Institut für
Angewandte Mathematik

Leibniz
Universität
Hannover

Dr. Max Winkler
Gregor Pfau, Martin Steinbach

Hannover, 22.04.2020

Computeralgebra – Übung 1

Theorieteil

In diesem Abschnitt des Arbeitsblattes lernen wir die ersten Funktionen von SAGEMATH kennen. Sie lernen hier folgende Aspekte:

- SAGE als Taschenrechner verwenden
- parameterabhängige Ausdrücke
- Funktionen definieren
- Funktionsgraphen erstellen, Nullstellenberechnung, Ableiten und Integrieren
- Fallunterscheidungen
- Einfache `for`-Schleifen

Lesen Sie sich diesen Teil bitte gründlich durch. Sie können gern nebenbei die Befehle in ihr Sage-Notebook abtippen und neue Dinge ausprobieren.

Einfache Rechenaufgaben

Wir können SageMath als einfachen Taschenrechner verwenden.

```
In [1]: 4+(9-2^3)     # Einfache Punkt- und Strichrechnung
```

```
Out[1]: 5
```

```
In [2]: pi/2+pi/3     # Bruchrechnung
```

```
Out[2]: 5/6*pi
```

```
In [3]: a=3*(pi/3+2/3)/pi # Ein komplizierter Ausdruck
        print("a =", a)    # Formatierte Ausgabe
```

```
a = (pi + 2)/pi
```

Wir sehen, dass Sage komplizierte Ausrücke automatisch vereinfacht. Bevorzugt man eine Ausgabe als Gleitkommazahl, kann man den Befehl N verwenden.

```
In [4]: print("a =", N(a)) # Umwandeln in eine Gleitkommazahl
```

```
a = 1.63661977236758
```

Offensichtlich beherrscht Sage sogar die Bruchrechnung und kann Terme vereinfachen.

```
In [5]: e=(x-1)*(x+2)                          # Definiert eine Expression
        print("Ausdruck: ", e)                 # Formatierte Ausgabe
        print("Ausmultipliziert: ", expand(e)) # Expression nach Ausmultiplizieren
```

```
Ausdruck:  (x + 2)*(x - 1)
Ausmultipliziert:  x^2 + x - 2
```

Variablen und Funktionen

Mit dem Zuweisungsoperator = lassen sich Variablen definieren. Der Befehl var legt eine generische Variable an, die später gesetzt werden kann.

```
In [6]: a=3+4 # Schreibt das Ergebnis in die Variable a
        a     # Gibt den Wert von a aus

Out[6]: 7

In [7]: c=var('C')    # Definiert einen (noch freien) Parameter
        b=3+c         # Definiert einen Parameterabhängigen Ausdruck
        b             # Gibt den Wert von b aus

Out[7]: C + 3

In [8]: b.subs(c==4)    # Gibt b für den Parameter c=4 erneut aus

Out[8]: 7
```

Diese Methode ist nicht dafür geeignet Funktionen zu definieren. Für Funktionen schreibt man einfach nur

```
f(x)=x^2
```

und zum Auswerten der Funktion

```
f(4)
```

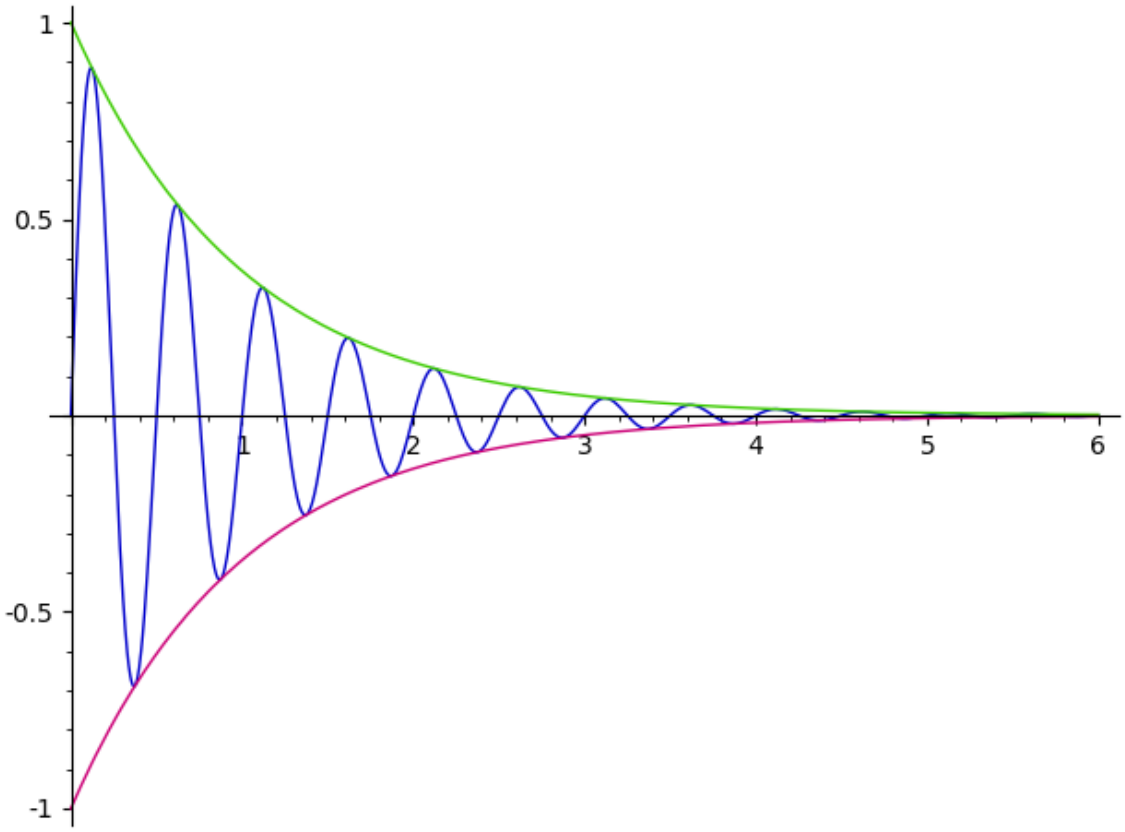
Sage bietet sehr viele Funktionen zum Rechnen und Darstellen von Funktionen an.

```
In [9]: f(x)=sin(4*pi*x)*exp(-x)    # definiert eine neue Funktion f
        g(x)=exp(-x)                # definiert eine weitere Funktion g
        pretty_print(f)             # Schöne Ausgabe

x |--> e^(-x)*sin(4*pi*x)


In [10]: plot([f,g,-g],xmin=0,xmax=6) # Graphische Darstellung von f, g und -g

Out[10]:
```



Ein wichtiger Befehl ist help. Dieser zeigt einen kurzen Hilfetext zu den implementierten Funktionen an. Dort findet man auch eine Auflistung aller Funktionsparameter (z.B. xmin, xmax aus dem letzten Beispiel). Geben Sie

```
help(plot)
```

ein und sehen Sie selbst.
Sage kann auch **Nullstellen** berechnen, **differenzieren** und **integrieren**. Lesen Sie dazu

```
help(f.roots)
help(f.derivative)
help(f.integral)
```

um zu sehen, wie das geht.

```
In [11]: # Nullstellen berechnen. Dies gibt eine Liste aus Tupeln
         # (bestehend aus der Nullstelle und deren Vielfachheit) zurück
         xr=f.roots()
         # Erste Nullstelle ausgeben (d.h. 1. Element der Liste,
         # und 1. Element des Tupels)
         print("Nullstelle:", xr[0][0])

Nullstelle: 0


In [12]: f.derivative() # Ableitung nach x
```

```
Out[12]: x |--> 4*pi*cos(4*pi*x)*e^(-x) - e^(-x)*sin(4*pi*x)

In [13]: f.integral(x) # Stammfunktion

Out[13]: x |--> -(4*pi*cos(4*pi*x) + sin(4*pi*x))*e^(-x)/(16*pi^2 + 1)
```

Einfache Schleifen und Fallunterscheidungen

Sage kann auch wie eine Programmiersprache verwendet werden und stellt wie (fast) jede Programmiersprache Fallunterscheidungen und Schleifen bereit. Die Syntax ist dabei ähnlich wie in der Programmiersprache Python. Die Syntax für Fallunterscheidungen lautet

```
if <condition>:
    do something
elif <another condition>:
    do something
else:
    do something
```

Beachte: Die Einrückung des Codes ist hier wichtig. Sie bestimmt, welche Zeilen zum Rumpf der einzelnen Fälle gehören.

```
In [14]: a=-2
         if a>0:                    # Hier steht Bedingung 1
             print("a ist positiv") # wird ausgeführt, falls die Bedingung erfüllt ist
         elif a<0:                  # Hier steht Bedingung 2
             print("a ist negativ") # wird ausgeführt, falls Bedingung 2 erfüllt
         else:                      # Falls beide Bedingungen nicht erfüllt
             print("a ist gleich null")

a ist negativ
```

Ein weiteres Konzept, welches in so gut wie jeder Programmiersprache vorzufinden ist, sind Schleifen. Es gibt verschiedene Arten von Schleifen, welche je nach Anwendung entsprechend gewählt werden müssen. Am einfachsten sind for-Schleifen, welche folgende Syntax besitzen:

```
for <element> in <iterable object>:
    do something for <element>
    do something else for <element>
```

Beachte: Auch hier ist die Einrückung des Schleifenblocks wichtig. Wäre die zweite Zeile im Rumpf nicht mehr eingerückt, würde sie nicht mehr zur Schleife gehören.
Ein iterierbares Objekt ist beispielsweise eine Liste. In Sage erkennt man Listen meist an den eckigen Klammern. So gibt beispielsweise der roots-Befehl eine Liste zurück:

```
In [15]: f=x^2-1
         # Berechnet Nullstellen ohne deren Vielfachheit
         f_roots = f.roots(multiplicities=False)
         f_roots

Out[15]: [-1, 1]
```

Über diese können wir mit einer for-Schleife drüber iterieren:

```
In [16]: for root in f_roots:
             print(root, "ist eine Nullstelle von f.")

-1 ist eine Nullstelle von f.
1 ist eine Nullstelle von f.
```

Folgendes Beispiel berechnet die Summe aller Listeneinträge:

```
In [17]: liste=[1,2,3]         # Definiert eine Liste mit Einträgen 1, 2 und 3
         summe=0               # legt neue Variable an
         for nummer in liste: # Führt folgenden Block für alle Elemente der Liste aus
             print("Die Zahl", nummer, "steht in der Liste")
             summe=summe+nummer
         print("Die Summe ist", summe)

Die Zahl 1 steht in der Liste
Die Zahl 2 steht in der Liste
Die Zahl 3 steht in der Liste
Die Summe ist 6
```

Beachte: Die letzte Zeile ist nicht mehr eingerückt und gehört damit nicht mehr zur Schleife.
Abschließender Hinweis: Google is your friend! Wenn Sie nicht wissen, wie Sie ihr Vorhaben in Sage umsetzen können, können Sie davon ausgehen, dass irgendjemand auf der Welt bereits schon vor dem gleichen Problem stand und jemand anderes im Internet eine Lösung veröffentlicht hat. Recherchieren Sie in diesem Fall zunächst also selbst nach Lösungen für Ihre Probleme. Falls dies nicht erfolgreich ist, Fragen Sie gern im Stud.IP-Forum der Lehrveranstaltung nach.

Hausaufgaben

Übungsaufgabe 1.1 [Quantile der Normalverteilung]

Die Dichte-Funktion der Standardnormalverteilung, die sogenannte **Gauß-Glocke** ist definiert durch

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}.$$

(a) Berechne die Stammfunktion von f. Finde über den help-Befehl heraus, was die erf-Funktion ist.

(b) Die Verteilungsfunktion-Funktion ist definiert durch $F(x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t$.

Berechne den Wert der Verteilungsfunktion an den Stellen $x = 0$ und $x = 1$. Geben Sie auch den numerischen Wert auf 20 Stellen genau aus. Wie sie ein unbestimmtes Integral (also ein Integral mit Integrationsgrenzen $\pm\infty$) berechnen lassen, verrät Ihnen der `help`-Befehl.

Erläuterung: Der Wert $F(x)$ ist bei einer standardnormaverteilten Zufallsgröße gerade die Wahrscheinlichkeit, dass das Ergebnis eines Zufallsversuchs kleiner als x ist.

(c) Berechne auch $\int_{-\infty}^{\infty} f(x)$ (also $F(\infty)$). Kommt der zu erwartende Wert heraus?

(d) Das α-Quantil $z_{1-\alpha}$ der Normalverteilung ist definiert durch

$$F(z_{1-\alpha}) = 1 - \alpha.$$

Berechne $z_{0.95}$. Geben Sie den numerischen Wert aus.

Erläuterung: Die Wahrscheinlichkeit, dass das Ergebnis des Zufallsexperimentes kleiner als die gesuchte Größe $z_{1-\alpha}$ ist, ist gerade $(1-\alpha) * 100\%$.

Hinweis: Nutzen Sie den solve-Befehl. Dieser gibt eine Liste aus Gleichungen zurück. An den gesuchten Wert kommen Sie mit `[0].rhs()`, wobei rhs für "right-hand side" (also die rechte Seite der Gleichung) steht.

Übungsaufgabe 1.2 [Kurvendiskussion für gebrochen rationale Funktionen]

Wir wollen die Funktion

$$f(x) = \frac{x^2}{2(x-1)^2}$$

genauer untersuchen.

(a) Berechnen Sie die Nullstellen und deren Vielfachheit. Geben Sie das Ergebnis in einer formatierten Ausgabe über den print-Befehl aus. Das Programm sollte beispielsweise Sätze der Form

```
1 ist eine Nullstelle der Vielfachheit 2.
```

ausgeben. Beachten Sie, dass die Funktion auch mehrere Nullstellen haben kann. Der roots-Befehl gibt ihnen daher eine Liste zurück und Sie brauchen eine for-Schleife für die Auswertung (siehe Theorieteil).

(b) Berechnen Sie alle Polstellen.

Hinweis: Polstellen von f sind Nullstellen der Funktion $1/f$.

(c) Berechne Lage und Art aller lokalen Extremstellen. Lösen Sie dazu mit dem `f.roots()`-Befehl die notwendige Optimalitätsbedingung

$$f'(x) = 0.$$

Implementieren Sie eine Schleife über alle Nullstellen und untersuchen für jede Nullstelle die hinreichenden Optimalitätsbedingungen um festzustellen, ob es sich um ein Maximum, Minimum oder Wendepunkt handelt. Geben Sie das Ergebnis wie in Aufgabe a) formatiert aus.

(d) Berechnen Sie die schrägen Asymptoten von f, also lineare Funktionenen $g(x) = m\,x + n$, welche

$$\lim_{x \to \pm\infty} (f(x) - g(x)) = 0$$

erfüllen.

Hinweis: Den Anstieg einer Asymptote berechnet man über $m = \lim_{x\to\pm\infty} f(x)/x$. Lesen Sie sich dazu die Hilfe des limit-Befehls durch. Die Verschiebung erhält man über $n = \lim_{x\to\pm\infty}(f(x) - m\,x)$.

(e) Stellen Sie die Funktion f und die berechneten Asymptoten grafisch in einem gemeinsamen Plot dar. Der Plot-Bereich sollte

$$x \in [-3, 5], \qquad f(x) \in [-5, 5]$$

sein. Suchen Sie im Hilfetext von plot eine Möglichkeit, sodass die Polstelle nicht mitgezeichnet wird.

(f) Ihr Skript sollte generisch programmiert sein, sollte also so funktionieren, dass man lediglich am Anfang des Skripts die Definition von f ändern muss, um für eine neue Funktion die entsprechenden Ergebnisse zu bekommen. Testen Sie dies für

$$f(x) = \frac{x-1}{x^2+2\,x-1}.$$

Hinweise zur Abgabe:
Bitte speichern Sie ihr Lösungsskript als Jupyter-Notebook (Dateiendung .ipynb) ab. Benennen Sie die Datei nach folgendem Schema:

`[Nachname]_[Vorname]_Hausaufgabe_1.ipynb`

Laden Sie diese Datei anschließend im Stud.IP in den Hausaufgaben-Ordner ihrer entsprechenden Übungsgruppe hoch. Die Abgabe kann in 3er-Gruppen erfolgen. Die Namen aller Mitarbeiter sollten am Anfang des Skripts in eine Markdown-Zelle geschrieben werden.

Abgabe bis: 04.05.2020