Mathematik mit Mathematica

Praktikum im Wintersemester 2021/22 an der TU Braunschweig
betreut von Prof. Dr. Michael Herrmann

Tutorium 03: Gleichungen

Vorbemerkung : Mathematica stellt Lösungen von Gleichungen in einer auf den ersten Blick sehr komplizierten Form bereit , nämlich als Liste formaler Ersetzungsregeln. Dies hat einige Nachteile, aber auch sehr viele Vorteile.

```
In[∘]:= (* einfache Beispiele *)
        Solve[x == 1, x]
        Solve[x * x == 1, x]

Out[∘]= {{x → 1}}

Out[∘]= {{x → -1}, {x → 1}}
```

Das Ergebnis besteht im zweiten Fall nicht etwa aus den Zahlen - 1 und + 1, sondern aus den Ersetzungsregeln x→ -1 und x → +1.

Ausdrücke und Ersetzungsregeln

```
In[∘]:= (* ein Ausdruck *)
        expr = t * x1 ^ 2 + x2 * Exp[t]
```

Out[∘]= $t\, x1^2 + e^t\, x2$

```
In[∘]:= (* eine Ersetzungregel *)
        rule = t → 2

Out[∘]= t → 2

        (* der Pfeil "→" kann als "->" eingegeben werden *)
```

/. Operator

```
In[∘]:= (* Regel wird auf Ausdruck angewendet *)
        expr /. rule
```

Out[∘]= $2\, x1^2 + e^2\, x2$

In[∘]:=
```
(* gleichzeitige Anwendung von zwei Regeln *)
expr /. {rule, x1 → 0, x2 → 5}
```
Out[∘]= $5\, e^2$

In[∘]:=
```
(* der ursprüngliche Ausdruck wird durch
 die Anwendung von Regeln nicht verändert *)
expr
```
Out[∘]= $t\, x1^2 + e^t\, x2$

In[∘]:=
```
(* es geht alles auch in einer Zeile *)
```
In[∘]:=
```
(s^3 + 5 s^2 - 7 s) /. s → 7
```
Out[∘]= 539

In[∘]:=
```
(* sequentielles Anwenden von Regeln *)
(s^3 + 5 s^2 - 7 s) /. {{s -> 1}, {s -> 2}}
```
Out[∘]= {-1, 14}

In[∘]:=
```
(* wichtiger Spezialfall;
"/." ist in gewisser Weise invers zu "->" *)x /. x → 1
```
Out[∘]= 1

Symbolisches Rechnen mit Ausdrücken

In[∘]:=
```
(* ein Ausdruck *)
expr = Sin[x] * y
```
Out[∘]= y Sin[x]

In[∘]:=
```
(* kann einfach differenziert und integriert werden *)
D[expr, x]
D[expr, y]
Integrate[expr, x]
Integrate[expr, y]
```
Out[∘]= y Cos[x]

Out[∘]= Sin[x]

Out[∘]= -y Cos[x]

Out[∘]= $\frac{1}{2} y^2$ Sin[x]

Algebraische Gleichungen

Beispiel 1: lineares Gleichungssystem

In[•]:=
```
sol = Solve[
   {
    2 x1 + 3 x2 == 0,  (* Gleichung 1 mit "==" *)
    x1 - x2 == 1       (* Gleichung 2, auch mit "==" *)
   },
   {x1, x2}            (* die zu bestimmenden Größen *)
  ]
```

Out[•]= $\left\{\left\{x1 \to \frac{3}{5},\ x2 \to -\frac{2}{5}\right\}\right\}$

In[•]:=
```
(* Ergebnis ist eine Liste von
  Ersetzungsregeln ! Hier gibt es aber nur eine Lösung! *)
Length[sol]
sol1 = First[sol]
```

Out[•]= 1

Out[•]= $\left\{x1 \to \frac{3}{5},\ x2 \to -\frac{2}{5}\right\}$

In[•]:=
```
(* Auswertung durch Elimiation von "→"   *)
```

In[•]:=
```
{x1, x2} /. sol1
```

Out[•]= $\left\{\frac{3}{5},\ -\frac{2}{5}\right\}$

Beispiel 2: nichtlineare Gleichung

In[•]:=
```
(* einfache quadratische Gleichung mit zwei Lösungen *)
sol = Solve[s * s + 2 s - 3 == 0, s]
```

Out[•]= {{s → -3}, {s → 1}}

In[•]:=
```
(* zwei Lösungen *)
Length[sol]
```

Out[•]= 2

In[•]:=
```
(* Auswertung durch Elimiation von "→" *)
```

In[•]:=
```
sol[[1]]
s /. sol[[1]]
```

Out[•]= {s → -3}

Out[•]= -3

In[∘]:=
```
sol[[2]]
s /. sol[[2]]
```

Out[∘]= $\{s \to 1\}$

Out[∘]= 1

Beispiel 3 : eine Gleichungen mit zwei Variablen x und y

In[∘]:=
```
(* suche x als Funktion y *)
solx = Solve[x * x + 2 * y * y == 1, x]
```

Out[∘]= $\left\{\left\{x \to -\sqrt{1-2 y^2}\right\}, \left\{x \to \sqrt{1-2 y^2}\right\}\right\}$

In[∘]:=
```
(* suche y als Funktion x *)
soly = Solve[x * x + 2 * y * y == 1, y]
```

Out[∘]= $\left\{\left\{y \to -\frac{\sqrt{1-x^2}}{\sqrt{2}}\right\}, \left\{y \to \frac{\sqrt{1-x^2}}{\sqrt{2}}\right\}\right\}$

Beispiel 3 : zwei nichtlineare Gleichungen für zwei Variablen

In[∘]:=
```
sol = Solve[
  {
   x * x + 2 * y * y == 1, (* Gleichung 1 *)
   x + 2 * y == 0          (* Gleichung 2 *)
  },
  {x, y}
 ]
```

Out[∘]= $\left\{\left\{x \to -\sqrt{\frac{2}{3}}, y \to \frac{1}{\sqrt{6}}\right\}, \left\{x \to \sqrt{\frac{2}{3}}, y \to -\frac{1}{\sqrt{6}}\right\}\right\}$

In[∘]:=
```
(* Ersetzungsregeln für erste Lösung *)
sol[[1]]
```

Out[∘]= $\left\{x \to -\sqrt{\frac{2}{3}}, y \to \frac{1}{\sqrt{6}}\right\}$

In[∘]:=
```
(* erste Lösung *)
{x, y} /. sol[[1]]
```

Out[∘]= $\left\{-\sqrt{\frac{2}{3}}, \frac{1}{\sqrt{6}}\right\}$

In[∘]:=
```
(* Ersetzungsregeln für zweite Lösung *)
sol[[2]]
```

Out[∘]= $\left\{x \to \sqrt{\frac{2}{3}}, y \to -\frac{1}{\sqrt{6}}\right\}$

In[∘]:=
```
(* zweite Lösung *)
{x, y} /. sol[[2]]
```

Out[∘]= $\left\{\sqrt{\frac{2}{3}}, -\frac{1}{\sqrt{6}}\right\}$

Beispiel 5: transzendente Funktionen

In[∘]:=
```
(* Mathematica ist sich manchmal nicht sicher *)
Solve[x * Exp[x] == y, x]
```

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

Out[∘]= {{x → ProductLog[y]}}

In[∘]:=
```
(* manchmal findet es Lösungen mit freiem Parameter und/oder Bedingungen *)
Solve[Tan[x] == y, x]
```

Out[∘]= {{x → ConditionalExpression[ArcTan[y] + $\pi\, c_1$, $c_1 \in \mathbb{Z}$]}}

In[∘]:=
```
(* manchmal gar keine *)
Solve[Exp[x] + Tan[x] == y, x]
```

Solve: This system cannot be solved with the methods available to Solve.

Out[∘]= Solve[e^x + Tan[x] == y, x]

Beispiel 6: Gleichungen mit Parameter

In[∘]:=
```
(* der Parameter heißt hier "a" *)
sol = Solve[
   {
    2 x1 + 3 x2 == 1,
    2 x1 + a * x2 == 0
   },
   {x1, x2}
  ]
```

Out[∘]= $\left\{\left\{x1 \to \frac{a}{2\,(-3+a)},\ x2 \to -\frac{1}{-3+a}\right\}\right\}$

In[∘]:=
```
(* Lösung für a=0 *)
sol /. a → 0
{x1, x2} /. First[sol /. a → 0]
```

Out[∘]= $\left\{\left\{x1 \to 0,\ x2 \to \frac{1}{3}\right\}\right\}$

Out[∘]= $\left\{0, \frac{1}{3}\right\}$

In[∘]:=
```
(* Lösung für a=1 *)
sol /. a → 1
{x1, x2} /. First[sol /. a → 1]
```

Out[∘]= $\left\{\left\{x1 \to -\frac{1}{4},\ x2 \to \frac{1}{2}\right\}\right\}$

Out[∘]= $\left\{-\frac{1}{4},\ \frac{1}{2}\right\}$

In[∘]:=
```
(* Lösung für a=3 ist singulär *)
sol /. a → 3
```

Power: Infinite expression $\frac{1}{0}$ encountered.

Power: Infinite expression $\frac{1}{0}$ encountered.

Out[∘]= {{x1 → ComplexInfinity, x2 → ComplexInfinity}}

Numerische Lösungen

In[∘]:=
```
(* wenn Solve versagt
 (weil es zum Beispiel keine geschlosse Lösungsformel gibt) *)
sol = Solve[x^6 + x^4 + 2 x^2 + x - 4 == 0, x]
```

Out[∘]= {{x → -1.07...}, {x → 0.925...}, {x → -0.675... - 1.27... i},
{x → -0.675... + 1.27... i}, {x → 0.746... - 1.19... i}, {x → 0.746... + 1.19... i}}

In[∘]:=
```
(* probiere NSolve *)
NSolve[x^16 + x^4 + 2 x^2 + x - 4 == 0, x]
```

Out[∘]= {{x → -1.03532}, {x → -1.02326 - 0.371717 i}, {x → -1.02326 + 0.371717 i},
{x → -0.809649 - 0.781125 i}, {x → -0.809649 + 0.781125 i},
{x → -0.419951 - 1.03956 i}, {x → -0.419951 + 1.03956 i},
{x → 0.0150521 - 1.10648 i}, {x → 0.0150521 + 1.10648 i}, {x → 0.440918 - 1.02134 i},
{x → 0.440918 + 1.02134 i}, {x → 0.820087 - 0.755154 i}, {x → 0.820087 + 0.755154 i},
{x → 0.949383}, {x → 1.01977 - 0.325497 i}, {x → 1.01977 + 0.325497 i}}

In[∘]:=
```
(* transzendentes Beispiel *)
NSolve[x * Log[x] == 4, x]
```

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

Out[∘]= {{x → 3.32732}}

FindRoot um benachbarte Lösung zu finden

FindRoot benutzt intern eine Variante des Newton - Verfahrens.
Dieser Befehl findet nur eine, niemals alle Lösungen

```
In[∘]:= (* wähle Startwert x=0 *)
FindRoot[x^6 + x^4 + 2 x^2 + x - 4, {x, 0}]
Out[∘]= {x → 0.92544}
```

```
In[∘]:= (* wähle Startwert x=-5 *)
FindRoot[x^6 + x^4 + 2 x^2 + x - 4, {x, -5}]
Out[∘]= {x → -1.06811}
```

```
In[∘]:= (* mehr Infos unter *)
```

Gleichungen

Optimierung

Gewöhnliche Differentialgleichungen

DSolve - Beispiel 1 (harmonischer Oszillator)

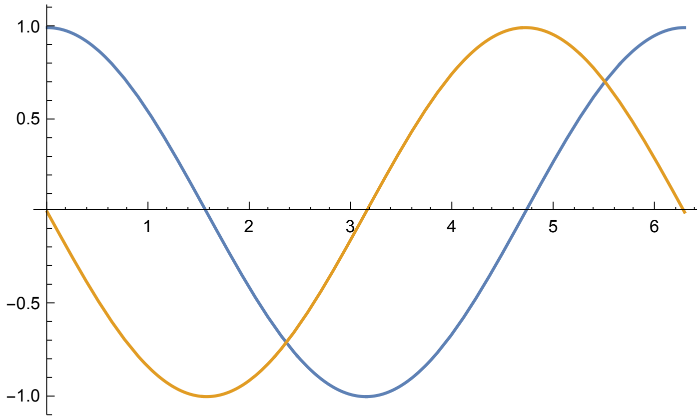
Variante A

```
In[∘]:= (* DSolve liefert exakte Lösung in Form von Ersetzungsregeln *)
solA =
 DSolve[
  {
   x1'[t] == x2[t],     (* DGl 1 *)
   x2'[t] == -x1[t],    (* DGl 2 *)
   x1[0] == a1,         (* AW 1 *)
   x2[0] == a2          (* AW 2 *)
  },
  {x1[t], x2[t]},       (* zu bestimmende Terme *)
  t                     (* unabhängige Koordinate *)
 ]
Out[∘]= {{x1[t] → a1 Cos[t] + a2 Sin[t], x2[t] → a2 Cos[t] - a1 Sin[t]}}
```

```
In[∘]:= (* spezielle Lösungen mit konkreten Anfangswerten  *)
X1[t_] = x1[t] /. First[solA] /. {a1 → 1, a2 → 0}
X2[t_] = x2[t] /. First[solA] /. {a1 → 1, a2 → 0}
Out[∘]= Cos[t]
Out[∘]= -Sin[t]
```

In[∘]:=
```
Plot[{X1[t], X2[t]}, {t, 0, 2 Pi}]
```

Out[∘]=



Variante 2

In[∘]:=
```
(* Losungen als abtrakte Funktionen *)
solB =
 DSolve[
  {
   x1'[t] == x2[t],
   x2'[t] == -x1[t],
   x1[0] == a1,
   x2[0] == a2
  },
  {x1, x2}, (* diese Zeile ist anders als vorher *)
  t
 ]
```

Out[∘]=
```
{{x1 → Function[{t}, a1 Cos[t] + a2 Sin[t]],
  x2 → Function[{t}, a2 Cos[t] - a1 Sin[t]]}}
```

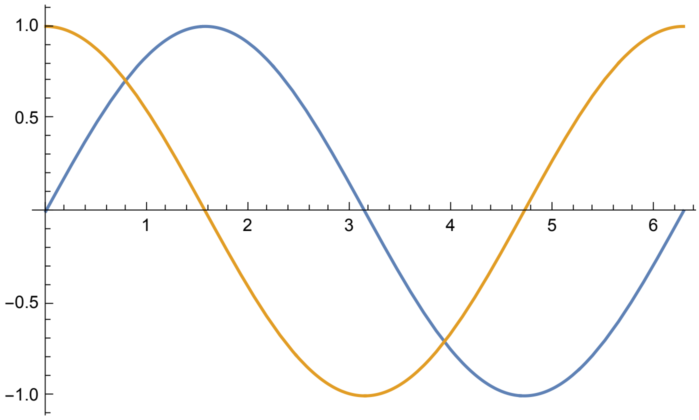
In[∘]:=
```
(* Auswertung *)
X1 = x1 /. First[solB] /. {a1 → 0., a2 → 1}
X2 = x2 /. First[solB] /. {a1 → 0., a2 → 1}
```

Out[∘]=
```
Function[{t}, 0. Cos[t] + 1 Sin[t]]
```

Out[∘]=
```
Function[{t}, 1 Cos[t] - 0. Sin[t]]
```

In[∘]:=
```
Plot[{X1[t], X2[t]}, {t, 0, 2. Pi}]
```

Out[∘]=



DSolve - Beispiel 2 (in einer Dimension)

In[∘]:=
```
(* nichtlineare skalare Gleichung *)
x[t] /. First[
  NDSolve[
   {
    x'[t] == -x[t] * Sin[x[t]],
    x[0] == 2
   },
   {x[t]},
   {t, 0, 10}
  ]
 ]
```

Out[∘]=
```
InterpolatingFunction[ Domain: {{0., 10.}} Output: scalar ][t]
```

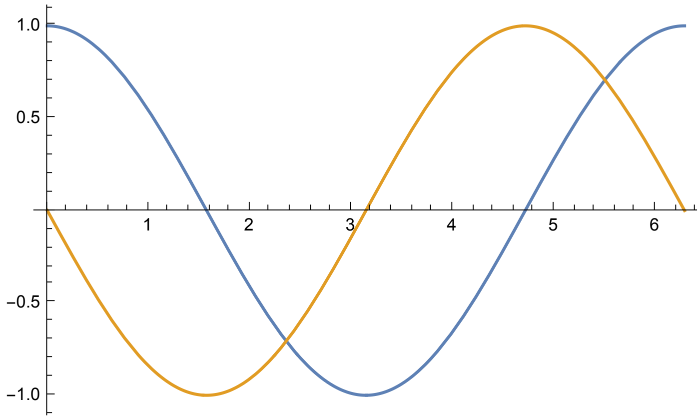
NDSolve - Beispiel 1 (nochmal harmonischer Oszillator)

In[∘]:=
```
(* Numerische Lösung konkreter Anfangswertprobleme *)
sol =
 First[
  NDSolve[
   {
    x1'[t] == x2[t],   (* DGl 1 *)
    x2'[t] == -x1[t],  (* DGl 2 *)
    x1[0] == 1,        (* konkrete Anfangsbedingung 1 *)
    x2[0] == 0         (* konkrete Anfangsbedingung 2 *)
   },
   {x1[t], x2[t]},
   {t, 0, 2 Pi}        (* konkretes Intervall *)
  ]
 ]
```

Out[∘]=
```
{x1[t] → InterpolatingFunction[ Domain: {{0., 6.28}} Output: scalar ][t],
 x2[t] → InterpolatingFunction[ Domain: {{0., 6.28}} Output: scalar ][t]}
```

In[∘]:=
```
Plot[{x1[t] /. sol, x2[t] /. sol}, {t, 0, 2 Pi}]
```

Out[∘]=



In[∘]:=
```
(* mehr unter *)
```

Differentialgleichungen