Einführung in Sage

Einheit 1

Jochen Schulz

Georg-August Universität Göttingen

7. Februar 2010

Organisatorisches

- Anmeldungen zu der Veranstaltung über StudIP
  `https://www.studip.uni-goettingen.de/`
  Einführung in Sage (Mathematische Anwendersysteme) (WS 2009/2010)
- Alle Unterlagen (Aufgabenblätter, Vorlesungsfolien, Beispiele, Musterlösungen) können von der StudIP-Seite (Reiter Dateien) heruntergeladen werden

Organisatorisches

- Anmeldungen zu der Veranstaltung über StudIP
  `https://www.studip.uni-goettingen.de/`
  Einführung in Sage (Mathematische Anwendersysteme) (WS 2009/2010)
- Alle Unterlagen (Aufgabenblätter, Vorlesungsfolien, Beispiele, Musterlösungen) können von der StudIP-Seite (Reiter Dateien) heruntergeladen werden

Dozent

Jochen Schulz
NAM, Zimmer 04 (Erdgeschoß)
Telefon: 39-4525 Email: `schulz@math.uni-goettingen.de`
XMPP: `schulz@jabber.num.math.uni-goettingen.de`

Starten des Programms

Vor.: Account im CIP-Pool der Mathematischen Fakultät (MI und NAM): Registrierungs-Formular unter `https://ldap.math.uni-goettingen.de` (Stud.It-Account nötig!)
Wiki (`https://wiki.math.uni-goettingen.de/mediawiki`)

- Sage ist in Version 4.2.1 auf allen Rechnern der Mathematischen Fakultät installiert
- login direkt oder per nxclient auf den Rechnern `s1.math.uni-goettingen.de` bis `s8.math.uni-goettingen.de` und `s241.math.uni-goettingen.de` bis `s245.math.uni-goettingen.de`
- Starten von Sage: Im Terminal
  - `sagenotebook` (Notebook)
  - `sage` (Kommandozeilen-Version)

Ablauf der Veranstaltung

- Blockveranstaltung vom 15.2.-26.2.2010
- Vorlesung: 9.15 Uhr bis 11.30 Uhr
- Nachmittags: 4 Übungsgruppen à je 1h 15min
  - 13:00-14:15 (Tutor: J. Schulz)
  - 14:15-15:30 (Tutor: C. Rügge)
  - 15:30-16:45 (Tutor: J. Schulz)
  - 16:45-18:00 (Tutor: C. Rügge)
- Scheinerwerb
  - Regelmäßige Teilnahme an den Übungen
  - Projektarbeit durchführen
  - Klausur am 1.3.2009; 10:00 - 12:00; Anmeldung über FlexNow!

Inhalt der Vorlesung

1. Tag Organisatorisches, Aufbau von Sage, Streifzug durch Sage

Inhalt der Vorlesung

1. Tag Organisatorisches, Aufbau von Sage, Streifzug durch Sage

2. Tag Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen

Inhalt der Vorlesung

1. Tag Organisatorisches, Aufbau von Sage, Streifzug durch Sage

2. Tag Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen

3. Tag Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken
8. **Tag** Differenzial- und Integralrechnung

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken
8. **Tag** Differenzial- und Integralrechnung
9. **Tag** Grundlagen der Programmierung, Zeichenketten (Strings)

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen, Ungleichungen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken
8. **Tag** Differenzial- und Integralrechnung
9. **Tag** Grundlagen der Programmierung, Zeichenketten (Strings)
10. **Tag** Fragestunde

Aufbau

1. **Was ist Sage?**

2. **Streifzug durch Sage**
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie

3. **Nützliches und Hilfe**

Aufbau

1. **Was ist Sage?**

2. **Streifzug durch Sage**
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie

3. **Nützliches und Hilfe**

Computeralgebra-Systeme

Computeralgebra

beschäftigt sich mit exakten Berechnungen von mathematischen Objekten

Mathematische Objekte

Natürliche Zahlen, reelle Zahlen, Polynome, Funktionen, Gruppen, Ringe, ...

Numerischen Berechnungen

Bei numerischen Rechnungen (z.B. Taschenrechner) benutzt man Zahlen in Gleitpunktdarstellung, also i.A. nur Näherungen an die gesuchte Lösung

Computeralgebra != Numerische Berechnung

Beispiel

Mathematische Objekte	π, $\sqrt{2}$
Gleitpunktdarstellung (8 Stellen)	3.1415927, 1.4142136

Computeralgebra-Systeme

Allgemein

Sage	Schnittstelle für Mathematik-Software
LiveMath	Maple
Maxima	Free, GPL, von Sage benutzt
Mathematica	Platzhirsch
Maple	Platzhirsch
Matlab/Octave	Für große Rechnungen, inkl. Mupad
Magma	Spezielle mathematische Rechungen
SymPy	Phython-Bibliotheken als CAS-Verwendbar
SymbolicC++	Bibliotheken zur CA in C++

Überblick:`http://en.wikipedia.org/wiki/Comparison_of_computer_algebra_systems`

Sage

- Open-source (GPL) Mathematik Software System.
- Alternative zu den 4 M's: Magma, Maple, Mathematica, Matlab.
- Basiert auf Python.
- Objektorientiert.
- Besitzt Frontends für viele externe Software $\longrightarrow$ Synergie-Effekte

von Joachim Neubüser (Gründer von GAP):

> *You can read Sylow's Theorem and its proof [. . .] and then you can use Sylow's Theorem for the rest of your life free of charge, but for many computer algebra systems license fees have to be paid regularly [. . .]. You press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.*
>
> *With this situation two of the most basic rules of conduct in mathematics are violated: in mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research [. . .] means moving in a most undesirable direction. Most important: can we expect somebody to believe a result of a program that he is not allowed to see?*

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem
- Einfaches Einbinden von C/C++ Routinen (dynamische Module)

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem
- Einfaches Einbinden von C/C++ Routinen (dynamische Module)
- Schnittstelle zu vielen anderen CAS (Maxima, Pari, GAP, R, Magma, ..., wovon einige bereits bei Sage enthalten sind)

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem
- Einfaches Einbinden von C/C++ Routinen (dynamische Module)
- Schnittstelle zu vielen anderen CAS (Maxima, Pari, GAP, R, Magma, ..., wovon einige bereits bei Sage enthalten sind)
- Viele freie (Unterrichts-)materialien im Netz

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem
- Einfaches Einbinden von C/C++ Routinen (dynamische Module)
- Schnittstelle zu vielen anderen CAS (Maxima, Pari, GAP, R, Magma, ..., wovon einige bereits bei Sage enthalten sind)
- Viele freie (Unterrichts-)materialien im Netz

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem
- Einfaches Einbinden von C/C++ Routinen (dynamische Module)
- Schnittstelle zu vielen anderen CAS (Maxima, Pari, GAP, R, Magma, ..., wovon einige bereits bei Sage enthalten sind)
- Viele freie (Unterrichts-)materialien im Netz

Schwächen

- Befehlsumfang nicht so mächtig wie bei Maple oder Mathematica

Sage - Stärken und Schwächen

Stärken

- interaktiver Quellcode-Debugger
- umfangreiches Hilfesystem
- Einfaches Einbinden von C/C++ Routinen (dynamische Module)
- Schnittstelle zu vielen anderen CAS (Maxima, Pari, GAP, R, Magma, ..., wovon einige bereits bei Sage enthalten sind)
- Viele freie (Unterrichts-)materialien im Netz

Schwächen

- Befehlsumfang nicht so mächtig wie bei Maple oder Mathematica
- es fehlt eine gute Entwicklungsumgebung

Kern von Sage ??

- Parser: Liest die Eingaben und überprüft die Syntax; Umwandlung in MuPAD-Datentyp
- Auswerter: Auswertung und Vereinfachung der Ergebnisse
- Speicherverwaltung: (MAMMUT $\equiv$ Memory Allocation Managment Unit) interne Verwaltung der MuPAD-Objekte
- Kernfunktionen: Oft benötigte Funktionen werden aus Effizienzgründen im Kern auf C-Ebene implementiert.

Aufbau

1. Was ist Sage?

2. **Streifzug durch Sage**
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie

3. Nützliches und Hilfe

Sage als Taschenrechner

Hier einige Beispiele:

```
>> 3+4*10+12
```

```
  55
```

```
>> sin(pi)
```

```
  0
```

```
>> float(pi)
```

```
  3.14159265358979
```

```
>> float(sqrt(2))
```

```
  1.41421356237310
```

Aufbau

1. Was ist Sage?

2. **Streifzug durch Sage**
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie

3. Nützliches und Hilfe

Kurvendiskussion I

Betrachte die durch die reelle Zahl a parametrisierte Funktionenschar:

$$f : x \quad \mapsto \quad \frac{2x^2 - 20x + 42}{x - 1} + a, \quad a \in \mathbb{R}$$

- Eingabe der Funktion

```
>> var('a')
>> f(x) = (2*x^2-20*x +42)/(x-1)+a
```

```
  x |--> a + 2*(x^2 - 10*x + 21)/(x - 1)
```

Kurvendiskussion II

- Pol ?

```
>> f.limit(x=1, dir='minus')
```

```
  x |--> -Infinity
```

```
>> f.limit(x=1, dir='plus')
```

```
  x |--> +Infinity
```

- Umformen

```
>> f.full_simplify()
```

```
x |--> ((a - 20)*x + 2*x^2 - a + 42)/(x - 1)
```

Kurvendiskussion III

- Nullstellen

```
>> solve(f==0,x)
```

```
[x == -1/4*a - 1/4*sqrt(a^2 - 32*a + 64) + 5, x == -1/4*a + 1/4*
    sqrt(a^2 - 32*a + 64) + 5]
```

- Berechnen der Ableitung

```
>> f.differentiate(x)
```

```
x |--> 4*(x - 5)/(x - 1) - 2*(x^2 - 10*x + 21)/(
   x - 1)^2
```

Kurvendiskussion IV

- Extremwerte

```
>> solve(f.differentiate(x)==0,x)
```

```
[x == -2*sqrt(3) + 1, x == 2*sqrt(3) + 1]
```

- Lokale Minima und Maxima

```
>> float( ((f.diff(x)).diff(x))(-2*sqrt(3)+1) )
```

```
-1.1547005383792501
```

```
>> float( ((f.diff(x)).diff(x))(2*sqrt(3)+1) )
```

```
1.1547005383792515
```

Kurvendiskussion V

- Verhalten von f für große x

```
>> f.limit(x=oo); f.limit(x=-oo)
```

```
x |--> +Infinity
x |--> -Infinity
```

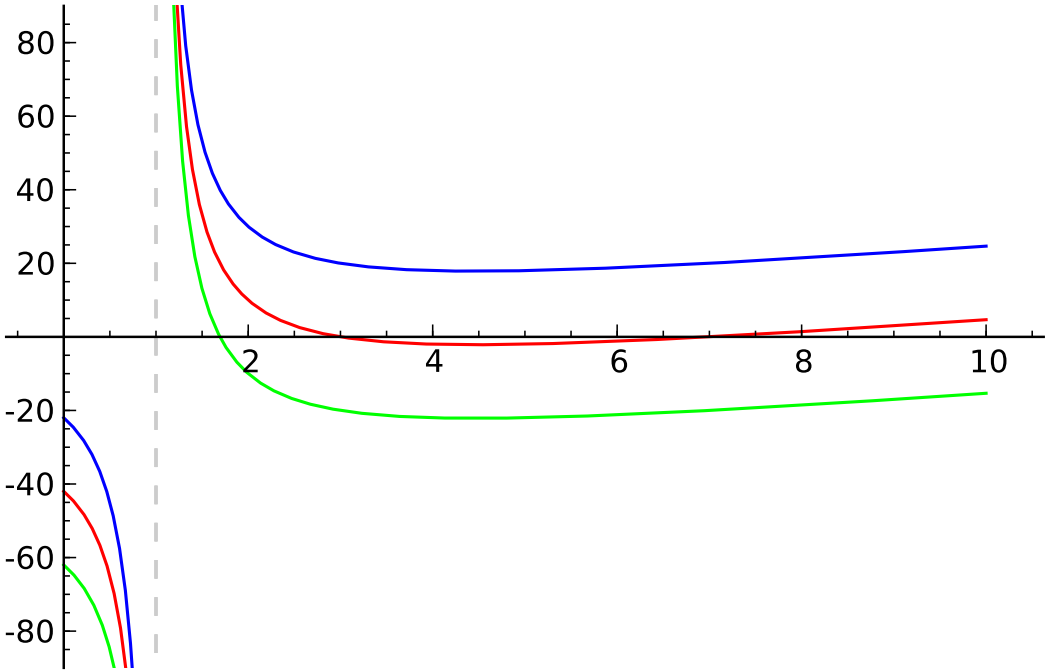
- Definiere f_0, f_1, f_2

```
>>  f0 = f(x, a=0)
>>  f1 = f(x, a=-20)
>>  f2 = f(x, a=20);f0,f1,f2
```

```
(2*(x^2 - 10*x + 21)/(x - 1), 2*(x^2 - 10*x + 21)/(x - 1) - 20,
    2*(x^2 - 10*x + 21)/(x - 1) + 20)
```

Plot

```
p = plot(f0,detect_poles='show',xmin=0, xmax=10,
    color='red')
p += plot(f1,detect_poles='show',xmin=0, xmax=10,
    color='green')
p += plot(f2,detect_poles='show',xmin=0, xmax=10,
    color='blue'); p.show(ymin=-80, ymax=80)
```



Zusammenfassung

- Deklarieren von Variablen mir `var()`, z.B.`var('a')`
- Definieren von Variablen mit '=', z.B. `a=3`
- Definieren von Funktionen mit'=', z.B. `f(x) = x^2 - 6*x`
- Symbolisches Rechnen
  - Grenzwertbestimmung: `f.limit(x=1, dir='<plus|minus>')`
  - Vereinfachen: `f.full_simplify()`
  - Bilden von Ableitungen `f.differentiate(x)`
- Lösen von Gleichungen: `solve( f(x)==0, x)`
- Berechnen numerischer Approximationen: `float(f(sqrt(3)+ 4))`
- Plotten einer Funktion: `plot(sin,(0,4))`

Aufbau

1. Was ist Sage?

2. **Streifzug durch Sage**
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie

3. Nützliches und Hilfe

Symbolisches Rechnen I

- Integrieren von $\int_0^\infty x^4 e^{-x} dx$

```
>> integrate(x^4*exp(-x),x,0,oo)
```

```
  24
```

- Stammfunktion von $\frac{1+\sin(x)}{1+\cos(x)}$

```
>> f(x) = (1+sin(x))/(1+cos(x))
>> g = f.integrate(x)
>> g.full_simplify()
```

```
x |--> -((cos(x) + 1)*log(cos(x) + 1) - sin(x))/(cos(x) + 1)
```

Symbolisches Rechnen II

- Faktorisieren und Ausmultiplizieren

```
>> expand((x-1)*(x-2)*(x-3)*(x-4))
```

```
x^4 - 10*x^3 + 35*x^2 - 50*x + 24
```

```
>> factor(_)
```

```
(x - 4)*(x - 3)*(x - 2)*(x - 1)
```

- Sortieren eines Ausdrucks bezüglich einer Unbekannten

```
var('b,a')
g = x^2+2*x+b*x^2+sin(x)+a*x
g.collect(x)
```

```
(b + 1)*x^2 + (a + 2)*x + sin(x)
```

Symbolisches Rechnen III

- Partialbruchzerlegung

```
>>g = x^ 2/( x^ 2- 1)
g.partial_fraction()
```

```
1/2/(x - 1) - 1/2/(x + 1) + 1
```

- Vereinfachen von Ausdrücken ($\frac{e^x-1}{e^{(1/2)x}+1}$)

```
>> g = (exp(x)-1)/(exp(x/2)+1); g.
   simplify_radical()
```

```
 e^(1/2*x) - 1
```

Aufbau

1. Was ist Sage?
2. **Streifzug durch Sage**
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie
3. Nützliches und Hilfe

Berechnen des Schnittpunkts der Ebene

$$E : \vec{x} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} + l \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + m \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix}, \quad l, m \in \mathbb{R}$$

mit der Geraden

$$g : \vec{x} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} + k \begin{pmatrix} 4 \\ -1 \\ 2 \end{pmatrix}, \quad k \in \mathbb{R}$$

Grafische Darstellung

```
var('l,m'); E1 = 2+l-3*m; E2 = 1-l+m; E3 =-1-l+4*m
```

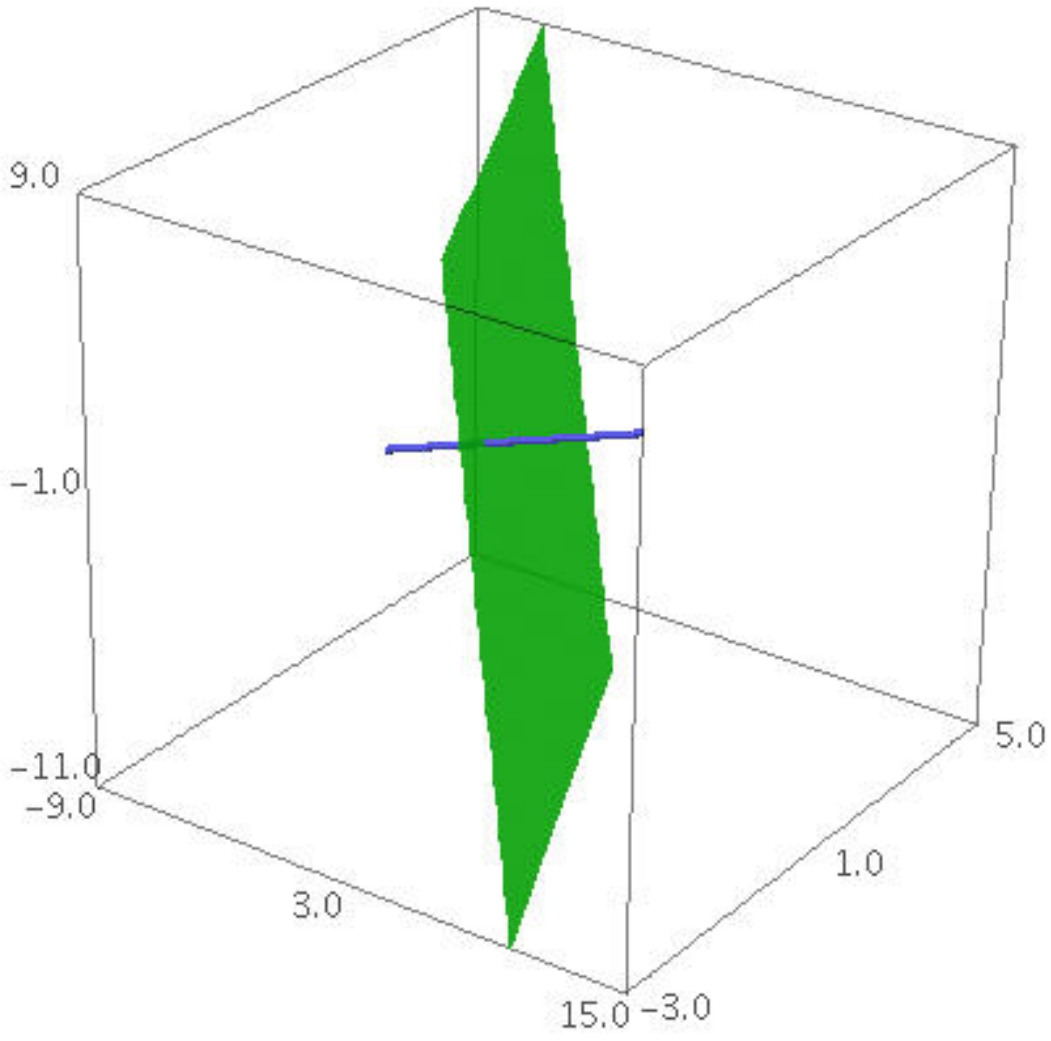
```
>> p = parametric_plot3d([E1,E2,E3],(l,-2,2),(m
   ,-2,2), color='green', opacity=0.8)
```

```
>> var('k'); g1 = 3+4*k; g2 = -k; g3 = 1+2*k
```

```
>> p += parametric_plot3d( (g1,g2,g3), (k, -3, 3),
   thickness='3' )
```

```
>> p.show()
```

Grafische Darstellung



Analytische Lösung

Gleichsetzen ergibt:

$$\begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} + l \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + m \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} + k \begin{pmatrix} 4 \\ -1 \\ 2 \end{pmatrix}$$

oder

$$\underbrace{\begin{pmatrix} 1 & -3 & -4 \\ -1 & 1 & 1 \\ -1 & 4 & -2 \end{pmatrix}}_{=: A} \underbrace{\begin{pmatrix} l \\ m \\ k \end{pmatrix}}_{=: L} = \underbrace{\begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}}_{=: b}$$

oder $AL = b$.

Definieren und Lösen des LGS

- Definieren der Matrix A

```
>> A = matrix([[1,-3,-4],[-1,1,1],[-1,4,-2]]); A
```

```
[ 1 -3 -4]
[-1  1  1]
[-1  4 -2]
```

- Definieren des Vektors b

```
>> b = vector([1,-1,2])
```

- Lösen von $A\ L = b$

```
A.solve_right(b)
```

oder

```
A\b
```

ergibt

```
(6/5, 3/5, -2/5)
```

- Einsetzen in die Geradengleichung

```
>> x_s = matrix([g1,g2,g3]).subs(k=L[2]); x_s
```

```
[7/5 2/5 1/5]
```

- Matrizenoperationen

```
>> B = matrix([[1,0,0],[0,1,1],[1,1,1]])
>> A*B; A-B; A+B
```

```
[-3 -7 -7]  [ 0 -3 -4] [ 2 -3 -4]
[ 0  2  2]  [-1  0  0] [-1  2  2]
[-3  2  2]  [-2  3 -3] [ 0  5 -1]
```

- Berechnen der Inversen (mit Probe)

```
>> A^(-1), A*A^(-1)
```

```
[  -2/5 -22/15   1/15]
[  -1/5   -2/5    1/5]
[  -1/5  -1/15  -2/15]

[1 0 0]
[0 1 0]
[0 0 1]
```

Aufbau

1. Was ist Sage?

2. **Streifzug durch Sage**
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie

3. Nützliches und Hilfe

Etwas Zahlentheorie I

Fermatsche Primzahlen: $F_n = 2^{2^n} + 1$. Finden Sie die kleinste Zahl F_n, die keine Primzahl ist!

```
>> def F(n): return 2^(2^n)+1
>> [[F(m),is_prime(F(m))] for m in xrange(1,6) if
   is_prime (F(m))]
```

```
[[5, True], [17, True], [257, True], [65537, True]]
```

```
>> divisors(int(F(5)))
```

```
[1, 641, 6700417, 4294967297]
```

Etwas Zahlentheorie II

- Eine Liste der ersten Primzahlen bis 100

```
>> menge = set(range(1,101))
>> [m for m in menge if is_prime(m)]
oder >> filter(is_prime,menge)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
    43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89,
   97]
```

- Mersenne-Primzahlen $2^p - 1$, p Primzahl. Bestimmen der ersten Mersenne Primzahlen im Bereich ≤ 200.

```
>> menge = set(range(1,201))
>> primes = [m for m in menge if is_prime(m)]
>> [m for m in primes if is_prime(2^m-1)]
```

```
[2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127]
```

Etwas Zahlentheorie III

Wir geben für die natürlichen Zahlen ≤ 1000 an, wieviele Zahlen $1, 2, 3, \ldots$ Teiler haben.

```
>> menge = set(range(1,1001))
>> liste = [number_of_divisors(int(m)) for m in
   menge]
>> for i in range(1,50):
>>     print (i,len([m for m in liste if m==i]))
```

```
(1, 1)
(2, 168)
(3, 11)
(4, 292)
...
```

Teiler der Zahl 840:

```
>> divisors(840)
```

Aufbau

1. Was ist Sage?

2. Streifzug durch Sage
   - Eine Kurvendiskussion
   - Symbolisches Rechnen
   - Etwas AGLA
   - Etwas Zahlentheorie

3. **Nützliches und Hilfe**

Überlebensregeln

- Mehrere Befehle in einer Zeile durch ';' trennen.
- Bei Eingaben, die über mehrere Zeilen gehen, kann ein Zeilenumbruch durch <ENTER> erreicht werden.
- Das Auswerten eines Blocks erfolgt mit <SHIFT>+<ENTER>.
- listen, tuple, for, if

Nützliches

- _ refenziert die letzte Ausgabe.
- Löschen aller eigenen Variablen und Zurücksetzen auf den Anfangsstatus: reset()
- Das Feld aktivieren von Typeset lässt alle Ausgaben ``schön'' erscheinen (LaTeXgerendert)
- Html- LaTeX-Dokumentation: `shift+klick` auf den blauen Balken über einer Textbox.
- publish

Hilfefunktionen in Sage

- Autocompletion : mit der <TAB>-Taste erhält man alle möglichen Funktionen und/oder Variablen im gegebenen Kontext.
- *command*? : gibt ausführliche Hilfe zu *command* an.
- help(*command*) : öffnet ein Hilfefenster zu *command*.
- Hilfe für eine Funktion: `limit?`
- Vorhandene Unterfunktionen: `expr.<tab key>`