

# Einführung in Sage

## Einheit 4

Jochen Schulz

Georg-August Universität Göttingen 

28. Januar 2010

## 1 Vektoren

- Matrizen
- Vektorräume

## 2 Etwas Programmieren

## 1 Vektoren

- Matrizen
- Vektorräume

## 2 Etwas Programmieren

## 1 Vektoren

- Matrizen
- Vektorräume

## 2 Etwas Programmieren

Eine  $m \times n$  Matrix  $A$  über einen Körper  $K$  ist ein rechteckiges Schema mit Einträgen  $a_{ij} \in K$ ,  $1 \leq i \leq m$ ,  $1 \leq j \leq n$  der Form

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

mit dem Zeilenindex  $i$  mit Werten zwischen 1 und  $m$  und Spaltenindex  $j$  mit Werten zwischen 1 und  $n$ . Man schreibt kurz  $A = (a_{ij}) \in K^{m \times n}$ .

- Die **Transponierte** von  $A = (a_{ij})$  ist  $A^T := (a_{ji})$ .
- $A$  heißt **symmetrisch**, wenn  $A = A^T$  gilt.
- Für Matrizen  $A = (a_{ij}) \in \mathbb{C}^{m \times n}$  ist  $A^* := (\overline{a_{ji}}) \in \mathbb{C}^{n \times m}$ .
- Die **Einheitsmatrix**  $I := I_n := (\delta_{ij}) \in K^{n \times n}$
- Seien  $A = (a_{ij}), B = (b_{ij}) \in K^{n \times m}$ . Dann ist die **Addition** definiert durch

$$C = (c_{ij}) := A + B \in K^{n \times m}$$

mit  $c_{ij} = a_{ij} + b_{ij}$ .

- Seien  $A = (a_{ij}) \in K^{m \times n}$  und  $B = (b_{ij}) \in K^{n \times p}$ . Dann ist die **Multiplikation** gegeben durch

$$C = (c_{ij}) := A \cdot B \in K^{m \times p}$$

mit  $c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$ .

- $A \in K^{n \times n}$  heißt **orthogonal**, wenn  $A \cdot A^T = A^T \cdot A = I_n$  gilt.
- $A \in \mathbb{C}^{n \times n}$  heißt **unitär**, wenn  $A \cdot A^* = A^* \cdot A = I_n$  gilt.
- $A \in K^{n \times n}$  heißt **invertierbar**, wenn eine Matrix  $A^{-1} \in K^{n \times n}$  existiert mit  $A \cdot A^{-1} = A^{-1} \cdot A = I_n$ .

# Definitionen und Bemerkungen

- Die Multiplikation ist assoziativ aber in der Regel **nicht kommutativ**.
- Die Matrizen aus  $K^{m \times n}$  bilden einen Vektorraum über  $K$  (mit komponentenweiser Skalarmultiplikation).
- Die Menge der invertierbaren Matrizen aus  $K^{n \times n}$  bilden bezüglich der Multiplikation eine Gruppe, die **allgemeine lineare Gruppe**  $GL(K, n) = GL_n(K) = GL(n, K)$ .



# Definition und Bemerkungen

- Die Menge der orthogonalen Matrizen in  $GL(\mathbb{R}, n)$  bilden eine Untergruppe von  $GL(\mathbb{R}, n)$ , die **orthogonale Gruppe**  $O(n)$ .
- Die entsprechende Untergruppe der unitären Matrizen in  $GL(\mathbb{C}, n)$  ist die **unitäre Gruppe**  $U(n)$ .

# Matrizen in Sage

- Matrizen werden in Sage mit Hilfe des Befehls `matrix()` konstruiert.
- Der Rückgabewert ist vom Typ `matrix`.
- Die Einträge der Matrix können beliebige Ausdrücke sein.
- Es ist auch möglich Matrizen über bestimmten Bereichen (z.B.  $\mathbb{R}$ ,  $\mathbb{C}$ ,  $\mathbb{Z}$ ) zu konstruieren.
- Es gibt auch spezielle Datenstrukturen für quadratische Matrizen und für dünnbesetzte Matrizen.

# Konstruktion von Matrizen I

Es gibt in Sage verschiedene Möglichkeiten eine Matrix zu konstruieren.

Beispiel:

$$A := \begin{pmatrix} 1 & 2 & 3 & 4 \\ a & 0 & 1 & b \end{pmatrix}, \quad a, b \in \mathbb{R}$$

- Eingabe der Einträge pro Zeile in eckigen Klammern [..]. Alle Spalten dann wieder in eckigen Klammern [..].

```
>> A = matrix([[1, 2, 3, 4], [a, 0, 1, b]])
```

- Mit expliziter Größenangabe

```
>> A = matrix(2,4,[[1,2,3,4], [a,0,1,b]])
```

# Konstruktion von Matrizen II

- Erzeugen einer Nullmatrix der Größe  $n \times m$

```
>> n = 3; m = 4; B = matrix(n,m)
```

- Erzeuge eine  $n \times m$  - Matrix  $A$  mit Hilfe einer Funktion  $f(i,j)$  mit Einträgen  $a_{ij} = f(i,j)$

```
>> f(i,j) = i*j  
>> C = matrix([[f(i,j) for i in range(1,6)] for  
               j in range(1,4)]); C
```

- Eingabe von Zeilen- und Spaltenvektoren

```
>> matrix(3,1,[1,2,3])  
>> matrix(1,3,[4,5,6])
```

# Zeilen- und Spaltenzahl

- Abfragen der Spaltenanzahl: `<matrix>.ncols()`

```
>> C.ncols()
```

```
5
```

- Abfragen der Zeilenanzahl: `<matrix>.nrows()`

```
>> C.nrows()
```

```
3
```

- Informationen über die Matrix: `<matrix>.parent()`

```
>> C.parent()
```

```
Full MatrixSpace of 3 by 5 dense matrices over  
Symbolic Ring
```

# Zugriff auf die Einträge I

- Abfragen von Einträgen in Zeile  $i$  und Spalte  $j$ :

```
>> i=1; j=2; C[i,j]
```

```
6
```

- Ändern des Eintrags in Zeile  $i$  und Spalte  $j$ :

```
>> i=1; j=2; C[i,j]=22
```

- Extrahieren von Zeilen/Spalten

```
>> zeile = C.row(0)  
>> spalte = C.column(4)
```

- Extrahieren von Teilmatrizen

```
>> C[1:3,1:3]
```

- Erzeugen von Diagonalmatrizen

```
>> x = [1,2,3,4,5]  
>> Diag = diagonal_matrix(x); Diag
```

- Addieren, Multiplizieren

```
>> var('a,b,c,d,g,h,f')  
>> A = matrix([[a, b], [c,d]])  
>> B = matrix([[e, f], [g,h]])  
>> A+B; A*B
```

- Bestimmung der Inversen und der Transponierten

```
>> A^(-1); A.transpose()
```



# Rang von Matrizen

Sei  $A \in K^{m \times n}$ .

- Die Dimension der linearen Hülle der Spaltenvektoren nennt man den **Spaltenrang** von  $A$ . Er ist höchstens gleich  $n$ .
- Die Dimension der linearen Hülle der Zeilenvektoren nennt man den **Zeilenrang** von  $A$ . Er ist höchstens gleich  $m$ .
- Zeilenrang und Spaltenrang von Matrizen sind gleich und man spricht deshalb vom **Rang** einer Matrix.

- Bestimmen des Ranges einer Matrix

```
>> S = matrix([[1,0,0],[0,1,1],[1,1,1]])  
>> S.rank()
```

2

- ist S symmetrisch ? ist S invertierbar ?

```
>> S.is_symmetric()  
>> S.is_invertible()
```

False  
False

## 1 Vektoren

- Matrizen
- Vektorräume

## 2 Etwas Programmieren

Ein Tripel  $(V, +, \cdot)$ , bestehend aus einer nichtleeren Menge  $V$  und Verknüpfungen

$$+ : V \times V \rightarrow V, \quad \cdot : K \times V \rightarrow V$$

heißt **Vektorraum** über einem Körper  $K$ , wenn gilt:

- ❶  $(V, +)$  ist eine abelsche Gruppe.
- ❷ Für alle  $v, w \in V$  und alle  $\lambda, \mu \in K$  gilt:
  - ❶  $(\lambda + \mu) \cdot v = (\lambda \cdot v) + (\mu \cdot v)$ .
  - ❷  $\lambda \cdot (v + w) = (\lambda \cdot v) + (\lambda \cdot w)$ .
  - ❸  $(\lambda \mu) \cdot v = \lambda \cdot (\mu \cdot v)$ .
  - ❹  $1 \cdot v = v$ .

- Die Elemente eines Vektorraums nennt man **Vektoren**.
- Die Abbildung  $\cdot : K \times V \rightarrow V$  heißt **Skalarmultiplikation**. Die Elemente des Körpers  $K$  nennt man **Skalare**.
- Ist  $U \subset V$  eine Teilmenge des Vektorraums  $V$  und gelten alle Vektorraumaxiome, so heißt  $U$  ein **Untervektorraum** oder **Unterraum** von  $V$ .
- **Vorsicht!**  $0$  ist nicht gleich  $0$ , d.h. man muß zwischen der  $0$  des Körpers und der  $0$  des Vektorraums (Nullvektor) unterscheiden. Es gilt  $0 \cdot v = 0$  für alle  $v \in V$ .

# Beispiele für Vektorräume

- $K^n := \{(x_1, \dots, x_n) \mid x_1, \dots, x_n \in K\}, n \in \mathbb{N}$
- Sei  $M$  eine beliebige Menge. Die Menge der Abbildungen von  $M$  in  $K$ ,  $\text{Abb}(M, K)$ , mit den punktweise definierten Verknüpfungen

$$(f+g)(x) := f(x) + g(x), \forall x \in M$$

$$(\alpha \cdot f)(x) := \alpha \cdot f(x), \forall x \in M$$

für  $\alpha \in K, f, g: M \mapsto K$ .

- Die Menge der Polynome bis zum Grad  $n$ .
- Die Menge aller Polynome.
- $\mathbb{R}$  als  $\mathbb{Q}$ -Vektorraum.
- $\mathbb{C}$  als  $\mathbb{R}$ -Vektorraum.

# Vektoren in Sage

- Konstruktion von Vektoren

```
>> a = vector([1,2,3,4]); b = vector([5,6,7]); a  
      ,b
```

```
((1, 2, 3, 4), (5, 6, 7))
```

- Datentyp: `vector_integer_dense` (für dichte Vektoren)

```
>> type(a)
```

```
<type 'sage.modules.vector_integer_dense.Vector_integer_dense'>
```

# Lineare Abhängigkeit

Sei  $V$  ein  $K$ -Vektorraum und  $(v_1, \dots, v_r)$  eine Familie von Elementen aus  $V$ .

- $v \in V$  heißt **Linearkombination** von  $(v_1, \dots, v_r)$ , falls  $\exists \lambda_1, \dots, \lambda_r \in K$  mit  $v = \lambda_1 v_1 + \dots + \lambda_r v_r$ .
- Die Menge aller Linearkombinationen wird **Lineare Hülle** genannt und durch  $\text{span}\{v_1, \dots, v_n\}$  bezeichnet. Die Lineare Hülle ist ein Unterraum von  $V$ .
- $(v_1, \dots, v_r)$  heißen **linear unabhängig**, falls gilt: Sind  $\lambda_1, \dots, \lambda_r \in K$  und ist  $\lambda_1 v_1 + \dots + \lambda_r v_r = 0$  so folgt  $\lambda_1 = \dots = \lambda_r = 0$ . Andernfalls sind sie **linear abhängig**.
- Ist  $M \subseteq V$  eine unendliche Menge, dann ist  $M$  linear unabhängig falls **alle endlichen** Teilmengen von  $M$  linear unabhängig sind.



# Weitere Notationen und Bemerkungen

Sei  $V$  ein  $K$ -Vektorraum und  $(v_1, \dots, v_r)$  eine Familie von Elementen aus  $V$

- $(v_1, \dots, v_r)$  sind genau dann linear unabhängig, wenn sich jeder Vektor  $v \in \text{span}\{v_1, \dots, v_r\}$  eindeutig linear kombinieren läßt.
- Gilt  $V = \text{span}\{v_1, \dots, v_r\}$ , so ist  $(v_1, \dots, v_r)$  ein **Erzeugendensystem**. Sind  $(v_1, \dots, v_r)$  zusätzlich linear unabhängig, so ist  $(v_1, \dots, v_r)$  eine **Basis**.
- Aus jedem Erzeugendensystem kann man eine Basis auswählen.

# Beispiele für Basen

- Seien  $(e_i)_{i=1,\dots,n} \in \mathbb{R}^n$  die Einheitsvektoren.  $(e_1, \dots, e_n)$  ist eine Basis des  $\mathbb{R}^n$ .
- Die Monombasis  $(1, x, x^2, \dots, x^n)$  ist eine Basis des Vektorraums der Polynome  $n$ -ten Grades.
- $(1, i)$  ist eine Basis von  $\mathbb{C}$  als  $\mathbb{R}$ -Vektorraum.
- $\mathbb{R}$  als  $\mathbb{Q}$ -Vektorraum hat keine endliche Basis.

- Sei  $(v_1, \dots, v_n)$  eine Basis eines Vektorraums  $V$ . Dann ist die **Dimension** des Vektorraums  $V$  definiert durch die Anzahl der Basiselemente, also  $n$ .
- Jeder Vektorraum besitzt eine Basis.
- Seien  $W, Z$  Unterräume von  $V$ . Dann ist  $W + Z := \text{span}(W \cup Z)$  die **Summe** von  $W$  und  $Z$ . Es gilt:

$$\dim(W + Z) = \dim(W) + \dim(Z) - \dim(W \cap Z)$$

- Bestimmen einer Basis von  $\text{span}(s_1, s_2, s_3)$

```
>> s1 = vector([1,0,0])
>> s2 = vector([0,1,1])
>> s3 = vector([1,1,1])
>> V = VectorSpace(QQ,3); S = V.subspace([s1,s2,
      s3]); S
```

```
Vector space of degree 3 and dimension 2 over
      Rational Field
```

```
Basis matrix:
```

```
[1 0 0]
```

```
[0 1 1]
```

- Bestimmen des Schnitts von  $\text{span}(s_1)$  und  $\text{span}(s_2, s_3)$

```
>> (V.subspace([s1])).intersection(V.subspace([  
    s2,s3]))
```

- Testen der linearen Unabhängigkeit

```
>> matrix([s1,s2,s3]).rank() >= len(s1)
```

```
False
```

```
test.apply_map(sqrt)
```

# Normen auf Vektorräumen

Sei  $V$  ein Vektorraum über  $K = \mathbb{R}$  oder  $K = \mathbb{C}$ .

Eine **Norm** auf  $V$  ist eine Abbildung

$$\| \cdot \| : V \rightarrow \mathbb{R}, v \mapsto \|v\|,$$

so dass für alle  $\alpha \in K$ ,  $u, v \in V$  gilt

$$\|v\| \geq 0$$

$$\|v\| = 0 \text{ impliziert } v = 0$$

$$\|\alpha v\| = |\alpha| \|v\|$$

$$\|u + v\| \leq \|u\| + \|v\| \text{ (Dreiecksungleichung).}$$

$(V, \| \cdot \|)$  heißt **normierter Raum**.

# Skalarprodukt

Eine skalarwertige binäre Abbildung

$$(\cdot, \cdot) : V \times V \rightarrow K$$

auf einem Vektorraum  $V$  über  $K = \mathbb{R}$  oder  $K = \mathbb{C}$  heißt **Skalarprodukt**, wenn für alle  $x, y, z \in V$ ,  $\alpha, \beta \in K$  gilt

$$\langle x, x \rangle \geq 0$$

$$\langle x, x \rangle = 0 \text{ impliziert } x = 0.$$

$$\langle x, y \rangle = \overline{\langle y, x \rangle}$$

$$\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$$



- Ein VR  $V$  mit Skalarprodukt heißt **Prä-Hilbert-Raum**. Ist  $K = \mathbb{R}$  so heißt der Raum auch **euklidisch**.
- Durch  $\|v\| := \sqrt{(v, v)}$ ,  $v \in V$  läßt sich eine Norm definieren. Es gilt die **Cauchy-Schwarzsche Ungleichung**

$$|(u, v)| \leq \|u\| \|v\|.$$

- Im euklidischen Raum ist der Winkel  $\alpha$  zwischen zwei Vektoren  $u, v \in V \setminus \{0\}$  definiert durch

$$\cos(\alpha) = \frac{(u, v)}{\|u\| \|v\|}.$$

- Zwei Vektoren  $u, v \in V$  heißen **orthogonal**, wenn  $(u, v) = 0$  gilt.
- Eine Basis aus paarweise orthogonalen Vektoren heißt **Orthogonalbasis**.
- Eine Orthogonalbasis, bei der alle Vektoren die Norm 1 haben, nennt man **Orthonormalbasis**.
- Jeder endlichdimensionale Prä-Hilbert-Raum hat eine Orthonormalbasis.
- Ist  $U$  ein Unterraum von  $V$ , so ist

$$U^\perp := \{v \in V \mid (v, u) = 0 \text{ für alle } u \in U\}$$

der **Orthogonalraum** zu  $U$ . Er ist ein Untervektorraum.

- Es gilt:  $\dim U + \dim U^\perp = \dim V$ , insb.  $U \cap U^\perp = 0$ .

- Die  $p$ -Norm  $\|v\| := (\sum_{i=1}^n |v_i|^p)^{1/p}$ ,  $p \in [1, \infty)$  auf dem  $K^n$  mit  $K = \mathbb{R}, \mathbb{C}$  wird berechnet durch:

```
>> x = vector([1,2,3,4,5]); p=2  
>> x.norm(p)
```

- Orthogonalisieren von Vektoren:

```
>> a1 = vector([1,2,3])  
>> a2 = vector([0,4,1])  
>> a3 = vector([1,1,1])  
>> linalg::orthog([a1,a2,a3]) ??
```

- Berechnen des Skalarprodukts

```
>> a2.dot_product(a3), a2*a3
```

```
(5, 5)
```

- Berechnen des Winkels zwischen zwei Vektoren

```
>> float(acos(a2*a3/(abs(a2)*abs(a3))))
```

```
0.79520271328967818
```

- Berechnen der Determinante

```
>> A = matrix([[1,2,3],[4,5,6],[7,8,0]])
```

```
>> A.det()
```

```
27
```

## 1 Vektoren

- Matrizen
- Vektorräume

## 2 Etwas Programmieren

# Ein erstes Programm

```
def MyMax(a,b):  
    #Maximum von a und b  
    if a<b:  
        return (b)  
    else:  
        return (a)
```

- Das erste Beispiel berechnet das Maximum zweier Zahlen  $a$  und  $b$ .
- Aufruf in Sage ist `MyMax(a,b)`.
- Die Funktion gibt dann entweder den Wert  $a$  oder den Wert  $b$  zurück.

- Eine Funktion beginnt mit `def` und dem Namen der Prozedur (hier: `MyMax`) gefolgt von den möglichen Übergabeparameter (`a,b,c...`)
- Jede Zeile, die einen folgenden Block einleitet, muss mit einem Doppelpunkt `:` abgeschlossen werden.
- Jede Zeile in diesem Block muss eine grössere Einrückung besitzen (typischerweise ein *Tab*)

- Mittels `return(a)` wird die Funktion abgebrochen und der Wert `a` zurückgegeben.
- Wird innerhalb der Prozedur kein Befehl ausgeführt, so wird eine leere Variable des Typs `NoneType` zurückgegeben.



- Kommentarzeilen: Diese beginnen mit `#`. Sie werden vom System ignoriert.
- Die durch `if <bedingung>`: eingeleitete Zeile ist eine sogenannte Verzweigung. Ist die Bedingung `<bedingung>` wahr, so wird der Block hinter `:` ausgeführt. Ist `<bedingung>` falsch, so wird die Alternative `else:` ausgeführt.
- *Achtung:* in Sage (und Python) wird der Block durch die Einrückung definiert.

Bei umfangreicheren Funktionen oder richtigen Programmen ist es evtl. sinnvoller dies mit einem Editor zu erstellen.

- Erstellen einer `<name>.sage`-Datei in `~/ .sage`. Diese mit Sage-Code füllen
- In sage diese mit `attach <name>.sage` an das Notebook anhängen  
⇒ wird automatisch geladen und aktuell gehalten.
- als Editor empfehle ich zum Einstieg `geany`. Prinzipiell sind alle Editoren geeignet die Python unterstützen.