Vektoren und Matrizen, Lineare Algebra

■ Vektoren

▪ Vektoren eingeben. Zusammengesetzte Bezeichner

```
ClearAll["Global`*"]
```

```
v1 = {a, b, c}; v2 ^= {u, v, w};
```

$\overrightarrow{v1}$ und $\overrightarrow{v2}$ sind zusammengesetzte Bezeichner, bei deren Verwendung einige Besonderheiten zu beachten sind.

Zunächst entsprechen diesen Bezeichnern keine eigenständigen Einträge in der Symboltabelle, weil sie selbst zusammengesetzte Ausdrücke sind, wie die **FullForm** an den Tag bringt.

```
v1 // Hold // FullForm
```

```
Hold[OverVector[v1]]
```

Die Werte der einzelnen Variablen werden standardmäßig als **DownValue** unter dem Funktionsbezeichner, hier also **OverVector**, gespeichert. Sie können aber auch als **UpValue** wie hier auf dem Bezeichner v2 selbst gespeichert werden.

```
?OverVector
DownValues[OverVector]
UpValues[v2]
```

```
OverVector[expr] displays with a right vector over expr.
```

$\{\text{HoldPattern}[\overrightarrow{v1}] :\mapsto \{a, b, c\}\}$

$\{\text{HoldPattern}[\overrightarrow{v2}] :\mapsto \{u, v, w\}\}$

```
?v2
```

```
Global`v2
```

$\overrightarrow{v2}$ ^= {u, v, w}

Je nachdem, wie Sie die Werte abspeichern, müssen Sie verschiedene Kommandos zum Löschen verwenden.

```
Clear[v1, v2]
```

$\{\vec{v1}, \vec{v2}\}$

$\{\{a, b, c\}, \vec{v2}\}$

```
Clear[OverVector]
```

$\{\vec{v1}, \vec{v2}\}$

$\{v1, v2\}$

MatrixForm ist ein Formatierungskommando, das sowohl auf Vektoren als auch auf Matrizen angewendet werden kann.

Beachten Sie, dass einspaltige Matrizen und Vektoren intern unterschiedlich dargestellt sind, von MatrixForm aber in gleicher Form ausgegeben werden.

```
v1 = {a, b, c}; v2 = {u, v, w};
v1 + v2 // MatrixForm
```

$$\begin{pmatrix} a + u \\ b + v \\ c + w \end{pmatrix}$$

$v1^{v2}$ sieht mathematisch recht seltsam (und nicht sehr sinnvoll) aus; die Auswertung gelingt, weil *Mathematica* Vektoren eigentlich nicht kennt, sondern als gewöhnliche Listen betrachtet

```
v1^v2 // MatrixForm
```

$$\begin{pmatrix} a^u \\ b^v \\ c^w \end{pmatrix}$$

Es gibt drei verschiedene Multiplikationsvarianten: Multiplikation eines Vektors mit einem Skalar, das skalare Produkt zweier Vektoren (Eingabe: ein einfacher Punkt .) und das Kreuzprodukt (Eingabe: ESCcrossESC oder Cross[$\vec{v1}$, $\vec{v2}$], nur für Vektoren im R^3)

```
3 v1 // MatrixForm
```

$$\begin{pmatrix} 3\,a \\ 3\,b \\ 3\,c \end{pmatrix}$$

```
v1 . v2
```

$a\,u + b\,v + c\,w$

```
v1 × v2 // MatrixForm
```

$$\begin{pmatrix} -c\,v + b\,w \\ c\,u - a\,w \\ -b\,u + a\,v \end{pmatrix}$$

Das Kreuzprodukt ist allgemein für (n−1)−Tupel von Vektoren im R^n definiert.

```
Cross[{1, 2, 3, 4}, {2, 3, 1, 4}, {3, 1, 2, 4}]
```

```
{12, 12, 12, -18}
```

■ Beispiel: Schnitt Gerade−Ebene

Schnitt Gerade mit Ebene

P, Q	Punkte im 3D Raum
$\vec{v}$	Vektoren
g	Gerade
ϵ	Ebene

```
Clear[OverVector]
P = {12, 3, 5}; Q = {9, 8, 4};
```

Verbundzuweisung und Darstellung als Liste von Vektoren in zweidimensionaler Notation.

```
MatrixForm /@ ({u⃗, v⃗, w⃗} = {{1, 0, -1}, {3, 2, 1}, {0, 1, 4}})
```

$$\left\{\begin{pmatrix}1\\0\\-1\end{pmatrix}, \begin{pmatrix}3\\2\\1\end{pmatrix}, \begin{pmatrix}0\\1\\4\end{pmatrix}\right\}$$

Dieselbe Liste von Listen als Matrix in zweidimensionaler Notation. Obige Vektoren sind die **Zeilen** dieser Matrix.

```
{u⃗, v⃗, w⃗} // MatrixForm
```

$$\begin{pmatrix}1 & 0 & -1\\3 & 2 & 1\\0 & 1 & 4\end{pmatrix}$$

```
(g = P + a u⃗) // MatrixForm
```

$$\begin{pmatrix}12 + a\\3\\5 - a\end{pmatrix}$$

```
(ε = Q + b v⃗ + c w⃗) // MatrixForm
```

$$\begin{pmatrix}9 + 3\,b\\8 + 2\,b + c\\4 + b + 4\,c\end{pmatrix}$$

```
lsg = Solve[g == ε]
```

```
{{a → -21, b → -6, c → 7}}
```

```
g /. lsg[[1]]
```

```
{-9, 3, 26}
```

Beachten Sie, dass Wertzuweisungen an zusammengesetzte Symbole nur über den Bezeichner gelöscht werden können, dem sie zugeordnet sind, hier

also dem Funktionsbezeichner **OverVector**.

```
?? OverVector
```

OverVector[*expr*] displays with a right vector over *expr*.

```
u⃗ = {1, 0, -1}
```

```
v⃗ = {3, 2, 1}
```

```
w⃗ = {0, 1, 4}
```

Löschen solches Symbole ist jetzt schwieriger, weil sie im Kontext **System'** gespeichert sind.

```
ClearAll["Global`*"]
v⃗
```

```
{3, 2, 1}
```

```
Clear[OverVector]
v⃗
```

```
v⃗
```

■ Unterscheiden Sie zwischen Vektoren und einspaltigen Matrizen!

Vorsicht: Vektoren und einspaltige Matrizen haben unterschiedliche interne Darstellungen, werden aber von MatrixForm auf dieselbe Weise dargestellt.

Über Palette oder 3D−Eingabe können nur einspaltige Matrizen eingegeben werden.

$\vec{a}$ und $\vec{b}$ sind einspaltige Matrizen, $\vec{c}$ dagegen ein Vektor. Das merkt man bei den ersten beiden Kommandos noch nicht, denn sie werden komponentenweise ausgeführt.

```
a⃗ = {{a1}, {a2}, {a3}}; b⃗ = {{b1}, {b2}, {b3}}; c⃗ = {c1, c2, c3};
```

```
{a⃗, b⃗, c⃗}
```

```
{{{a1}, {a2}, {a3}}, {{b1}, {b2}, {b3}}, {c1, c2, c3}}
```

```
MatrixForm /@ {a⃗, b⃗, c⃗}
```

$$\left\{\begin{pmatrix} a1 \\ a2 \\ a3 \end{pmatrix}, \begin{pmatrix} b1 \\ b2 \\ b3 \end{pmatrix}, \begin{pmatrix} c1 \\ c2 \\ c3 \end{pmatrix}\right\}$$

```
VectorQ[a⃗]
```

```
False
```

Einspaltige Matrizen können wie Vektoren addiert und skalar vervielfacht werden. Das Ergebnis ist wieder eine einspaltige Matrix.

```
a⃗ + b⃗ // MatrixForm
```

$$\begin{pmatrix} a1 + b1 \\ a2 + b2 \\ a3 + b3 \end{pmatrix}$$

Selbst die Summe einer einspaltigen Matrix und eines Vektors kann gebildet werden. Das Ergebnis ist eine einspaltige Matrix.

```
a⃗ + c⃗
```

```
{{a1 + c1}, {a2 + c2}, {a3 + c3}}
```

```
% // MatrixForm
```

$$\begin{pmatrix} a1 + c1 \\ a2 + c2 \\ a3 + c3 \end{pmatrix}$$

Als einspaltige Matrizen sind $\vec{a}$ und $\vec{b}$ nicht verkettet − deshalb endet dieses Dot−Kommando mit einer Fehlermeldung. Wären es Vektoren, würde es als Skalarprodukt interpretiert und ausgeführt.

```
a⃗ . b⃗
```

```
Dot::dotsh : Tensors {{a1}, {a2}, {a3}}
   and {{b1}, {b2}, {b3}} have incompatible shapes.
```

```
{{a1}, {a2}, {a3}}.{{b1}, {b2}, {b3}}
```

Dasselbe gilt für das Vektorprodukt, das ebenfalls nur für Vektoren, nicht aber für einspaltige Matrizen definiert ist.

```
a⃗ × b⃗
```

```
Cross::nonn1 :
 The arguments are expected to be vectors of equal length, and the
   number of arguments is expected to be 1 less than their length.
```

```
{{a1}, {a2}, {a3}} × {{b1}, {b2}, {b3}}
```

Verkettet sind hingegen $\vec{a}$ und die zu $\vec{b}$ transponierte Matrix, und zwar in beiden Reihenfolgen.
Im ersten Fall ergibt sich eine 3×3−Matrix, deren Einträge die paarweisen Produkte sind; im zweiten Fall eine 1×1−Matrix mit dem Skalarprodukt als Eintrag.

```
a⃗ .Transpose[ b⃗] // MatrixForm
```

$$\begin{pmatrix} a1\,b1 & a1\,b2 & a1\,b3 \\ a2\,b1 & a2\,b2 & a2\,b3 \\ a3\,b1 & a3\,b2 & a3\,b3 \end{pmatrix}$$

```
Transpose[b⃗].a⃗ // MatrixForm
```

```
( a1 b1 + a2 b2 + a3 b3 )
```

Auf die Kombination von Matrix und Vektor wirkt . nicht als Skalarprodukt, sondern als Produkt von Matrix und Vektor. Dafür müssen beide verkettet sein. Als Ergebnis ergibt sich ein 1−Vektor, dessen Eintrag der Wert des Skalarprodukts ist.

```
a⃗ . c⃗
```

```
Dot::dotsh :
 Tensors {{a1}, {a2}, {a3}} and {c1, c2, c3} have incompatible shapes.
```

```
{{a1}, {a2}, {a3}}.{c1, c2, c3}
```

```
c⃗ . b⃗
```

```
{b1 c1 + b2 c2 + b3 c3}
```

Matrizen erzeugen

Matrizen definieren

```
mat = {{1, 2, 3}, {4, 5, 6}};
```

```
mat // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

```
mat = ⎛1 2 3⎞;
      ⎝4 5 6⎠
```

```
mat1 = IdentityMatrix[3];
mat1 // MatrixForm
```

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

```
mat2 = DiagonalMatrix[{-1, 3, -5}];
mat2 // MatrixForm
```

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -5 \end{pmatrix}$$

```
mat3 = Table[If[i ≤ j, 1, 0], {i, 3}, {j, 4}];
mat3 // MatrixForm
```

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

```
Table[Random[], {3}, {3}] // MatrixForm
```

$$\begin{pmatrix} 0.63025 & 0.255601 & 0.994947 \\ 0.760625 & 0.675664 & 0.615577 \\ 0.166102 & 0.337102 & 0.796316 \end{pmatrix}$$

```
Clear[UDM];
UDM[f_Function, n_Integer?(# > 0 &)] := Table[If[i ≤ j, f[i, j], 0], {i, n}, {j, n}];
```

```
UDM[#1^#2 &, 3] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & 8 \\ 0 & 0 & 27 \end{pmatrix}$$

```
UDM[x[#1, #2] &, 3] // MatrixForm
```

$$\begin{pmatrix} x[1, 1] & x[1, 2] & x[1, 3] \\ 0 & x[2, 2] & x[2, 3] \\ 0 & 0 & x[3, 3] \end{pmatrix}$$

```
HilbertMatrix[{2, 3}] // MatrixForm
```

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \end{pmatrix}$$

```
HankelMatrix[{u, v, w}] // MatrixForm
```

$$\begin{pmatrix} u & v & w \\ v & w & 0 \\ w & 0 & 0 \end{pmatrix}$$

° ist die Kurzform der Konstanten **Degree**, kann auch als ESC deg ESC eingegeben werden. In entsprechenden Kontexten wird sie durch $\frac{\pi}{180}$ ersetzt.

```
Sin[12 ° + π / 180]
```

$$\text{Sin}\left[12 ^\circ + \frac{\pi}{180}\right]$$

```
Sin[12 ° + π / 180] // FullForm
```

```
Sin[Plus[Times[12, Degree], Times[Rational[1, 180], Pi]]]
```

```
Sin[12 ° + π / 180] // FullSimplify
```

$$\text{Sin}\left[\frac{13\,\pi}{180}\right]$$

```
RotationMatrix[30 °] // MatrixForm
```

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

```
RotationMatrix[30 °, {1, 1, 1}] // MatrixForm
```

$$\begin{pmatrix} \frac{1}{3}\left(1+\sqrt{3}\right) & \frac{1}{3}\left(1-\sqrt{3}\right) & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3}\left(1+\sqrt{3}\right) & \frac{1}{3}\left(1-\sqrt{3}\right) \\ \frac{1}{3}\left(1-\sqrt{3}\right) & \frac{1}{3} & \frac{1}{3}\left(1+\sqrt{3}\right) \end{pmatrix}$$

Matrizen mit indizierten Variablen.
Die erste Möglichkeit x als zweistellige Funktion

```
Clear[x];
mat = Table[x[i, j], {i, 0, 2}, {j, 0, 2}];
mat // MatrixForm
```

$$\begin{pmatrix} x[0, 0] & x[0, 1] & x[0, 2] \\ x[1, 0] & x[1, 1] & x[1, 2] \\ x[2, 0] & x[2, 1] & x[2, 2] \end{pmatrix}$$

Die zweite Möglichkeit: x mit Subscript−Indizes

```
mat = Table[x_{i,j}, {i, 0, 2}, {j, 0, 2}];
mat // MatrixForm
```

$$\begin{pmatrix} x_{0,0} & x_{0,1} & x_{0,2} \\ x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \end{pmatrix}$$

Aber beachten Sie, dass es sich auch hierbei um zusammengesetzte Bezeichner handelt!

```
mat // InputForm
```

```
{{Subscript[x, 0, 0], Subscript[x, 0, 1], Subscript[x, 0, 2]},
 {Subscript[x, 1, 0], Subscript[x, 1, 1], Subscript[x, 1, 2]},
 {Subscript[x, 2, 0], Subscript[x, 2, 1], Subscript[x, 2, 2]}}
```

▪ Auf einzelne Matrixelemente zugreifen

```
ClearAll["Global`*"];
mat = ⎛a b c d⎞
      ⎜e f g h⎟;
      ⎝i j k l⎠
```

```
mat[[2, 3]]
```

```
g
```

Achtung, das ist die zweite Zeile obiger Matrix, aber ein Vektor und von **MatrixForm** als Spaltenvektor dargestellt.

```
mat[[2]] // MatrixForm
```

$$\begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

```
Transpose[mat][[2]] // MatrixForm
```

$$\begin{pmatrix} b \\ f \\ j \end{pmatrix}$$

```
mat[[{2, 3}, {1, 4, 3}]] // MatrixForm
```

$$\begin{pmatrix} e & h & g \\ i & l & k \end{pmatrix}$$

```
mat[[2 ;; 3, 1 ;; 4]] // MatrixForm
```

$$\begin{pmatrix} e & f & g & h \\ i & j & k & l \end{pmatrix}$$

```
mat[[Range[2, 3], Range[1, 4]]] // MatrixForm
```

$$\begin{pmatrix} e & f & g & h \\ i & j & k & l \end{pmatrix}$$

■ Mit Matrizen rechnen

```
ClearAll["Global`*"]
```

```
mat1 = {{a, b}, {c, c}, {e, f}}; mat2 = {{r, s, t}, {u, v, w}};
```

```
3 mat1 // MatrixForm
```

$$\begin{pmatrix} 3a & 3b \\ 3c & 3c \\ 3e & 3f \end{pmatrix}$$

```
mat2 + Transpose[mat1] // MatrixForm
```

$$\begin{pmatrix} a+r & c+s & e+t \\ b+u & c+v & f+w \end{pmatrix}$$

```
mat1 . mat2 // MatrixForm
```

$$\begin{pmatrix} ar+bu & as+bv & at+bw \\ cr+cu & cs+cv & ct+cw \\ er+fu & es+fv & et+fw \end{pmatrix}$$

```
mat = {{a, b, c}, {d, e, f}, {g, h, i}}; vec = {u, v, w};
```

Bei der Multiplikation mit einer Matrix von links wird ein Vektor als Zeilenvektor, bei der Multiplikation von rechts als Spaltenvektor interpretiert. In beiden Fällen wird ein Vektor erzeugt, der im ersten Fall aber von **MatrixForm** als Spaltenvektor und nicht als Zeilenvektor dargestellt wird.

```
vec . mat // MatrixForm
```

$$\begin{pmatrix} a\,u + d\,v + g\,w \\ b\,u + e\,v + h\,w \\ c\,u + f\,v + i\,w \end{pmatrix}$$

```
mat . vec // MatrixForm
```

$$\begin{pmatrix} a\,u + b\,v + c\,w \\ d\,u + e\,v + f\,w \\ g\,u + h\,v + i\,w \end{pmatrix}$$

Besser ist es deshalb, wenn Sie in diesen Fällen nicht mit Vektoren, sondern mit einzeiligen oder einspaltigen Matrizen rechnen. Eine einzeilige Matrix lässt sich aus einem Vektor einfach durch zusätzliche Listenklammern herstellen, eine einspaltige Matrix durch **List/@...** . Das Produkt wird nun von **MatrixForm** auch wie erwartet dargestellt.

```
(esm = List /@ vec) // MatrixForm
mat.esm // MatrixForm
Transpose[esm]. mat // MatrixForm
```

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

$$\begin{pmatrix} a\,u + b\,v + c\,w \\ d\,u + e\,v + f\,w \\ g\,u + h\,v + i\,w \end{pmatrix}$$

$$\begin{pmatrix} a\,u + d\,v + g\,w & b\,u + e\,v + h\,w & c\,u + f\,v + i\,w \end{pmatrix}$$

```
mat1 = {{1, 2, 3}, {4, -5, 6}, {9, 8, 7}};
```

```
Det[mat1]
```

```
200
```

```
mat2 = Inverse[mat1]; mat2 // MatrixForm
```

$$\begin{pmatrix} -\frac{83}{200} & \frac{1}{20} & \frac{27}{200} \\ \frac{13}{100} & -\frac{1}{10} & \frac{3}{100} \\ \frac{77}{200} & \frac{1}{20} & -\frac{13}{200} \end{pmatrix}$$

```
MatrixPower[{{a, b}, {c, d}}, 3] // MatrixForm
```

$$\begin{pmatrix} a\,(a^2 + b\,c) + b\,(a\,c + c\,d) & a\,(a\,b + b\,d) + b\,(b\,c + d^2) \\ c\,(a^2 + b\,c) + d\,(a\,c + c\,d) & c\,(a\,b + b\,d) + d\,(b\,c + d^2) \end{pmatrix}$$

```
MatrixPower[{{1, 2}, {3, 4}}, 20] // MatrixForm
```

$$\begin{pmatrix} 95\,799\,031\,216\,999 & 139\,620\,104\,992\,450 \\ 209\,430\,157\,488\,675 & 305\,229\,188\,705\,674 \end{pmatrix}$$

```
MatrixExp[{{0.1, 0.2}, {0.3, 0.4}}] // MatrixForm
```

$$\begin{pmatrix} 1.14209 & 0.260351 \\ 0.390526 & 1.53262 \end{pmatrix}$$

```
Sum[MatrixPower[{{0.1, 0.2}, {0.3, 0.4}}, i] / i!, {i, 0, 5}] // MatrixForm
```

$$\begin{pmatrix} 1.14208 & 0.260338 \\ 0.390507 & 1.53259 \end{pmatrix}$$

■ Dünn besetzte Matrizen

Große Matrizen (und allgemeiner Tensoren) können mit **SparseArray** erzeugt werden.
Diese dünne Darstellung verwendet intern eine andere Speicherform als die bisher betrachtete dichte Darstellung von Matrizen.

```
SparseArray[{{1, 1} → 1, {2, 2} → 2, {3, 3} → 3, {1, 5} → 4}] // MatrixForm
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 4 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \end{pmatrix}$$

```
SparseArray[{{1, 1} → 1, {2, 2} → 2, {3, 3} → 3, {1, 5} → 4}, 5, x] // MatrixForm
```

$$\begin{pmatrix} 1 & x & x & x & 4 \\ x & 2 & x & x & x \\ x & x & 3 & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix}$$

```
SparseArray[{{i_, i_, i_} -> 2}, 3] // TensorRank
```

```
3
```

```
m = SparseArray[{{i_, i_} -> 1}, 100]
```

```
SparseArray[<100>, {100, 100}]
```

```
ByteCount /@ {m, IdentityMatrix[100]}
```

```
{1656, 40 080}
```

```
m == IdentityMatrix[100]
```

```
True
```

```
m + IdentityMatrix[100] // Shallow
```

```
{{2, 0, 0, 0, 0, 0, 0, 0, 0, 0, <<90>>}, {0, 2, 0, 0, 0, 0, 0, 0, 0, 0, <<90>>},
 {0, 0, 2, 0, 0, 0, 0, 0, 0, 0, <<90>>}, {0, 0, 0, 2, 0, 0, 0, 0, 0, 0, <<90>>},
 {0, 0, 0, 0, 2, 0, 0, 0, 0, 0, <<90>>}, {0, 0, 0, 0, 0, 2, 0, 0, 0, 0, <<90>>},
 {0, 0, 0, 0, 0, 0, 2, 0, 0, 0, <<90>>}, {0, 0, 0, 0, 0, 0, 0, 2, 0, 0, <<90>>},
 {0, 0, 0, 0, 0, 0, 0, 0, 2, 0, <<90>>}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 2, <<90>>}, <<90>>}
```

```
SparseArray[{Band[{1, 1}, {10, 10}] → 2}, 10] // MatrixForm
```

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

```
SparseArray[{Band[{1, 1}, {10, 10}] → {2, 1}}, 10] // MatrixForm
```

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
m = SparseArray[{Band[{1, 1}, {5, 10}, {1, 2}] → 2}, {5, 10}];
m // MatrixForm
```

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{pmatrix}$$

```
m // ArrayRules
```

```
{{1, 1} → 2, {2, 3} → 2, {3, 5} → 2, {4, 7} → 2, {5, 9} → 2, {_, _} → 0}
```

```
Clear[n];
sparsemat := SparseArray[{Band[{1, 1}] → 2, Band[{2, 1}] → 1, Band[{1, 2}] → 1}, n];
```

```
Block[{n = 5}, sparsemat // MatrixForm]
```

$$\begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$

```
(n = #; ByteCount /@ {sparsemat, sparsemat // Normal}) & /@ {10, 20, 50}
```

```
{{736, 480}, {1016, 1680}, {1856, 10 080}}
```

■ Lineare Gleichungssysteme lösen

Die einfachste und allgemeinste Weg zum Lösen eines linearen Gleichungssystems geht über die **Solve**−Funktion. Über die ausgegebenen Warnungen muss man sich keine Gedanken machen.Wir verwenden im Gegensatz zum *Mathematica*−Handbuch die Notation der Vektoren als einspaltige Matrizen.

$$\text{mat} = \begin{pmatrix} 1 & 2 & 3 & 1 \\ 3 & 1 & 2 & -1 \\ 2 & 3 & 1 & 0 \end{pmatrix}; \quad b = \begin{pmatrix} 14 \\ 12 \\ 7 \end{pmatrix}; \quad x = \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix};$$

```
sol = Solve[mat.x == b]
```

```
Solve::svars :
 Equations may not give solutions for all "solve" variables.
```

$$\left\{\left\{x1 \to \frac{7}{6} + \frac{2\,x4}{3},\ x2 \to \frac{1}{6} - \frac{x4}{3},\ x3 \to \frac{25}{6} - \frac{x4}{3}\right\}\right\}$$

Die Solve−Funktion gibt als Anwort eine Substitutionsliste zurück, aus der sich die allgemeine Lösung durch Substitution in die einspaltige Matrix **x** gewinnen lässt. Beachten Sie, dass die Lösung eine einelementige *Menge* ist.

```
MatrixForm /@ (x /. sol /. x4 → t)
```

$$\left\{\begin{pmatrix} \frac{7}{6} + \frac{2\,t}{3} \\ \frac{1}{6} - \frac{t}{3} \\ \frac{25}{6} - \frac{t}{3} \\ t \end{pmatrix}\right\}$$

Gelegentlich muss die Menge der Variablen angegeben werden, etwa wenn die Ausdrücke noch weitere Parameter enthalten. Diese müssen dazu aus **x** extrahiert werden.

```
vars = Flatten[x]
sol = Solve[mat.x == b, vars]
```

```
{x1, x2, x3, x4}
```

```
Solve::svars :
 Equations may not give solutions for all "solve" variables.
```

$$\left\{\left\{x1 \to \frac{7}{6} + \frac{2\,x4}{3},\ x2 \to \frac{1}{6} - \frac{x4}{3},\ x3 \to \frac{25}{6} - \frac{x4}{3}\right\}\right\}$$

Lineare Gleichungssysteme lassen sich auch mit dem Kommando **LinearSolve** lösen. Dieses ist besonders dann angebracht, wenn es sich um große lineare Gleichungssysteme handelt oder ein und dasselbe System mit vielen verschiedenen rechten Seiten gelöst werden soll.
Es gibt zwei Aufrufvarianten des Kommandos **LinearSolve**. Die erste Variante **LinearSolve[mat]** bekommt nur die Koeffizientenmatrix als Parameter übergeben und generiert daraus ein **LinearSolveFunction**−Objekt.

```
lsol = LinearSolve[mat]
```

```
LinearSolve::sqmat1 :
 The matrix {{1, 2, 3, 1}, {3, 1, 2, -1}, {2, 3, 1, 0}}
   is not square so a factorization will not be saved.
```

```
LinearSolveFunction[{3, 4}, <>]
```

Es handelt sich dabei wie bei einer **InterpolatingFunction** um eine namenlose Funktion, von der auch die Berechnungsvorschrift nicht explizit bekannt ist. Sie berechnet zu einer vorgegebenen rechten Seite **b** eine Lösung des Gleichungssystems.

```
lsol[b] // MatrixForm
```

$$\begin{pmatrix} \frac{7}{6} \\ \frac{1}{6} \\ \frac{25}{6} \\ 0 \end{pmatrix}$$

Die zweite Aufrufvariante **LinearSolve[mat,b]** bekommt als zweiten Parameter die rechte Seite übergeben und entspricht dem Aufruf **LinearSolve[mat][b].**

```
(l0 = LinearSolve[mat, b]) // MatrixForm
```

$$\begin{pmatrix} \frac{7}{6} \\ \frac{1}{6} \\ \frac{25}{6} \\ 0 \end{pmatrix}$$

Im Gegensatz zu **Solve** gibt **LinearSolve** (im Fall eines lösbaren Systems) immer nur eine spezielle Lösung zurück. Da die allgemeine Lösung eines linearen Gleichungssystems sich zusammensetzt aus einer solchen speziellen Lösung und der allgemeinen Lösung des zugehörigen homogenen linearen Gleichungssystems, bleibt eine Basis des Lösungsraums des homogenen Systems zu bestimmen. Dies kann mit dem Kommando **NullSpace** erfolgen.

```
MatrixForm /@ (ns = NullSpace[mat])
```

$$\left\{\begin{pmatrix} 2 \\ -1 \\ -1 \\ 3 \end{pmatrix}\right\}$$

Beachten Sie, dass **NullSpace** eine Liste von *Vektoren* und nicht von *einspaltigen Matrizen* zurückgibt, so dass noch einige Transformationen erforderlich sind, um daraus die allgemeine Lösung zu extrahieren.

```
{{t}.ns} // Transpose // MatrixForm
```

$$\begin{pmatrix} 2\,t \\ -t \\ -t \\ 3\,t \end{pmatrix}$$

```
l0 + Transpose[{{t}.ns}] // MatrixForm
```

$$\begin{pmatrix} \frac{7}{6} + 2\,t \\ \frac{1}{6} - t \\ \frac{25}{6} - t \\ 3\,t \end{pmatrix}$$

Und hier noch ein Beispiel mit einer mehrdimensionalen Lösungsraum.

```
mat = {{1, 2, 3, 1}, {3, 1, 2, -1}}; b = {{14}, {12}}; x = {{x1}, {x2}, {x3}, {x4}};
```

```
(l0 = LinearSolve[mat, b]) // MatrixForm
```

$$\begin{pmatrix} 2 \\ 6 \\ 0 \\ 0 \end{pmatrix}$$

```
MatrixForm /@ (ns = NullSpace[mat])
```

$$\left\{\begin{pmatrix} 3 \\ -4 \\ 0 \\ 5 \end{pmatrix}, \begin{pmatrix} -1 \\ -7 \\ 5 \\ 0 \end{pmatrix}\right\}$$

```
Transpose[{{t1, t2}.ns}] // MatrixForm
```

$$\begin{pmatrix} 3\,t1 - t2 \\ -4\,t1 - 7\,t2 \\ 5\,t2 \\ 5\,t1 \end{pmatrix}$$

```
l0 + Transpose[{{t1, t2}.ns}] // MatrixForm
```

$$\begin{pmatrix} 2 + 3\,t1 - t2 \\ 6 - 4\,t1 - 7\,t2 \\ 5\,t2 \\ 5\,t1 \end{pmatrix}$$

Es geht allerdings auch einfacher, da die Summe einer einspaltigen Matrix und eines Vektors berechnet werden kann und wieder eine einspaltige Matric ergibt.

```
l0 + {t1, t2}.ns // MatrixForm
```

$$\begin{pmatrix} 2 + 3\,t1 - t2 \\ 6 - 4\,t1 - 7\,t2 \\ 5\,t2 \\ 5\,t1 \end{pmatrix}$$

Und nun noch einmal Matrizen in dünner Darstellung.

```
Clear[m, n, x, b];
m := SparseArray[{Band[{1, 1}] → 2, Band[{2, 1}] → 1, Band[{1, 2}] → 1}, n];
vx := Table[x_i, {i, 1, n}];
b := Table[1, {n}];

(n = #; Det[m]) & /@ Range[5, 15]
```

```
{6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}
```

Die Rechnung legt nahe, dass die Determinante gleich $n+1$ ist. Zum Beweis dieser Beziehung muss aus der Definition der Determinante eine Rekursionsbeziehung hergeleitet werden, die zusammen mit den entsprechenden Startwerten dann von *Mathematica* gelöst werden kann.

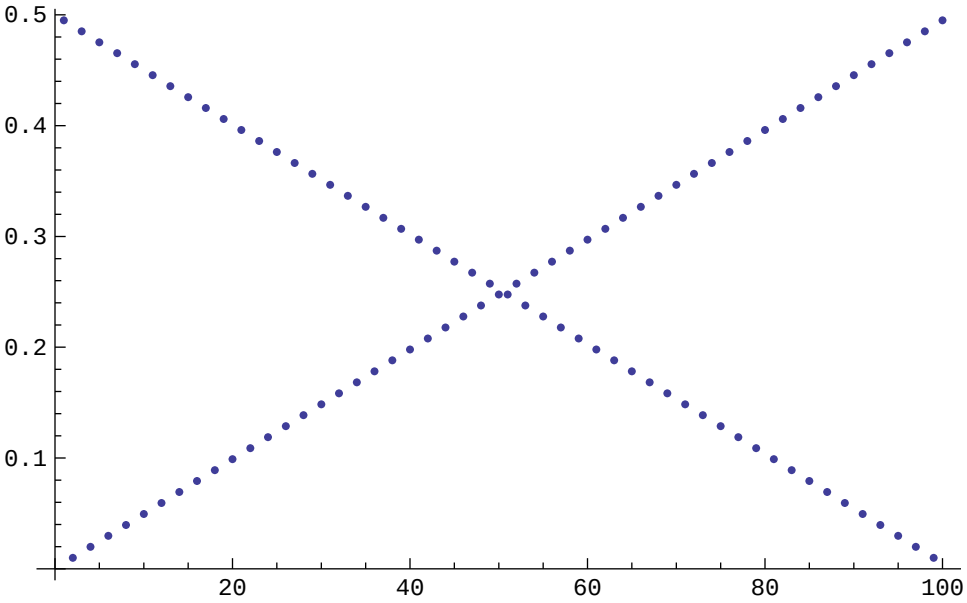
```
Clear[n];
RSolve[{d[n] == 2 d[n - 1] - d[n - 2], d[1] == 2, d[2] == 3}, d[n], n]
```

```
{{d[n] → 1 + n}}
```

```
(n = 100; sol1 = Solve[m.vx == b, vx];) // Timing
```

```
{0., Null}
```

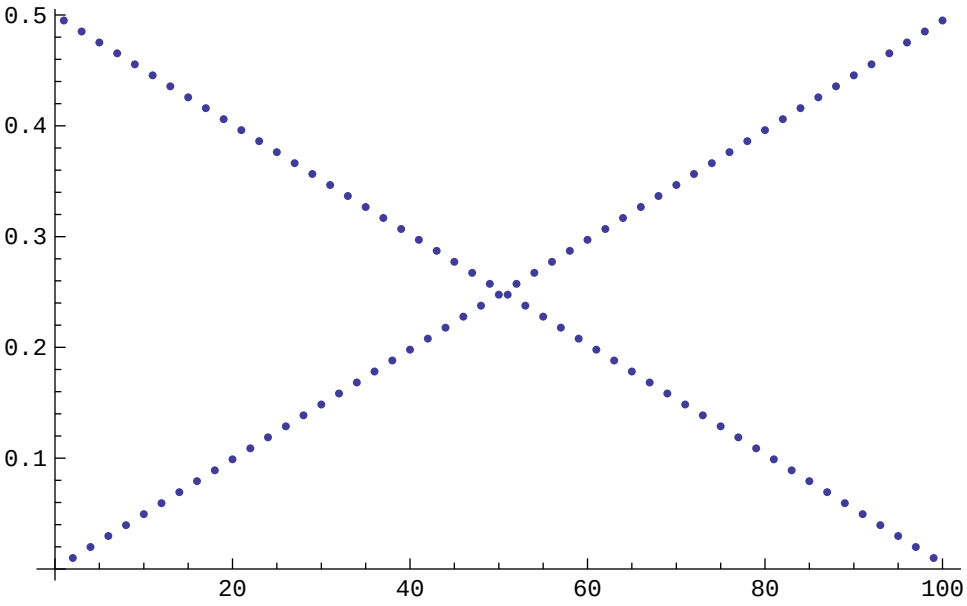
```
ListPlot[Evaluate[vx /. sol1]]
```




```
(n = 100; sol2 = LinearSolve[m, b];) // Timing
```

```
{1.36557 × 10^-14, Null}
```

```
ListPlot[sol2]
```




```
Clear[a];
b := Table[a_i, {i, n}]

(n = 10; sol1 = Solve[m.vx == b, vx];) // Timing
sol1
```

```
{0.02, Null}
```

$$\left\{\left\{x_1 \to \frac{1}{11}\,(10\,a_1 - 9\,a_2 + 8\,a_3 - 7\,a_4 + 6\,a_5 - 5\,a_6 + 4\,a_7 - 3\,a_8 + 2\,a_9 - a_{10}),\right.\right.$$
$$x_2 \to \frac{1}{11}\,(-9\,a_1 + 18\,a_2 - 16\,a_3 + 14\,a_4 - 12\,a_5 + 10\,a_6 - 8\,a_7 + 6\,a_8 - 4\,a_9 + 2\,a_{10}),$$
$$x_3 \to \frac{1}{11}\,(8\,a_1 - 16\,a_2 + 24\,a_3 - 21\,a_4 + 18\,a_5 - 15\,a_6 + 12\,a_7 - 9\,a_8 + 6\,a_9 - 3\,a_{10}),$$
$$x_4 \to \frac{1}{11}\,(-7\,a_1 + 14\,a_2 - 21\,a_3 + 28\,a_4 - 24\,a_5 + 20\,a_6 - 16\,a_7 + 12\,a_8 - 8\,a_9 + 4\,a_{10}),$$
$$x_5 \to \frac{1}{11}\,(6\,a_1 - 12\,a_2 + 18\,a_3 - 24\,a_4 + 30\,a_5 - 25\,a_6 + 20\,a_7 - 15\,a_8 + 10\,a_9 - 5\,a_{10}),$$
$$x_6 \to \frac{1}{11}\,(-5\,a_1 + 10\,a_2 - 15\,a_3 + 20\,a_4 - 25\,a_5 + 30\,a_6 - 24\,a_7 + 18\,a_8 - 12\,a_9 + 6\,a_{10}),$$
$$x_7 \to \frac{1}{11}\,(4\,a_1 - 8\,a_2 + 12\,a_3 - 16\,a_4 + 20\,a_5 - 24\,a_6 + 28\,a_7 - 21\,a_8 + 14\,a_9 - 7\,a_{10}),$$
$$x_8 \to \frac{1}{11}\,(-3\,a_1 + 6\,a_2 - 9\,a_3 + 12\,a_4 - 15\,a_5 + 18\,a_6 - 21\,a_7 + 24\,a_8 - 16\,a_9 + 8\,a_{10}),$$
$$x_9 \to \frac{1}{11}\,(2\,a_1 - 4\,a_2 + 6\,a_3 - 8\,a_4 + 10\,a_5 - 12\,a_6 + 14\,a_7 - 16\,a_8 + 18\,a_9 - 9\,a_{10}),$$
$$\left.\left.x_{10} \to \frac{1}{11}\,(-a_1 + 2\,a_2 - 3\,a_3 + 4\,a_4 - 5\,a_5 + 6\,a_6 - 7\,a_7 + 8\,a_8 - 9\,a_9 + 10\,a_{10})\right\}\right\}$$

```
(n = 10; sol2 = LinearSolve[m, b];) // Timing
sol2
```

```
{0.02, Null}
```

$$\left\{\frac{1}{11}(10\,a_1 - 9\,a_2 + 8\,a_3 - 7\,a_4 + 6\,a_5 - 5\,a_6 + 4\,a_7 - 3\,a_8 + 2\,a_9 - a_{10}),\right.$$
$$\frac{1}{11}(-9\,a_1 + 18\,a_2 - 16\,a_3 + 14\,a_4 - 12\,a_5 + 10\,a_6 - 8\,a_7 + 6\,a_8 - 4\,a_9 + 2\,a_{10}),$$
$$\frac{1}{11}(8\,a_1 - 16\,a_2 + 24\,a_3 - 21\,a_4 + 18\,a_5 - 15\,a_6 + 12\,a_7 - 9\,a_8 + 6\,a_9 - 3\,a_{10}),$$
$$\frac{1}{11}(-7\,a_1 + 14\,a_2 - 21\,a_3 + 28\,a_4 - 24\,a_5 + 20\,a_6 - 16\,a_7 + 12\,a_8 - 8\,a_9 + 4\,a_{10}),$$
$$\frac{1}{11}(6\,a_1 - 12\,a_2 + 18\,a_3 - 24\,a_4 + 30\,a_5 - 25\,a_6 + 20\,a_7 - 15\,a_8 + 10\,a_9 - 5\,a_{10}),$$
$$\frac{1}{11}(-5\,a_1 + 10\,a_2 - 15\,a_3 + 20\,a_4 - 25\,a_5 + 30\,a_6 - 24\,a_7 + 18\,a_8 - 12\,a_9 + 6\,a_{10}),$$
$$\frac{1}{11}(4\,a_1 - 8\,a_2 + 12\,a_3 - 16\,a_4 + 20\,a_5 - 24\,a_6 + 28\,a_7 - 21\,a_8 + 14\,a_9 - 7\,a_{10}),$$
$$\frac{1}{11}(-3\,a_1 + 6\,a_2 - 9\,a_3 + 12\,a_4 - 15\,a_5 + 18\,a_6 - 21\,a_7 + 24\,a_8 - 16\,a_9 + 8\,a_{10}),$$
$$\frac{1}{11}(2\,a_1 - 4\,a_2 + 6\,a_3 - 8\,a_4 + 10\,a_5 - 12\,a_6 + 14\,a_7 - 16\,a_8 + 18\,a_9 - 9\,a_{10}),$$
$$\left.\frac{1}{11}(-a_1 + 2\,a_2 - 3\,a_3 + 4\,a_4 - 5\,a_5 + 6\,a_6 - 7\,a_7 + 8\,a_8 - 9\,a_9 + 10\,a_{10})\right\}$$

■ Eigenwerte und Eigenvektoren

```
ClearAll["Global`*"]
```

$$\text{mat} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & -1 & 0 \end{pmatrix};$$

```
CharacteristicPolynomial[mat, x]
```

$$-9 + 6\,x^2 - x^3$$

```
Det[mat - x IdentityMatrix[3]]
```

$$-9 + 6\,x^2 - x^3$$

```
ew = Eigenvalues[mat // N]
```

```
{5.72545, 1.39853, -1.12398}
```

```
MatrixForm /@ (ev = Eigenvectors[mat // N])
```

$$\left\{\begin{pmatrix} 0.329575 \\ 0.938128 \\ -0.106289 \end{pmatrix}, \begin{pmatrix} -0.0693105 \\ -0.834274 \\ 0.546975 \end{pmatrix}, \begin{pmatrix} 0.782984 \\ 0.0914102 \\ -0.615289 \end{pmatrix}\right\}$$

```
mat.ev[1] == ew[1] ev[1]
```

```
True
```