

Einführung in Sage

Einheit 6

Jochen Schulz

Georg-August Universität Göttingen



4. Februar 2010

- 1 Funktionen
- 2 Grenzwerte und Stetigkeit
- 3 Funktionenfolgen
- 4 Grafiken

- 1 Funktionen
- 2 Grenzwerte und Stetigkeit
- 3 Funktionenfolgen
- 4 Grafiken

Man spricht von einer (reellen) **Funktion**, wenn ein **Definitionsbereich** $D \subset \mathbb{R}$, $D \neq \emptyset$ gegeben ist und eine Vorschrift, die jedem $x \in D$ in eindeutiger Weise eine reelle Zahl $f(x)$ zuordnet. Man schreibt

$$f: D \rightarrow \mathbb{R}.$$

Die Menge $f(D)$ ist die Menge aller reellen Zahlen, die als Werte der Funktion vorkommen. Die Menge $f(D)$ wird als **Wertebereich** bezeichnet. Der **Graph** einer Funktion ist die Menge aller Punkte

$$\{(x, f(x)) \in \mathbb{R}^2 \mid x \in D\}.$$

Verknüpfungen

Seien f und g Funktionen mit einem gemeinsamen Definitionsbereich.
Dann definiert man:

- Summe: $(f + g)(x) := f(x) + g(x)$
- Differenz: $(f - g)(x) := f(x) - g(x)$
- Produkt: $(f \cdot g)(x) := f(x) \cdot g(x)$
- Quotient: $(\frac{f}{g})(x) := \frac{f(x)}{g(x)}$, falls $g(x) \neq 0$ für alle $x \in D$

Sind $f: D_f \rightarrow \mathbb{R}$ und $g: D_g \rightarrow \mathbb{R}$ mit $f(D_f) \subset D_g$ so ist die Komposition definiert durch:

$$(g \circ f)(x) := g(f(x)).$$

Mehrere Veränderliche

Ist $D \subseteq \mathbb{R}^n$ und $f: D \Rightarrow \mathbb{R}$ dann spricht man von einer reellen Funktion in **mehreren Veränderlichen**. Das Studium dieser Funktionen ist einer der Hauptinhalte der Diff2-Vorlesung.

Weiterhin können Funktionen auch Wertebereiche außerhalb der reellen Zahlen haben. Z.B.

$$f: D \Rightarrow \mathbb{R}^m.$$

Im physikalischen Umfeld spricht man für $m = 1$ dann von **skalarwertigen Funktionen** und für $m > 1$ von **vektorwertigen Funktionen** oder **Vektorfeldern**.

Abbildungen in Sage I

In Sage wird eine Abbildung f durch einen Ausdruck der Form $f(x,y,\dots)$ gebildet, wobei x,y die sind.

```
>> f(x,y) = x^2+y^2; f
```

```
(x, y) |--> x^2 + y^2
```

Die so definierte Funktion f kann wie jede beliebige andere Funktion aufgerufen werden. Funktionen haben den Datentyp `expression`.

```
>> _=var('a,b');f(a,b+1)
```

```
(b + 1)^2 + a^2
```

```
>> type(f)
```

```
<type 'sage.symbolic.expression.Expression'>
```

Abbildungen in Sage II

Wie gewohnt können Abbildungen addiert, subtrahiert, multipliziert und dividiert werden:

```
f(x) = 1/(1+x); g(x) = sin(x^2)
h = f+g; k = f*g; l = f/g
h(a),k(a),l(a)
```

```
(1/(a + 1) + sin(a^2), sin(a^2)/(a + 1), 1/((a + 1)*
sin(a^2)))
```


Kompositionen in Sage I

Eine Komposition $f \circ g$ wird in Sage durch Ineinanderschachteln gelöst.

```
f_g(x) = f(g); g_f(x) = g(f)
f_g(x), g_f(x)
```

```
(1/(sin(x^2) + 1), sin((x + 1)^(-2)))
```

Mehrfaches Hintereinanderschalten $f(f(\dots f(\cdot))) = f \circ \dots \circ f(\cdot)$ wird in Sage ebenso durchgeführt.

```
g4(x) = g(g(g(g))); g4
```

```
x |--> sin(sin(sin(sin(x^2)^2)^2)^2)
```

Kompositionen in Sage II

Diese Konstruktionen funktionieren auch mit Systemfunktionen:

```
>> abs(real(-2+3*I))
```

```
2
```

Kompliziertere Funktionen können besser durch selbst definierte Funktionen erklärt werden. Dies sind im Wesentlichen kleine Programme, die mit `def <func>():` beginnen (vgl. letzte Vorlesung).

Ausdrücke und Funktionen I

Man kann in Sage wählen, ob man eine Funktion f als Funktion oder als Ausdruck darstellt. Die Funktionsauswertung ist i.A. allerdings unterschiedlich:

```
>> Funktion(x) = 2*x*cos(x); Funktion(1)
```

```
2*cos(1)
```

```
Ausdruck = 2*x*cos(x); Ausdruck(x=1)
```

```
2*cos(1)
```

Ausdrücke und Funktionen II

Auch mehrere Veränderliche sind möglich:

```
_ = var('y'); Funktion2(x) = x + sin(y); Funktion2
```

```
x |--> x + sin(y)
```

```
>> Funktion3(x,y) = x + sin(y); Funktion3
```

```
(x, y) |--> x + sin(y)
```

1 Funktionen

2 Grenzwerte und Stetigkeit

3 Funktionenfolgen

4 Grafiken

Grenzwerte von Funktionen

Sei f eine Funktion mit Definitionsbereich D und $a \in D$. f strebt für $x \rightarrow a$ gegen den **Grenzwert** $b \in \mathbb{R}$, wenn es zu jedem $\varepsilon > 0$ ein $\delta > 0$ gibt, so dass für alle $x \in D \setminus \{a\}$ mit $|x - a| < \delta$ gilt

$$|f(x) - b| < \varepsilon.$$

Der Grenzwert b ist eindeutig bestimmt und man schreibt

$$\lim_{x \rightarrow a} f(x) = b \text{ oder } f(x) \rightarrow b \text{ für } x \rightarrow a.$$

Die Aussage überträgt sich sinngemäß auf $a = \pm\infty$.

- Folgenkriterium: Es gilt $\lim_{x \rightarrow a} f(x) = b$ genau dann, wenn für **jede** Folge $a_n \in D$ mit $a_n \neq a$ und $a_n \rightarrow a$ gilt $\lim_{n \rightarrow \infty} f(a_n) = b$.
- Es gelten die üblichen Rechenregeln:

$$\begin{aligned}\lim_{x \rightarrow a} (f(x) + g(x)) &= \lim_{x \rightarrow a} f(x) + \lim_{x \rightarrow a} g(x) \\ \lim_{x \rightarrow a} (f(x) \cdot g(x)) &= \lim_{x \rightarrow a} f(x) \cdot \lim_{x \rightarrow a} g(x)\end{aligned}$$

wenn $\lim_{x \rightarrow a} f(x)$ **und** $\lim_{x \rightarrow a} g(x)$ **existieren**.

- Gilt $\lim_{x \rightarrow a} f(x) = b$, $\lim_{x \rightarrow b} g(x) = c$ bei entsprechenden Definitionsgebieten für f und g , so folgt $\lim_{x \rightarrow a} g(f(x)) = c$.

Grenzwerte werden in Sage mit dem Befehl `limit` gebildet. Die Syntax des Befehls lautet

```
>> expr.limit(x = a, dir=None, taylor=False)
>> limit(expr, x = a, dir=None, taylor=False)
```

Hierdurch wird der Grenzwert eines Ausdrucks mit Unbekannten x an der Stelle a bestimmt. a kann auch $\pm\infty$ sein (in Sage `infinity` oder `oo`). Ruft man `limit` ohne option auf, so wird der beidseitige Limes berechnet. Falls `dir='minus'` ist, wird der linksseitige Limes berechnet; für `dir='plus'` der rechtsseitige.

Beispiele in Sage I

- Bestimme den Grenzwert $\lim_{x \rightarrow 0} \frac{\sin(x)}{x}$

```
>> limit(sin(x)/x,x=0)
```

1

- Bestimme den Grenzwert $\lim_{x \rightarrow \infty} \frac{\log(x)}{x}$

```
>> limit(log(x)/x,x=infinity)
```

0

- Bestimme den Grenzwert $\lim_{x \rightarrow \infty} \sqrt[x]{x}$

```
>> limit(x^(1/x),x=infinity)
```

1

Beispiele in Sage II

- Bestimme den Grenzwert $\lim_{x \rightarrow 0} \sin(1/x)$

```
>> limit(sin(1/x), x=0)
```

```
ind
```

Der Grenzwert existiert nicht. Sage gibt in diesem Fall `ind` (indefinite aber beschränkt) zurück.

- Bestimme den Grenzwert $\lim_{x \rightarrow 0} |x|'$

```
>> limit(diff(abs(x), x), x=0),  
limit(diff(abs(x), x), x=0, dir='minus'),  
limit(diff(abs(x), x), x=0, dir='plus'))
```

```
(und, -1, 1)
```

Eine Funktion $f: D \rightarrow \mathbb{R}$ heißt **stetig an der Stelle $x_0 \in D$** , wenn es zu jedem $\varepsilon > 0$ ein $\delta > 0$ gibt, so dass für alle $x \in D$ mit $|x - x_0| < \delta$ gilt

$$|f(x) - f(x_0)| < \varepsilon.$$

Man sagt, dass f **stetig** ist, wenn f an jeder Stelle $x_0 \in D$ stetig ist.
Sind f und g an x_0 stetig, so auch $f + g$, $f - g$, $f \cdot g$ und $\frac{f}{g}$ (falls $g(x_0) \neq 0$).

Wichtige Sätze I

- Sei f auf einem offenen Intervall I definiert. f ist an $x_0 \in I$ genau dann stetig, wenn gilt

$$\lim_{x \rightarrow x_0} f(x) = f(x_0).$$

- Für $f: I \rightarrow \mathbb{R}$ und $g: J \rightarrow \mathbb{R}$ gelte $f(I) \subset J$ und es seien f an $x_0 \in I$ und g an $y_0 = f(x_0)$ stetig. Dann ist $g \circ f$ an x_0 stetig.
- Eine Funktion $f: D \rightarrow \mathbb{R}$ ist **linksstetig** bzw. **rechtsstetig**, wenn $f|_{D \cap (-\infty, x_0)}$ bzw. $f|_{D \cap (x_0, \infty)}$ an x_0 stetig ist. Eine Funktion f ist dann an x_0 stetig, genau dann wenn f links- und rechtsstetig an x_0 ist.

Wichtige Sätze II

- Eine stetige Funktion auf einem abgeschlossenen Intervall $I = [a, b]$ besitzt ein Maximum und ein Minimum.
- Eine stetige Funktion f auf einem abgeschlossenen Intervall $[a, b]$ nimmt in I jeden Wert zwischen $f(a)$ und $f(b)$ an.
- Potenzreihen $f(x) = \sum_{n=0}^{\infty} a_n(x - x_0)^n$ sind stetig innerhalb ihres Konvergenzintervalls.

$f: D \rightarrow \mathbb{R}$ heißt **gleichmäßig stetig auf D** , wenn es zu jedem $\varepsilon > 0$ ein $\delta > 0$ gibt, so dass für alle Paare $x, x_0 \in D$ mit $|x - x_0| < \delta$ gilt

$$|f(x) - f(x_0)| < \varepsilon.$$

- Die Exponentialfunktion ist auf jedem kompakten Intervall gleichmäßig stetig (aber nicht auf ganz \mathbb{R}).
- $\log : (0, 1) \rightarrow \mathbb{R}$ ist stetig aber nicht gleichmäßig stetig.

Für die Diskussion der Stetigkeit einer Funktion f an einer Stelle x_0 sei auf den Abschnitt zu Grenzwerten verwiesen.

```
limit(1/x,x=0,dir='plus')
```

```
+Infinity
```

```
limit(1/x,x=0,dir='minus')
```

```
-Infinity
```

- 1 Funktionen
- 2 Grenzwerte und Stetigkeit
- 3 Funktionenfolgen**
- 4 Grafiken

Funktionenfolgen

Seien $f_n : D \rightarrow \mathbb{R}$, $n \in \mathbb{N}$ reellwertige Funktionen auf $D \subset \mathbb{R}$.

- $(f_n)_n$ heißt **Funktionenfolge**.
- Ist für jedes $x \in D$ die Folge $(f_n(x))_n$ konvergent, so wird durch

$$f(x) := \lim_{n \rightarrow \infty} f_n(x), \quad x \in D$$

die **Grenzfunktion** $f : D \rightarrow \mathbb{R}$ definiert.

- Man sagt f_n strebe **punktweise** auf D gegen f .
- Durch $\sum_{i=1}^{\infty} f_i$ definierte **Funktionenreihen** sind spezielle Funktionenfolgen.

Beispiele: Grenzübergänge

- $x^n \rightarrow 0$ auf dem Intervall $(-1, 1)$.
- $(1 + \frac{x}{n})^n \rightarrow \exp(x)$ auf \mathbb{R} .
- Potenzreihen konvergieren innerhalb ihres Konvergenzradius.
- **Warnung** zum Vertauschen der Grenzprozesse für $x \in (0, 1)$:

$$\lim_{x \rightarrow 1} \lim_{n \rightarrow \infty} x^n = 0 \neq 1 = \lim_{n \rightarrow \infty} \lim_{x \rightarrow 1} x^n.$$

Gleichmäßige Konvergenz

Definition

$(f_n)_n$ konvergiert **gleichmäßig** auf D gegen f , wenn es zu jedem $\varepsilon > 0$ ein $n_0 \in \mathbb{N}$ gibt, so dass für alle $x \in D$ und $n \geq n_0$ gilt:

$$|f_n(x) - f(x)| < \varepsilon.$$

Satz

Konvergiert $(f_n)_n$ gleichmäßig auf D und existiert $\lim_{x \rightarrow a} f_n(x)$ für $a \in D$, so gilt:

$$\lim_{x \rightarrow a} \lim_{n \rightarrow \infty} f_n(x) = \lim_{n \rightarrow \infty} \lim_{x \rightarrow a} f_n(x).$$

- Die Grenzfunktion einer gleichmäßig konvergenten Folge stetiger Funktionen ist stetig.
- Alle Aussagen übertragen sich analog auf Funktionenreihen: Ist f_1, f_2, \dots , eine Folge von Funktionen auf $D \subseteq \mathbb{R}$ dann definiert

$$s := \sum_{n=1}^{\infty} f_n$$

eine Funktionenreihe. Aussagen über die Funktionenreihe sind, analog zum Fall der „normalen“ Reihen, Aussagen über die Folge der Partialsummen

$$s_k := \sum_{n=1}^k f_n.$$

- 1 Funktionen
- 2 Grenzwerte und Stetigkeit
- 3 Funktionenfolgen
- 4 Grafiken**

- Die Funktionen `plot` und `plot3d` dienen zur Darstellung von Graphen von Funktionen mit einem bzw. zwei Argumenten.
- Grafiken werden im Notebook integriert.
- Die meisten 3D-Grafiken können interaktiv bearbeitet werden.

plot()

Skalare Funktionen f_1, f_2, \dots , können durch den Befehl

```
>> plot(f1, (x, a, b))
```

auf dem Intervall $x \in [a, b]$. Dabei ist die Angabe von (x, a, b) optional.
Beispiele:

```
>> plot(x^2-1)
>> plot(sin(1/x), (x, -1, 1))
>> plot(sin(1/x), (x, -1, 1), adaptive_recursion=0)
```

```
>> p = plot(sin(x), color='red', xmin=-pi, xmax=pi)
>> p += plot(cos(x), xmin=-pi, xmax=pi); p.show()
```

plot3d()

Es werden Funktionen $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ grafisch dargestellt. (Genauer gesagt wird auch hier der Funktionsgraph als Teilmenge des \mathbb{R}^3 gezeichnet). Der Funktionsaufruf ist

```
>> plot3d(f,(x,a,b),(y,c,d), plot_points=[nx,ny])
```

Hierdurch werden Ausdrücke f_1, f_2, \dots , auf $[a, b] \times [c, d]$ mit Hilfe von n_x Gitterpunkten in x -Richtung und n_y Punkten in y -Richtung dargestellt.

Plot von $f(x, y) = \sin(y^2 + x) - \cos(y - x^2)$ auf $[0, \pi]^2$:

```
>> f(x,y) = sin(y^2+x)-cos(y-x^2)
>> plot3d(f,(x,0,pi),(y,0,pi))
>> plot3d(f,(x,0,pi),(y,0,pi),plot_points=[10,10])
```

Plot von $f(x, y) = \cos(20 \exp(-x^2 - y^2))$ auf $[-1, 1] \times [-1, 1]$:

```
>> g(x,y) = cos(20*exp(-x^2-y^2))
>> plot3d(g,(x,-1,1),(y,-1,1))
>> plot3d(g,(x,-1,1),(y,-1,1),plot_points=[80,80])
```

Grafiken werden nicht direkt erzeugt, d.h. ausgegeben, sondern erst werden **grafische Objekte** wie Geraden, Funktionsgraphen oder Kurven erzeugt. Erst beim `show()` oder falls sie in der letzten Zeile ohne Zuweisung stehen, werden die Objekte zu einer gemeinsamen **grafischen Szene** zusammengefaßt.

```
>> W = plot3d(sin(pi*((x)^2+(y)^2))/2,(x,-1,1),(y
    ,-1,1), frame=False, color='purple', opacity=0.8)
>> S = sphere((0,0,0),size=0.3, color='red',
    aspect_ratio=[1,1,1])
>> show(W + S, figsize=8)
```

Optionen für grafische Szenen

<code>aspect_ratio</code>	Verhältnis der Achsen (Breite/Höhe). 1 für 1:1 Verhältnis. <code>aspect_ratio = 2</code>
<code>figsize</code>	Grösse des Bildes <code>figsize = [width, height]</code>
<code>axes_labels</code>	Tuple oder Liste der Achsenbeschriftungen <code>axes_labels = ('\$x\$', '\$y\$')</code>
<code>gridlines</code>	Gitterlinien <code>gridlines = True</code>

(ein paar) Optionen für grafische Objekte

linestyle	Darstellung von Linien ('-' (solid), '- .' (dashed), ':') (dotted) <code>linestyle = '.'</code>
thickness	Linienstärke in mm <code>thickness = 4</code>
color	Zuweisung einer Farbe <code>color='red'</code>
plot_points	Anzahl Stützstellen <code>plot_points = [nx,ny]</code> (2 Parameter)
alpha	Transparent <code>alpha = 0.8</code>

Zweidimensionale Kurven

Eine Kurve des \mathbb{R}^2 in Parameterdarstellung sei gegeben durch Funktionen $x(t), y(t)$, also die Menge aller Punkte:

$$\{(x(t), y(t)) \in \mathbb{R}^2 \mid t \in [a, b]\}.$$

Zum Beispiel ergibt sich der Graph einer Funktion $f(x)$, $x \in [a, b]$ durch $t, f(t)$ mit $t \in [a, b]$. Der Befehl zum Erzeugen des Objekts ist

```
>> Objekt = parametric_plot([x,y],(t,a,b))
```

x und y sind Ausdrücke mit der Unbekannten t . Zusätzlich können noch Optionen übergeben werden.

Beispiele

```
>> _=var('t');f1 = parametric_plot([t,sin(t)],(t
    ,0,2*pi),color='red')
>> f2 = parametric_plot([t,cos(t)],(t,0,2*pi),color=
    'green')
>> f3 = parametric_plot([cos(t),sin(t)],(t,0,2*pi))
>> (f1+f2+f3).show(figsize=7)
```

```
>> [parametric_plot([cos(t),sin(t)],(t,0,2*pi),
    plot_points=2*k+1,randomize=False,
    adaptive_recursion=0 ) for k in [2..10]]
```

Dreidimensionale Kurven

Das Erzeugen einer dreidimensionalen Kurve, d.h. einer Kurve des \mathbb{R}^3 geschieht analog durch die Angabe von drei Funktionen $x(t), y(t), z(t)$:

$$\{(x(t), y(t), z(t)) \in \mathbb{R}^3 \mid t \in [a, b]\}.$$

Der Befehl ist

```
>> Objekt = parametric_plot([x,y,z],(t,a,b))
```

x , y und z sind Ausdrücke mit der Unbekannten t . Zusätzlich können noch Optionen übergeben werden. Beispiel:

```
>> parametric_plot3d([(1-t*t)*cos(99*t),(1-t*t)*sin(99*t),t], (t,0,1),plot_points=400)
```

- Typische dreidimensionale Grafikobjekte sind *parametrisierte Flächen*.
- Die x, y, z Koordinaten sind als Funktionen $x(t_1, t_2)$, $y(t_1, t_2)$, $z(t_1, t_2)$ zweier Parameter $t_1 \in [a, b]$ und $t_2 \in [c, d]$ definiert.

- Beispielsweise lassen sich Graphen von Funktionen $f: [a, b] \times [c, d] \rightarrow \mathbb{R}$ als Flächen

$$x = t_1, y = t_2, z = f(t_1, t_2)$$

mit $a \leq t_1 \leq b, c \leq t_2 \leq d$ erklären.

- Die Oberfläche einer Kugel mit Radius r ist gegeben durch:

$$x = r \cos(t_1) \sin(t_2), y = r \sin(t_1) \sin(t_2), z = r \cos(t_2)$$

mit $0 \leq t_1 \leq 2\pi, 0 \leq t_2 \leq \pi$.

Flächen werden in Sage erzeugt durch

```
parametric_plot3d( [x(t1,t2), y(t1,t2), z(t1,t2)],(  
    t1,a,b), (t2,c,d), Optionen):
```

Beispiel: Ellipsoid-Oberfläche

```
>> _=var('t1,t2')  
>> r=1  
>> x=r*cos(t1)*sin(t2)  
>> y=2*r*sin(t1)*sin(t2)  
>> z=r*cos(t2)  
>> parametric_plot3d([x,y,z],(t1,0,2*pi),(t2,0,pi),  
    aspect_ratio=1)
```

- Man kann für Funktionen $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ auch zweidimensionale Grafiken erstellen.
- Man zeichnet die Niveaulinien (wie die Höhenmeter auf einer Landkarte), d.h. man sucht zu $c \in \mathbb{R}$ die Menge von Punkten (x, y) mit $f(x, y) = c$.
- In Sage geschieht das mit dem Befehl

```
>> contour_plot(f, (x,,), (y,,), contours=[c1,c2  
    ,...], options)
```

Dabei geben c_1, c_2, \dots die entsprechenden Niveaulinien an.

Beispiel

Wir betrachten die Funktion $\sin(4\pi x)y$ und zeichnen die Niveaulinien für $-0.5, 0, 0.5$.

```
>> contour_plot(sin(pi*4*x)*y,(x,-1,1),(y,-1,1),  
    contours = [-0.5, 0, 0.5],fill=False,labels=True)
```

`fill=True` würde die Flächen zwischen den Linien ausfüllen.

Punkte zeichnen

Mittels `point` können Punkte gezeichnet werden.

Beispiel:

```
>> point([(i,sin(i*6.28/50)) for i in [0..50]],color
        ='red', pointsize=30)
>> point2d.options
```

Ein Beispiel: Das Collatz Problem

Sei $x_0 \in \mathbb{N}$. Dann definiert man die folgende Folge

$$x_n := \begin{cases} x_{n-1}/2, & \text{falls } x_{n-1} \text{ gerade ist} \\ 3x_{n-1} + 1 & \text{falls } x_{n-1} \text{ ungerade ist} \end{cases}.$$

Man kann zeigen, dass für alle Startwerte ein N_0 existiert mit $x_{N_0} = 1$.

```
def collatz(n):  
    """ Collatz problem """  
    sequence = [n]; next_value = n;  
    while next_value > 1:  
        if next_value % 2 == 0:  
            next_value = next_value/2  
        else:  
            next_value = 3*next_value+1  
            sequence.append(next_value)  
    Objekt = point([(i,sequence[i]) for i in [0..len(  
        sequence)-1] ])  
    Objekt.show()  
    return sequence
```

mit dem Befehl `animate()` lassen sich einfach Animationen erstellen.
Beispiele:

- Zweidimensionales Beispiel

```
>> a = animate([plot(a*x^2, (x,-5,5)) for a in  
    [-10..10]], ymin=-100, ymax=100)  
>> a.show(iterations=1)
```