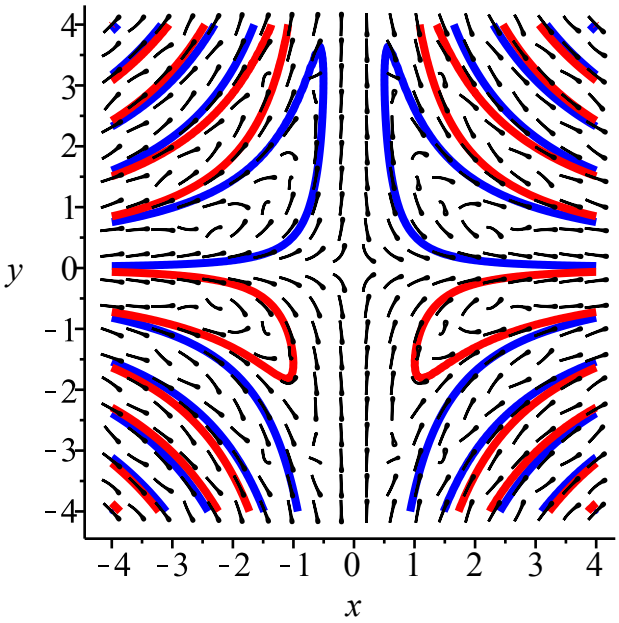
```
> DEplot(Dgl, y(x), x = -4 .. 4, y = -4 .. 4, [AnfBed],
  linecolor = blue, color = black, axes = frame, arrows =
  comet, numpoints = 100);
```




```
> f := (x,y) -> x*sin(x*y);
```

$$f := (x, y) \rightarrow x \sin(x\,y)$$

```
> P1 := implicitplot(f(x,y) = 0.5, x = -4 .. 4, y = -4 .. 4,
  axes = frame, color = blue, numpoints = 5000, thickness = 3):
> P2 := implicitplot(f(x,y) = -1.0, x = -4 .. 4, y = -4 .. 4,
  axes = frame, color = red, numpoints = 5000, thickness = 3):
> P3 := DEplot(Dgl, y(x), x = -4 .. 4, y = -4 .. 4, color =
  black, axes = frame, arrows = comet):
> display(P1,P2,P3);
```



3. Übung, Aufgabe 11, Thema: Implizite Differentialgleichung

```
> restart; with(DEtools): with(plots):
> Dgl := y(x) = 1/2*x^2 - x*diff(y(x),x) + diff(y(x),x)^2;
```

$$Dgl := y(x) = \frac{1}{2}x^2 - x\left(\frac{d}{dx}y(x)\right) + \left(\frac{d}{dx}y(x)\right)^2$$

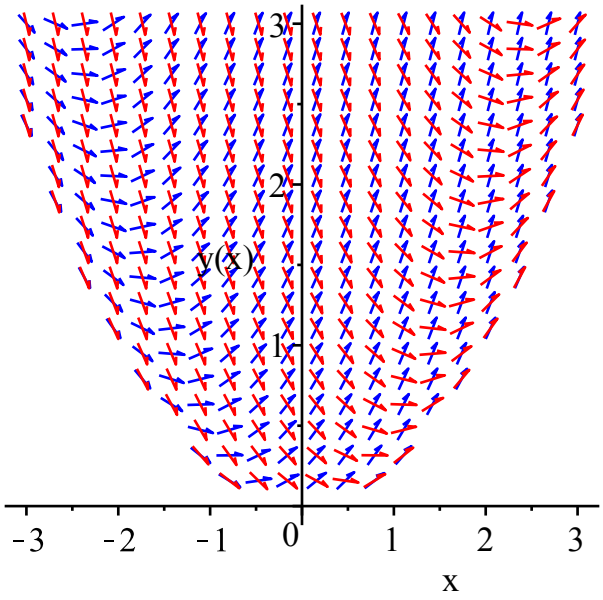
```
> dsolve(Dgl, y(x));
```

$$y(x) = \frac{1}{4}x^2, y(x) = -\frac{1}{2}\frac{\_C1\left(-x\_C1+\sqrt{2}\right)x-1}{\_C1^2}, y(x) = -\frac{1}{2}\frac{-\_C1\left(x\_C1+\sqrt{2}\right)x-1}{\_C1^2}, y(x) = \frac{1}{2}x\left(x-\sqrt{2}\,\_C1\right)+\frac{1}{2}\,\_C1^2, y(x) = \frac{1}{2}x\left(x+\sqrt{2}\,\_C1\right)+\frac{1}{2}\,\_C1^2$$

```
> S := solve(Dgl, diff(y(x),x));
```

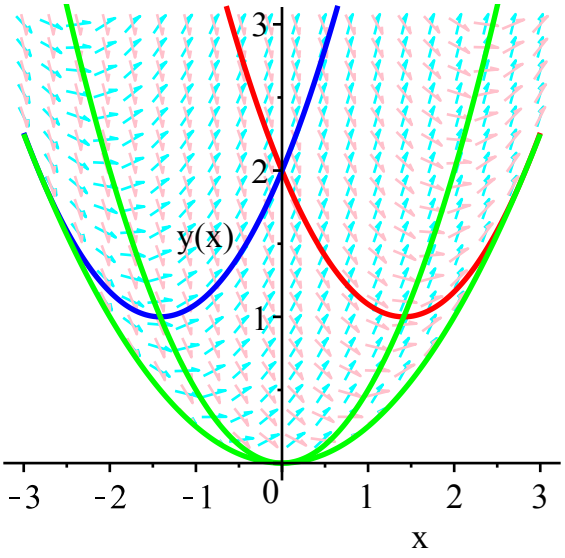
$$S := \frac{1}{2}x + \frac{1}{2}\sqrt{-x^2+4y(x)}, \frac{1}{2}x - \frac{1}{2}\sqrt{-x^2+4y(x)}$$

```
> P1 := DEplot(diff(y(x),x) = S[1], y(x), x = -3 .. 3, y = 0 ..
  3, color = blue):
> P2 := DEplot(diff(y(x),x) = S[2], y(x), x = -3 .. 3, y = 0 ..
  3, color = red):
> display(P1,P2);
```




```
> P3 := DEplot(diff(y(x),x) = S[1], y(x), x = -3 .. 3, y = 0 ..
  3, color = cyan):
> P4 := DEplot(diff(y(x),x) = S[2], y(x), x = -3 .. 3, y = 0 ..
  3, color = pink):
> P5 := plot(1/2*x^2 - sqrt(2)*x + 2, x = -3 .. 3, y = 0 .. 3,
  color = red, thickness = 2):
> P6 := plot(1/2*x^2 + sqrt(2)*x + 2, x = -3 .. 3, y = 0 .. 3,
  color = blue, thickness = 2):
```

```
> P7 := plot(1/4*x^2, x = -3 .. 3, y = 0 .. 3, color = green,
  thickness = 2):
> P8 := plot(1/2*x^2, x = -3 .. 3, y = 0 .. 3, color = green,
  thickness = 2):
> display(P3,P4,P5,P6,P7,P8);
```

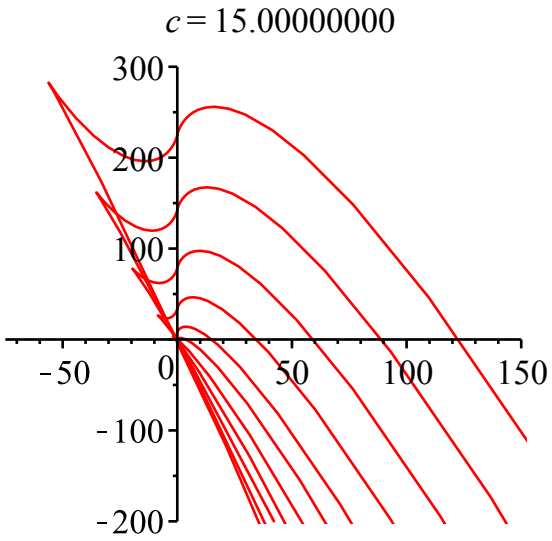


3. Übung, Aufgabe 12 c), Thema: Implizite Differentialgleichung - "Der Flügel"

```
> restart; with(DEtools): with(plots):
> Dgl := x^2*exp(diff(y(x),x)) + x*diff(y(x),x) - y(x) = 0;
```

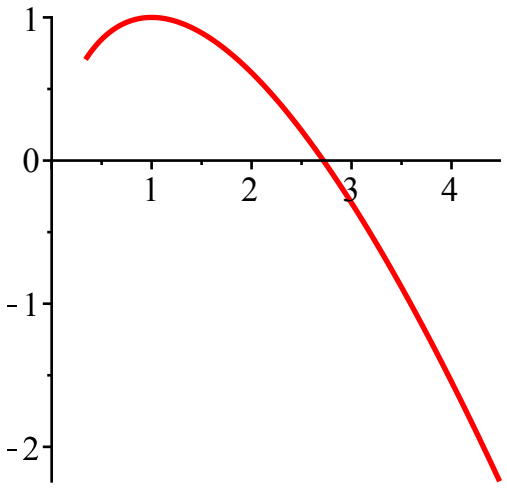
$$Dgl := x^2 \, e^{\frac{d}{dx} y(x)} + x \left(\frac{d}{dx} y(x) \right) - y(x) = 0$$

```
> animate(plot,[[c*exp(-1/2*t) + exp(-t), c^2 + c*(2 + t)*exp
  (-1/2*t) + (1 + t)*exp(-t), t = -10 .. 100]], c = -15 .. 15,
  view = [-75 .. 150,-200 .. 300], trace = 10, frames = 50);
```



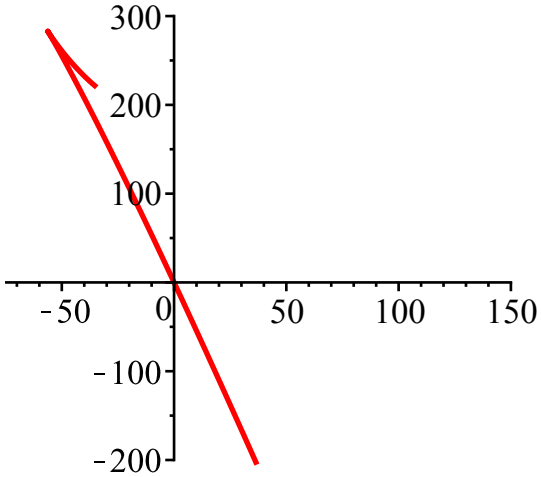
```
> animate(plot,[[exp(-t), (1 + t)*exp(-t), t = -1.5 .. T],
  thickness = 2], T = -1.5 .. 10, frames = 50);
```

$$T = 1.081632653$$




```
> animate(plot,[[-15*exp(-1/2*t) + exp(-t), 225 - 15*(2+t)*exp
  (-1/2*t) + (1 + t)*exp(-t), t = -6 .. T], thickness = 2], T =
  -6 .. 10, view = [-75 .. 150,-200 .. 300], frames = 50);
```

$$T = -2.081632653$$



5. Tutorium, Beispiel 3, Thema: Picarditeration

```
> restart; with(plots): with(DEtools):
> f := (x,y,z) -> (-z,y);
```

$$f := (x, y, z) \rightarrow (-z, y)$$

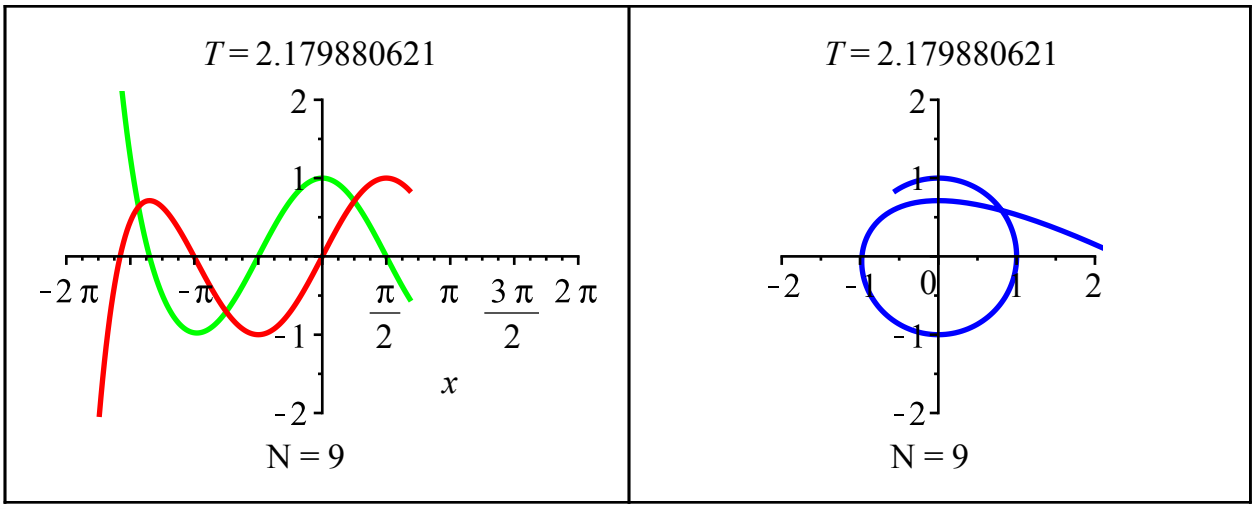
```
> Dgls := diff(y(x),x) = 0*y(x) - z(x), diff(z(x),x) = y(x) +
  0*z(x);
```

$$Dgls := \frac{d}{dx} y(x) = -z(x), \frac{d}{dx} z(x) = y(x)$$

```
> dsolve([Dgls, y(0) = 1, z(0) = 0]);
```

$$\{y(x) = \cos(x), z(x) = \sin(x)\}$$

```
> for N from 0 to 11 do
  animation2[N] := animate(plot,[[add( 1/(2*k)!*(-1)^k*x^(2*k),
  k = 0 .. floor(N/2)), add( 1/(2*k+1)!*(-1)^k*x^(2*k+1), k = 0
  .. floor((N-1)/2)), x = -2*Pi .. T], color = blue, thickness
  = 2, view = [-2 .. 2 , -2 .. 2], scaling = constrained,
  caption = typeset("N = ",N)], T = -2*Pi .. 2*Pi, frames = 50)
  :
  animation1[N] := animate(plot,[[add( 1/(2*k)!*(-1)^k*x^(2*k),
  k = 0 .. floor(N/2)), add( 1/(2*k+1)!*(-1)^k*x^(2*k+1), k = 0
  .. floor((N-1)/2))], x = -2*Pi .. T, color = [green, red],
  thickness = 2, view = [-2*Pi .. 2*Pi , -2 .. 2], caption =
  typeset("N = ",N)], T = -2*Pi .. 2*Pi, frames = 50):
  od:
> an1 := display([seq(animation1[N], N = 0 .. 11)], insequence
  = true):
> an2 := display([seq(animation2[N], N = 0 .. 11)], insequence
  = true):
> display(array([an1,an2]));
```



5. Übung, Aufgabe 24 a) und b), Thema: Differentialgleichungssysteme

a)

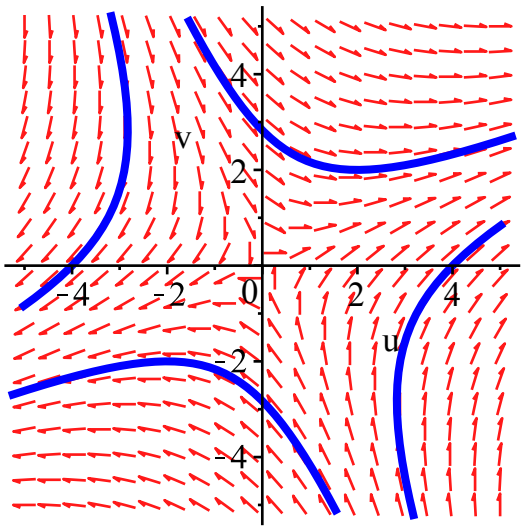
```
> restart; with(plots): with(DEtools):
> Dgls := diff(u(x),x) = u(x) + v(x), diff(v(x),x) = u(x) - v
  (x);
```

$$Dgls := \frac{d}{dx}u(x) = u(x) + v(x),\ \frac{d}{dx}v(x) = u(x) - v(x)$$

```
> dsolve([Dgls, u(0) = 1, v(0) = 0]);
```

$$\left\{u(x) = \left(\frac{1}{2} + \frac{1}{4}\sqrt{2}\right)e^{\sqrt{2}x} + \left(\frac{1}{2} - \frac{1}{4}\sqrt{2}\right)e^{-\sqrt{2}x},\ v(x) = \left(\frac{1}{2} + \frac{1}{4}\sqrt{2}\right)\sqrt{2}\,e^{\sqrt{2}x} - \left(\frac{1}{2} - \frac{1}{4}\sqrt{2}\right)\sqrt{2}\,e^{-\sqrt{2}x} - \left(\frac{1}{2} + \frac{1}{4}\sqrt{2}\right)e^{\sqrt{2}x} - \left(\frac{1}{2} - \frac{1}{4}\sqrt{2}\right)e^{-\sqrt{2}x}\right\}$$

```
> DEplot([Dgls], [u(x), v(x)], x = -1.5 .. 1, [[0,-2,-2],[0,2,
  2],[0,-4,0],[0,4,0]], u = -5 .. 5, v = -5 .. 5, linecolor =
  blue);
```



b)

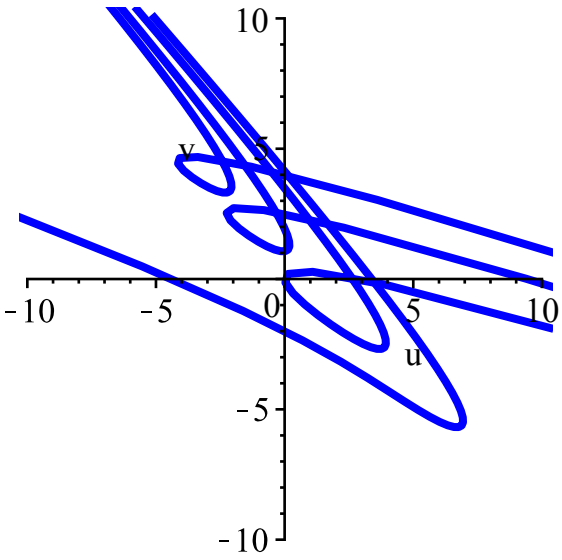
```
> restart; with(plots): with(DEtools):
> Dgls := diff(u(x),x) = 3*u(x) + 3*v(x) + x, diff(v(x),x) = -u
  (x) - v(x) + 1;
```

$$Dgls := \frac{d}{dx}u(x) = 3u(x) + 3v(x) + x,\ \frac{d}{dx}v(x) = -u(x) - v(x) + 1$$

```
> dsolve([Dgls, u(0) = 0, v(0) = 0]);
```

$$\left\{u(x) = -\frac{1}{4}x^2 + \frac{9}{8}e^{2x} - \frac{9}{4}x - \frac{9}{8},\ v(x) = \frac{7}{4}x - \frac{3}{8}e^{2x} + \frac{3}{8} + \frac{1}{4}x^2\right\}$$

```
> DEplot([Dgls], [u(x), v(x)], x = -15 .. 2, [[0,0,-2],[0,0,
  2.5],[0,0,0],[0,0,4]], u = -10 .. 10, v = -10 .. 10,
  linecolor = blue);
```



6. Tutorium, Beispiele 1, 2 und 3, Thema: Dgl.-systeme

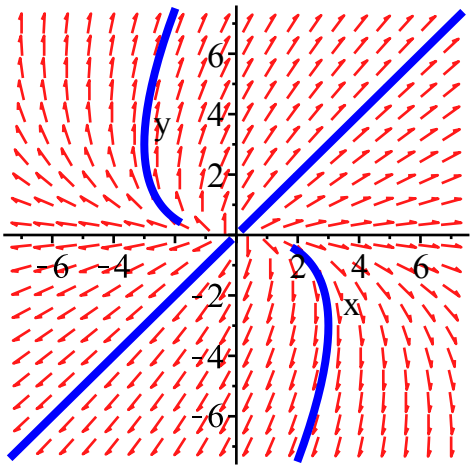
```
> restart; with(plots): with(DEtools):
> DG := diff(x(t),t) = x(t) + y(t), diff(y(t),t) = 2*y(t);
```

$$DG := \frac{d}{dt}x(t) = x(t) + y(t),\ \frac{d}{dt}y(t) = 2\,y(t)$$

```
> dsolve({DG, x(0) = 1, y(0) = 1}, {x(t),y(t)});
```

$$\{x(t) = e^{2t}, y(t) = e^{2t}\}$$

```
> DEplot({DG}, {x(t),y(t)}, t = -1 .. 1, [[0,1,1],[0,3,-3],[0,
  -3,3],[0,-1,-1]], x = -7 .. 7, y = -7 .. 7, linecolor = blue)
  ;
```



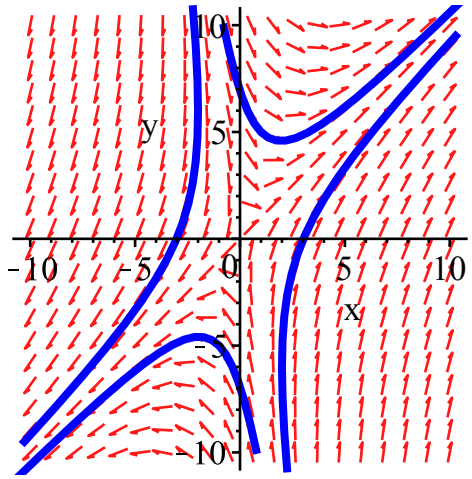

```
> restart; with(plots): with(DEtools):
> DG := diff(x(t),t) = 3*x(t) + y(t), diff(y(t),t) = 7*x(t) -
  3*y(t);
```

$$DG := \frac{d}{dt}x(t) = 3\,x(t) + y(t),\ \frac{d}{dt}y(t) = 7\,x(t) - 3\,y(t)$$

```
> dsolve({DG, x(0) = 2, y(0) = -6}, {x(t),y(t)});
```

$$\{x(t) = e^{4t} + e^{-4t}, y(t) = e^{4t} - 7\,e^{-4t}\}$$

```
> DEplot({DG}, {x(t),y(t)}, t = -1 .. 1, [[0,2,-6],[0,-1,-5],
  [0,1,5],[0,-2,6]], x = -10 .. 10, y = -10 .. 10, linecolor =
  blue);
```



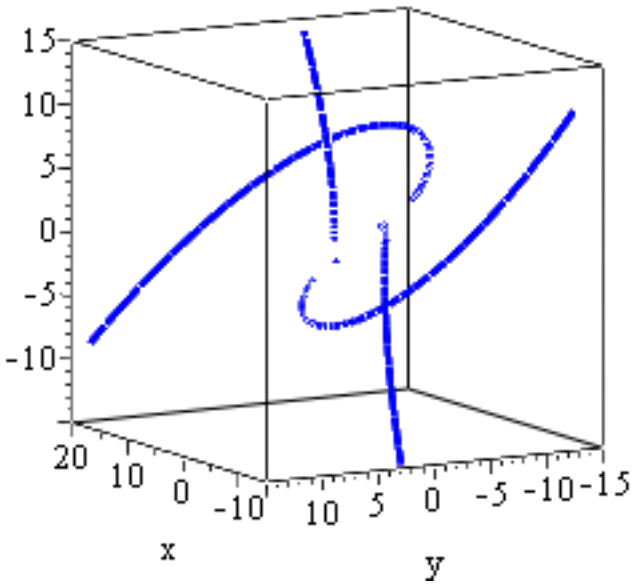
```
> restart; with(plots): with(DEtools):
> DG := diff(x(t),t) = 2*y(t) + 4*z(t), diff(y(t),t) = 4*x(t) -
  2*y(t) - 5*z(t), diff(z(t),t) = -4*x(t) + 4*y(t) + 8*z(t);
```

$$DG := \frac{d}{dt}x(t) = 2y(t) + 4z(t),\ \frac{d}{dt}y(t) = 4x(t) - 2y(t) - 5z(t),\ \frac{d}{dt}z(t) = -4x(t) + 4y(t) + 8z(t)$$

```
> dsolve({DG}, {x(t),y(t),z(t)});
```

$$\left\{x(t) = e^{2t}(\_C1 + \_C2\,t + \_C3\,t^2),\ y(t) = \frac{1}{2}e^{2t}(2\_C1 + 2\_C2\,t - 3\_C2 + 2\_C3\,t^2 - 6\_C3\,t + 4\_C3),\ z(t) = e^{2t}(\_C2 + 2\_C3\,t - \_C3)\right\}$$

```
> DEplot3d({DG}, {x(t),y(t),z(t)}, t = -1 .. 1, [[0,1,1,1],[0,
  -2,-2,-2],[0,-5,5,-5],[0,8,-8,8]], linecolor = blue,
  orientation = [150,80,0], x = -15 .. 20, y = -15 .. 15, z =
  -15 .. 15);
```



weitere Beispiele folgen ...