Computer und Software 1

Christof Köhler

5. Maple – Lineare Algebra, Vektorfelder

Folien: Balint Aradi

Grundsätzliche Bemerkungen im voraus

Die meisten Befehle zur linearen Algebra befinden sich in Paketen, die entsprechend geladen werden müssen.

```
> with(LinearAlgebra):
```

Zwei verschiedene Pakete für lineare Algebra:

linalg	Veraltet, sollte nicht mehr verwendet werden
LinearAlgebra	Neue Version, enthält mehr und bequemere Routinen, ist für die Manipulation von größeren Matrizen wesentlich schneller

Sie sollten immer das Paket **LinearAlgebra** verwenden!

Manipulation von sehr großen (numerischen) Matrizen ist nicht unbedingt die Stärke von Maple (obwohl im Paket **LinearAlgebra** stark verbessert).
Man sollte in dafür eher andere Programme (z.B. Octave, Matlab, etc.) einsetzen.

Eingabe von Vektoren

Maple arbeitet grundsätzlich mit Spaltenvektoren (Dimension: beliebig)

Eingabe über die Funktion **Vektor()** oder **< , , >** Notation

```
> a := Vector([1, 2, 3]);
```

$$a := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
> b := < 1, 2, 3 >;
```

$$b := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Zeilenvektoren können über die Funktion **Vector[row]()** oder mit der **< | | >** Notation eingegeben werden:

```
> c := Vector[row]([1, 2, 3 ]);
```

$$c := [1, 2, 3]$$

```
> d := < 1 | 2 | 3 >;
```

$$d := [1, 2, 3]$$

Eingabe von Matrizen

Matrizen werden über die Funktion **Matrix()** eingegeben.
Die Eingabe erfolgt **zeilenweise** (als eine Liste der Zeilen):

```
> m1 := Matrix([[1, 2], [ 3, 4]]);
```

$$m1 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Alternativ kann die Matrix als ein (Spalten-) Vektor von Zeilenvektoren eingegeben werden:

```
> m2 := << 1 | 2 >, < 3 | 4 >>;
```

$$m2 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Eingabe als (Zeilen-) Vektor von Spaltenvektoren auch möglich. In diesem Fall muss die Matrix entsprechend **spaltenweise** eingegeben werden.

```
> m3 := << 1, 3 > | < 2, 4 >>;
```

$$m3 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Spezielle Vektoren/Matrizen erstellen

Vektoren und Matrizen können auch ohne Werte erstellt werden (Auffüllen mit 0):

```
> v := Vector(2);
```

$$v := \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

```
> m := Matrix(2, 3);
```

$$m := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Einheitsmatrix:

```
> I2 := IdentityMatrix(2);
```

$$I2 := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Diagonalmatrix aus der Liste der Diagonalelemente

```
> D2 := DiagonalMatrix([1, 2]);
```

$$D2 := \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Einheitsvektor

```
> UnitVector(1, 2);
```

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

1. Einheitsvektor im 2-dimensionalen Raum

Zugriff auf Elemente von Vektoren/Matrizen

Elemente von Vektoren werden wie Listenelemente manipuliert:

```
> v := < 1 | 2 >;
```

$$v := [1, 2]$$

```
> v[1];
```

$$1$$

```
> v[2] := 9;
```

$$v_2 := 9$$

```
> v;
```

$$[1, 9]$$

Bei Matrizen müssen alle Indizes (Zeile, Spalte) angegeben werden:

```
> m := << 1 | 2 >, < 3 | 4 >>;
```

$$m := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> m[2,1];
```

$$3$$

```
> m[2,1] := -12;
```

$$m_{2,1} := -12$$

```
> m;
```

$$\begin{bmatrix} 1 & 2 \\ -12 & 4 \end{bmatrix}$$

Einzelne Spalten bzw. Zeilen können mit **Row()** und **Column()** **ausgelesen** werden:

```
> Row(m, 1);
```

$$[1, 2]$$ ← 1. Zeile von m (Ergebnis: Zeilenvektor)

```
> Column(m,1);
```

$$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$$ ← 1. Spalte von m (Ergebnis: Spaltenvektor)

Zugriff auf Elemente von Vektoren/Matrizen (#2)

Teilbereiche einer Matrix / eine Vektors können mit entsprechenden Bereichselektoren ausgegeben und manipuliert werden:

```
> M := << 1 | 2 >, < 3 | 4 >>;
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> M[1..2, 1];
```

$$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

```
> M[1..2, 1] := < -1, -1 >;
```

$$M_{1\,..\,2,\,1} := \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

```
> M;
```

$$\begin{bmatrix} -1 & 2 \\ -1 & 4 \end{bmatrix}$$

```
> M[1, 1..2] := < 1 | 1 >;
```

$$M_{1,\,1\,..\,2} := [1, 1]$$

```
> M;
```

$$\begin{bmatrix} 1 & 1 \\ -1 & 4 \end{bmatrix}$$

```
> M[1..2, 1..2] :=
  << 9 | 8 >, < 7 | 6 >>;
```

$$M_{1\,..\,2,\,1\,..\,2} := \begin{bmatrix} 9 & 8 \\ 7 & 6 \end{bmatrix}$$

```
> M;
```

$$\begin{bmatrix} 9 & 8 \\ 7 & 6 \end{bmatrix}$$

Operationen mit Vektoren

Multiplikation mit einem Skalar:

```
> v1 := < 1, 2 >;
```

$$v1 := \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> 5 * v1;
```

$$\begin{bmatrix} 5 \\ 10 \end{bmatrix}$$

Addition/Subtraktion von Vektoren:

```
> < 1, 2 > - < 3, 4 >;
```

$$\begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

```
> < 1, 2 > + < 3, 4 >;
```

$$\begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

Skalarprodukt (Operator **.**)

```
> v2 := < 3, 4 >;
```

$$v2 := \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

```
> v1 . v2;
```

$$11$$

Vektorprodukt (Operator **&x**, nur für 3D-Vektoren!):

```
> u1 := < 1, 2, 3 >;
  u2 := < 4, 5, 6 >;
```

$$u1 := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad u2 := \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

```
> u1 &x u2;
```

$$\begin{bmatrix} -3 \\ 6 \\ -3 \end{bmatrix}$$

Operationen mit Matrizen

Multiplikation mit einem Skalar:

```
> mm := Matrix([[ 1, 2 ], [ 3, 4 ]]);
```

$$mm := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> 3 * mm;
```

$$\begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

Addition/Subtraktion von Matrizen:

```
> nn := << 3 | -1 >, < 2 | 9 >>;
```

$$nn := \begin{bmatrix} 3 & -1 \\ 2 & 9 \end{bmatrix}$$

```
> mm + nn;
```

$$\begin{bmatrix} 4 & 1 \\ 5 & 13 \end{bmatrix}$$

Matrixmultiplikation (Matrix-Matrix, Matrix-Vektor)

```
> mm . nn;
```

$$\begin{bmatrix} 7 & 17 \\ 17 & 33 \end{bmatrix}$$

```
> mm, v1;
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> mm . v1;
```

$$\begin{bmatrix} 5 \\ 11 \end{bmatrix}$$

Vektor-/Matrixeigenschaften

Länge (Zweiernorm) eines Vektors: Funktion **Norm()**

```
> v := < a, b, c >;
```

$$v := \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

```
> Norm(v, 2);
```

$$\sqrt{|a|^2 + |b|^2 + |c|^2}$$

Vektor

Welche Norm?
(2: Euklidisch)

Winkel zwischen zwei Vektoren (in Radiant): Funktion **VectorAngle()**

```
> v1 := < 1, 0 >;
```

$$v1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

```
> v2 := < 0, 1 >;
```

$$v2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

```
> VectorAngle(v1, v2);
```

$$\frac{1}{2}\pi$$

Determinante einer Matrix: Funktion **Determinant()**

```
> mm := << a | b >, < c | d >>;
```

$$mm := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> Determinant(mm);
```

$$a\,d - b\,c$$

Matrixtransformationen

Transponieren: Funktion **Transpose()** oder Potenz **%T**

```
> mm;
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> Transpose(mm);
```

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

```
> mm^%T;
```

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

Adjungieren :Funktion **HermitianTranspose()** oder Potenz **%H**:

```
> HermitianTranspose(mm);
```

$$\begin{bmatrix} \overline{a} & \overline{c} \\ \overline{b} & \overline{d} \end{bmatrix}$$

```
> mm^%H;
```

$$\begin{bmatrix} \overline{a} & \overline{c} \\ \overline{b} & \overline{d} \end{bmatrix}$$

Invertieren (A^{-1} A = 1): Funktion **MatrixInverse():**

```
> m2 := << 1 | 2 >, < 3 | 4 >>;
```

$$m2 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> MatrixInverse(m2);
```

$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & \frac{-1}{2} \end{bmatrix}$$

```
> m2 . MatrixInverse(m2);
```

$\longrightarrow$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Eigenwerte/Eigenvektoren

Eigenwerte/Eigenvektoren einer Matrix M: Lösungen der Gleichung $M\,v = \mu\,v$

μ – Eigenwert (Skalar), v – dazu gehörender Eigenvektor (Vektor)

Funktion **Eigenvectors()** liefert Eigenwerte und Eigenvektoren als Sequenz

```
> mm := << 1 | 2 >, < 3 | 4 >>;
```

$$mm := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> Eigenvectors(mm);
```

$$\begin{bmatrix} \frac{5}{2} + \frac{1}{2}\sqrt{33} \\ \frac{5}{2} - \frac{1}{2}\sqrt{33} \end{bmatrix}, \begin{bmatrix} \frac{2}{\frac{3}{2} + \frac{1}{2}\sqrt{33}} & \frac{2}{\frac{3}{2} - \frac{1}{2}\sqrt{33}} \\ 1 & 1 \end{bmatrix}$$

Eigenwert 1

Eigenwert 2

Eigenvektor 1

Eigenvektor 2

Sequenz: Vektor mit Eigenwerten,
Matrix mit Eigenvektoren als Spaltenvektoren

Eigenwerte/Eigenvektoren

Wird die Matrix mit Fließkommazahlen eingegeben, ist das Ergebnis auch numerisch:

```
> mm := << 1.0 | 2.0 >, < 3.0 | 4.0 >>;
```

$$mm := \begin{bmatrix} 1.0 & 2.0 \\ 3.0 & 4.0 \end{bmatrix}$$

```
> eigvals, eigvecs := Eigenvectors(mm):
> eigvals;
```

$$\begin{bmatrix} -0.37228132326901431 + 0.\,I \\ 5.3722813232690143 + 0.\,I \end{bmatrix}$$

```
> eigvecs;
```

$$[[-0.82456840132393840 + 0.\,I,\ -0.41597355791928419 + 0.\,I],\ [0.56576746496892328 + 0.\,I,\ -0.90937670913212409 + 0.\,I]]$$

Zuweisung des Ergebnisses (Sequenz mit zwei Elementen) zu einzelnen Variablen

Bei größeren Matrizen sollte immer die numerische Berechnung verwendet werden (viel schneller)

Berechnung wird mit Maschinengenauigkeit ausgeführt (unabhängig von **Digits**)

Auswerten von Vektoren/Matrizen

Matrizen/Vektoren können mit **evalf()** numerisch ausgewertet bzw. zu Fließkommamatrizen/vektoren konvertiert werden:

```
> mm := << 1 | 2 >, < 3 | 4 >>;
```

$$mm := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> m2 := evalf(mm);
```

$$\begin{bmatrix} 1. & 2. \\ 3. & 4. \end{bmatrix}$$

Der **subs()** Befehl wird bei Matrizen/Vektoren komponentenweise angewendet:

```
> M := << x^2 | 2*x - 1 >, < 1/x | ln(x) >>;
```

$$M := \begin{bmatrix} x^2 & 2x - 1 \\ \frac{1}{x} & \ln(x) \end{bmatrix}$$

```
> subs({ x=2 }, M);
```

$$\begin{bmatrix} 4 & 3 \\ \frac{1}{2} & \ln(2) \end{bmatrix}$$

Komponentenweise Anwendung von Funktionen

Funktionen, die ein Skalar als Argument erwarten, können mit dem **map()** Befehl auf die Komponenten eines entsprechenden Objektes (Liste, Menge, Vektor, Matrix) angewendet werden.

```
> a := [ 1, 2, 3 ];
```

$$a := [1, 2, 3]$$

```
> map(sin, a);
```

$$[\sin(1), \sin(2), \sin(3)]$$

```
> a := { 1, 2, 3 };
```

$$a := \{1, 2, 3\}$$

```
> map(sin, a);
```

$$\{\sin(2), \sin(3), \sin(1)\}$$

```
> v := << 1 | 2 >, < 3 | 4 >>;
```

$$v := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> map(sin, v);
```

$$\begin{bmatrix} \sin(1) & \sin(2) \\ \sin(3) & \sin(4) \end{bmatrix}$$

Erwartet die Funktion mehrere Argumente, können für die weiteren Argumente entsprechende **Skalar**werte der map-Funktion übergeben werden:

```
> mult := (x, y) -> x * y;
```

$$mult := (x, y) \to x\,y$$

```
> map(mult, v, 2);
```

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

Komponentenweise Anwendung von Funktionen (#2)

```
> int_cube := f -> rho *
  Int(Int(Int(f,
  x=0..a), y=0..a), z=0..a);
```

$$int\_cube := f \to \rho \int_0^a \int_0^a \int_0^a f\,dx\,dy\,dz$$

```
> r := < x, y, z >;
```

$$r := \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

```
> myint := map(int_cube, r);
```

Komponentenweise Integrieren

$$myint := \begin{bmatrix} \rho \int_0^a \int_0^a \int_0^a x\,dx\,dy\,dz \\ \rho \int_0^a \int_0^a \int_0^a y\,dx\,dy\,dz \\ \rho \int_0^a \int_0^a \int_0^a z\,dx\,dy\,dz \end{bmatrix}$$

Berechnung des Massenmittelpunktes eines Würfels

$$r_x = \frac{\rho}{M} \int_0^a \int_0^a \int_0^a x\,dx\,dy\,dz \qquad \rho = \frac{M}{a^3}$$

$$r_y = \frac{\rho}{M} \int_0^a \int_0^a \int_0^a y\,dx\,dy\,dz$$

$$r_z = \frac{\rho}{M} \int_0^a \int_0^a \int_0^a z\,dx\,dy\,dz$$

```
> subs(rho=M/a^3, value(myint) / M);
```

Einsetzen für die Dichte

Auswertung der Integrale

$$\begin{bmatrix} \frac{1}{2}a \\ \frac{1}{2}a \\ \frac{1}{2}a \end{bmatrix}$$

Gleichungssysteme

Lineare Gleichungssysteme (A x = b) können in Matrixform mit **LinearSolve()** gelöst werden:

```
> A := << 1 | 2 >, < 3 | 4 >>;
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> b := < 4, 5 >;
```

$$b := \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$x = A^{-1} b$

```
> x := LinearSolve(A, b);
```

$$x := \begin{bmatrix} -3 \\ \frac{7}{2} \end{bmatrix}$$

Alternativ kann nur eine Matrix der Form *n* x (*n* – 1) **LinearSolve()** übergeben werden:

```
> Ab := << 1 | 2 | 4 >, < 3 | 4 | 5 >>;
```

$$Ab := \begin{bmatrix} 1 & 2 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

```
> x := LinearSolve(Ab);
```

$$x := \begin{bmatrix} -3 \\ \frac{7}{2} \end{bmatrix}$$

Koordinatensysteme für Vektoren

Um Vektoren/Vektorfelder in verschiedenen Koordinatensystemen zu manipulieren muss (außer **LinearAlgebra**) das Paket **VectorCalculus** geladen werden.

```
> with(VectorCalculus):
```

Nach dem Laden von **VectorCalculus** werden alle Vektoren als Linearkombination der entsprechenden Basisvektoren angezeigt:

```
> < x, y, z >;
```

$$x\,e_x + y\,e_y + z\,e_z$$

Das Koordinatensystem kann mit **SetCoordinates()** gesetzt werden und mit **GetCoordinates()** abgefragt werden.

```
> SetCoordinates(cylindrical[r, z, phi]);
```

$$cylindrical_{r,\,z,\,\phi}$$

Zylinderkoordinaten

```
> < r, 1/z * r, 0 >;
```

$$r\,e_r + \frac{r}{z}\,e_z$$

Vektoreingabe als Vektorfeld in Zylinderkoordinaten interpretiert

```
> GetCoordinates();
```

$$cylindrical_{r,\,z,\,\phi}$$

Koordinatensysteme für Vektoren (#2)

Operationen (z.B. Skalarprodukt) werden immer im aktuellen Koordinatensystem interpretiert.

```
> GetCoordinates();
```

$$cartesian$$

```
> < a, b, c > . < d, e, f >;
```

$$a\,d + b\,e + c\,f$$

```
> SetCoordinates(spherical);
```

$$spherical$$

```
> < a, b, c > . < d, e, f >;
```

$$a \sin(b) \cos(c)\, d \sin(e) \cos(f) + a \sin(b) \sin(c)\, d \sin(e) \sin(f) + a \cos(b)\, d \cos(e)$$

Mit **MapToBasis()** kann zwischen Koordinatensystemen konvertiert werden.

```
> GetCoordinates();
```

$$cartesian$$

```
> MapToBasis(< x, y, z >, spherical);
```

$$\sqrt{x^2 + y^2 + z^2}\, e_r + \arctan(\sqrt{y^2 + x^2}, z)\, e_\phi + \arctan(y, x)\, e_\theta$$

Basis des Vektors kann mit **attributes()** abgefragt werden:

```
> v := < 1, 1, 1 >;
```

$$v := e_x + e_y + e_z$$

```
> attributes(v);
```

$$coords = cartesian$$

Vektorfelder

Vektorfelder sind $R^3 \rightarrow R^3$ Funktionen (z.B. elektrisches Feld)

Vektorfelder können mit dem Befehl **VectorField()** angegeben werden.

Als zweites Argument muss das Koordinatensystem (kartesisch, sphärisch, zylindrisch) und die Namen der einzelnen Koordinaten spezifiziert werden.

```
> field := VectorField( < x, y, z >, cartesian[x, y, z]);
```

$$field := x\,\underline{e}_x + y\,\underline{e}_y + z\,\underline{e}_z$$

Koordinateneinheitsvektoren

Kartesische Koordinaten (x, y, z)

```
> field2 := VectorField(< r, 0, 0 >, spherical[r, theta, phi]);
```

$$field2 := r\,\underline{e}_r$$

```
> field3 := VectorField(<r, 1, 1 >, cylindrical[r, h, phi]);
```

$$field3 := r\,\underline{e}_r + \underline{e}_h + \underline{e}_\phi$$

Koordinatensystem eines Vektorfeldes kann via **attributes()** abgefragt werden:

```
> attributes(field);
```

$$vectorfield,\ coords = cartesian_{x,\,y,\,z}$$

Operationen mit Vektorfelder

Vektorfeld als Gradient einer Skalarfunktion: Funktion **Gradient()**

```
> field := Gradient(x^2 - 2*y + ln(z), [ x, y, z ]);
```

$$field := 2\,x\,\bar{e}_x - 2\,\bar{e}_y + \frac{1}{z}\,\bar{e}_z$$

Divergenz eines Vektorfeldes: **Divergence()**

```
> Divergence(field);
```

$$2 - \frac{1}{z^2}$$

Rotation eines Vektorfeldes: **Curl()**

```
> Curl(field);
```

$$0\,\bar{e}_x$$

Laplace-Operator: Funktion **Laplacian()**

```
> Laplacian(x^2 - 2*y + ln(z), cartesian[x, y, z]);
```

$$2 - \frac{1}{z^2}$$

Nabla-Operator

Differentialoperationen an Vektorfelder können auch mit dem Nablaoperator beschrieben werden.

```
> f := x^2 - 2*y + ln(z);
```

$$f := x^2 - 2\,y + \ln(z)$$

```
> field := Nabla(f, [x, y, z]);
```

$$field := 2\,x\,\bar{e}_x - 2\,\bar{e}_y + \frac{1}{z}\,\bar{e}_z$$

← Gradient

```
> Nabla . field;
```

$$2 - \frac{1}{z^2}$$

← Divergenz

```
> Nabla &x field;
```

$$0\,\bar{e}_x$$

← Rotation

```
> (Nabla . Nabla)(f, [ x, y, z ]);
```

$$2 - \frac{1}{z^2}$$

← Laplace-Operator

Koordinatensystem für Vektorenfelder

Operationen mit Vektorfelder werden immer im **aktuellen** Koordinatensystem ausgeführt.

```
> GetCoordinates();
```

cartesian

```
> Gradient(r + sin(theta), [ r, theta, phi ]);
```

$$\bar{e}_r + \cos(\theta)\,\bar{e}_\theta$$

← Gradient in kartesischen Koordinaten !

```
> SetCoordinates(spherical);
```

spherical

```
> Gradient(r + sin(theta), [ r, theta, phi ]);
```

$$\bar{e}_r + \frac{\cos(\theta)}{r}\,\bar{e}_\theta$$

← Korrekter Gradient in Kugelkoordinaten