Module 13 Vectoren en matrices

Onderwerp	Definitie van vectoren en matrices; eenvoudige manipulaties met vectoren en matrices
Voorkennis	Vectoren: optellen, norm, inproduct, vectorproduct; Matrices: vermenigvuldiging, elementaire operaties.
Expressies	`Vector, Matrix, < >, |, %T, Transpose, fill, shape, IdentityMatrix, UnitVector, DiagonalMatrix, RandomMatrix, Dimension, . , RowOperation, ColumnOperation, inshape, Copy, Equal, Norm, Determinant, Trace, MatrixInverse, &x, Diagonal`
Bibliotheken	`LinearAlgebra`
Zie ook	Module 8, 14

13.1 Invoeren van matrices en vectoren in Maple

Een lijst, dat wil zeggen: een geordende rij van bij elkaar horende elementen (zie Module 8), kan worden opgevat als een *vector*. Op dezelfde manier zou men een *matrix* kunnen representeren als een lijst van lijsten.
Om gemakkelijker allerlei standaardbewerkingen met matrices en vectoren te kunnen verrichten (matrixvermenigvuldiging, inproduct berekenen, enzovoort) én om Maple in de gelegenheid te stellen die berekeningen zo efficiënt mogelijk uit te voeren, zijn de Maple-types `Vector` en `Matrix` gemaakt.

`Vector`
`Matrix`

Voorbeeldsessie

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

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

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

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

```
> whattype(v);
```

$$Vector_{column}$$

```
> Vector[row](lijst);
```

$$[1, 2, 3]$$

```
> A := Matrix([[1,2,3],[4,5,6]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
> whattype(A);
```

$$Matrix$$

```
> op(A);
```

$$2,\ 3,\ \{(1,\ 1) = 1,\ (1,\ 2) = 2,\ (1,\ 3) = 3,\ (2,\ 1) = 4,\ (2,\ 2) = 5,\ (2,\ 3) = 6\},\ datatype = anything,\ storage = rectangular,\ order = Fortran\_order,\ shape = []$$

```
> v[2];
```

$$2$$

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

$$4$$

```
> A[2,1] := 14:  A;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 14 & 5 & 6 \end{bmatrix}$$

Toelichting

Met het commando `Vector` maakt men een kolomvector van een lijst; Een kolomvector is de 'normale' vorm van een vector; een *rij*vector kan worden gemaakt met `Vector[row]`.
Een *matrix* A wordt gemaakt van een lijst, met de rijen van A als lijsten. Het resultaat van `op(A)` laat zien wat er zoal wordt vastgelegd: de afmetingen (2 rijen en 3 kolommen), de inhoud ($A_{11} = 1, A_{12} = 2$, enzovoort.
A_{ij}, het $(i,j)^{\text{de}}$ element van A wordt verkregen met `A[i,j]`. De waarde van A_{ij} kan op deze manier ook veranderd worden. ◇

> **!** Pas op dat u `Matrix` en `Vector` gebruikt, dus met een hoofdletter. De types `matrix` en `vector` bestaan namelijk óók (nog), maar hebben een andere betekenis.

Uiteraard kan men ook het palet, links van het werkblad, gebruiken om een matrix in te voeren; een kolomvector is een $n \times 1$-matrix. In het volgende voorbeeld laten we verschillende manieren zien, zodat u uw favoriete manier kunt kiezen..

Voorbeeldsessie

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

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

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

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

```
> v^%T;
```

$$[1, 2, 3]$$

```
> LinearAlgebra:-Transpose(v);
```

$$[1, 2, 3]$$

Matrix rij voor rij

```
> A := Matrix[[1,2,3],[4,5,6]]:
```

Dit kan worden afgekort tot

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

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Per kolom:

```
> A := <<1,4>|<2,5>|<3,6>>;
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
> A^%T;
```

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

```
> LinearAlgebra:-Transpose(A);
```

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

Toelichting

`< ... >`

In het algemeen geldt dat een expressie van de vorm `< ... >` een Vector of Matrix is. De expressies op de puntjes worden gescheiden door komma's (betekenis: 'nieuwe regel') of verticale streepjes (betekenis: 'nieuwe kolom').

`|`

De *getransponeerde* vector $\mathbf{v}^{\mathrm{T}}$ of matrix A^{T} wordt verkregen met `A^%T` of met `Transpose` uit de bibliotheek `LinearAlgebra`.[32] ◇

`^%T`
`Transpose`

Een andere manier om een matrix (of vector) in te voeren is vooral handig als hij veel nullen bevat.

[32] Maple-woorden zijn vaak nogal lang. Maar als u een paar letters getypt hebt, maakt Maple het meestal af met [Ctrl]-[Space] (in Windows) of [Ctrl]-[Shift]-[Space] (in Linux).

Voorbeeldsessie

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

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

```
> A[1,1] := 1:  A[2,3] := 8:
> A;
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 8 \end{bmatrix}$$

```
> B := Matrix(2,3,fill=1/2);
```

$$B := \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

```
> C := Matrix(3,4,[[1,2],[],[4,5]],fill=23);
```

$$C := \begin{bmatrix} 1 & 2 & 23 & 23 \\ 23 & 23 & 23 & 23 \\ 4 & 5 & 23 & 23 \end{bmatrix}$$

Toelichting

Als we in het `Matrix`-commando alleen de afmeting aangeven, dan wordt het automatisch een nulmatrix. De waarden die van 0 verschillen, kunnen we dan later toevoegen.

`fill` Met de optie `fill` kunnen we de matrix ook met andere waarden vullen.

Als we wél de afmeting van een matrix geven, maar hem 'onvolledig' vullen, dan vult Maple hem automatisch verder op met nullen, of met de waarde die we met een `fill`-optie meegeven. ◇

13.2 Matrices samenstellen, submatrices

Rijen of kolommen selecteren. Zoals Met `A[i,j]` het $(i,j)^{\text{de}}$ element van matrix A wordt geselecteerd, kan met `A[i..j,k..l]` een submatrix van A worden gemaakt, bestaande uit: van rij i t/m j, de k^{de} t/m de ℓ^{de} kolom. Het nummer van het *laatste* element van een rij of kolom kunnen we altijd met -1 aangeven, het één na laatste element met -2, enzovoort.

Voorbeeldsessie

```
> A := <1,-2,1,0; 0,2,-8,1; -4,5,9,-2>;
```

$$A := \begin{bmatrix} 1 & -2 & 1 & 0 \\ 0 & 2 & -8 & 1 \\ -4 & 5 & 9 & -2 \end{bmatrix}$$

De tweede kolom van A krijgen we met

```
> A[1..-1, 2];
```

$$\begin{bmatrix} -2 \\ 2 \\ 5 \end{bmatrix}$$

en de derde *rij*:

```
> A[3, 1..-1];
```

$$[-4, 5, 9, -2]$$

Toelichting

Uiteraard hadden we ook `A[1..3,2]`, respectievelijk `A[3,1..4]` kunnen doen. ◇

Samenstellen van matrices. Vaak zal men een *aangevulde matrix* willen maken: A^a is de matrix A, met een vector $\mathbf{b}$ als laatste kolom toegevoegd.
In het volgende voorbeeld is A dezelfde als in het vorige voorbeeld.

Voorbeeldsessie

```
> A := <1,-2,1,0; 0,2,-8,1; -4,5,9,-2>:
```

Een kolomvector:

```
> b := <5,-1,11>;
```

$$b := \begin{bmatrix} 5 \\ -1 \\ 11 \end{bmatrix}$$

```
> Aa := <A|b>;
```

$$\begin{bmatrix} 1 & -2 & 1 & 0 & 5 \\ 0 & 2 & -8 & 1 & -1 \\ -4 & 5 & 9 & -2 & 11 \end{bmatrix}$$

Een rijvector

```
> c := <6|5|4|3>;
```

$$c := [6, 5, 4, 3]$$

```
> <A,c>;
```

$$\begin{bmatrix} 1 & -2 & 1 & 0 \\ 0 & 2 & -8 & 1 \\ -4 & 5 & 9 & -2 \\ 6 & 5 & 4 & 3 \end{bmatrix}$$

Andere mogelijkheid

```
> <A|b|A>;
```

$$\begin{bmatrix} 1 & -2 & 1 & 0 & 5 & 1 & -2 & 1 & 0 \\ 0 & 2 & -8 & 1 & -1 & 0 & 2 & -8 & 1 \\ -4 & 5 & 9 & -2 & 11 & -4 & 5 & 9 & -2 \end{bmatrix}$$

Toelichting

Met behulp van `<..|..>` en `<..,..>` kunnen we willekeurig matrices en vectoren 'aan elkaar plakken', mits de afmetingen kloppen, natuurlijk. ◇

13.3 Speciale matrices

Indexfuncties. Soms kan van een matrix A het element a_{ij} uit i en j worden berekend. We noemen het voorschrift f, zodat $f(i,j) = a_{ij}$ een *indexfunctie*. Hij kan als derde argument aan een `Matrix`-commando worden meegegeven. Een bestaat ook een aantal standaard-indexfuncties. Deze worden in de vorm van een `shape`-optie aan `Matrix` doorgegeven.

`shape`

Voorbeeldsessie

```
> A := Matrix( 3,4, (i,j) -> x^abs(i-j) );
```

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

Diagonaalmatrix

```
> B := Matrix( <2,4,3>, shape=diagonal );
```

$$B := \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

```
> B[1,3] := 9;

Error, attempt to assign non-zero to off-diagonal entry of a
diagonal Matrix
```

Eenheidsmatrix

```
> I3 := Matrix( 3, shape=identity );
```

$$I3 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Een symmetrische matrix

```
> C := Matrix( 3, shape=symmetric );
```

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

```
> C[1,2] := 4:  C[1,3] := -5:  C;
```

$$\begin{bmatrix} 0 & 4 & -5 \\ 4 & 0 & 0 \\ -5 & 0 & 0 \end{bmatrix}$$

```
> S := Matrix( [[1],[2,3],[4,5,6]], shape=symmetric );
```

$$S := \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix}$$

Toelichting

De 3×4-matrix A is gedefinieerd door $a_{ij} = x^{|i-j|}$.
Als we een diagonaalmatrix definiëren met `shape=diagonal`, dan is het niet zonder meer mogelijk er een ander soort matrix van te maken. Als men dat zou willen, dan zou er bijvoorbeeld een nulmatrix van de juiste afmetingen bij opgeteld kunnen worden.
De *symmetrische matrix* C zonder verdere gegevens wordt uiteraard aanvankelijk een nulmatrix. Als c_{12} daarna een waarde krijgt, dan krijgt c_{21} automatisch dezelfde waarde. Als we hem in één keer willen 'vullen', dan hoeven we alleen de onderdriehoeksmatrix op te geven: van de eerste rij alleen het eerste element, van de tweede rij de eerste twee, enzovoort. ◇

Matrices maken met `LinearAlgebra`-commando's. De bibliotheek `LinearAlgebra` bevat een verontrustend groot aantal procedures. Informatie daarover krijgt u natuurlijk via `?LinearAlgebra`, maar het volgende is vaak ook handig. Als u de bibliotheek laadt, en `with(LinearAlgebra)` met een puntkomma afsluit (inplaats van de dubbele punt), dan krijgt u een lijst van alle beschikbare procedures. Als u de naam van de procedure waar u meer van wilt weten met de muis selekteert, dan krijgt u die informatie door op [F2] te drukken.

Met `LinearAlgebra` zijn onder andere de volgende soorten matrices gemakkelijk te maken:

`IdentityMatrix`
- *Eenheidsmatrices* met `IdentityMatrix`

`UnitVector`
- De *eenheidsvector* $\mathbf{e_i} \in \mathbb{R}^n$ krijgt men met het commando `UnitVector(i,n)` als kolomvector. Net als een eenheidsmatrix kan deze niet zonder meer veranderd worden.

`DiagonalMatrix`
- *Diagonaalmatrices* met `DiagonalMatrix`.
  Van een met `DiagonalMatrix` gemaakte matrix kunnen wél de elementen buiten de diagonaal worden veranderd.

`RandomMatrix`
- Een *willekeurige* matrix (of vector) met `RandomMatrix` (of `RandomVector`), handig om iets uit te testen.

Voorbeeldsessie

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

Eenheidsmatrix:

```
> I3 := IdentityMatrix(3);
```

$$I3 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> A := DiagonalMatrix([1,2,3]);
```

$$A := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Dit commando kan ook worden gebruikt om matrices te maken met 'blokken' op de diagonaal' (die niet per se vierkant hoeven te zijn), bijvoorbeeld

```
> B := DiagonalMatrix([2,3,<4,5,6;7,8,9;10,11,12>,<13,14,15>]);
```

$$B := \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 5 & 6 & 0 \\ 0 & 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 10 & 11 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 13 \\ 0 & 0 & 0 & 0 & 0 & 14 \\ 0 & 0 & 0 & 0 & 0 & 15 \end{bmatrix}$$

```
> C := RandomMatrix(2,3);
```

$$C := \begin{bmatrix} 99 & 44 & -31 \\ 29 & 92 & 67 \end{bmatrix}$$

```
> RandomMatrix( 2,3, generator=-5..9 );
```

$$\begin{bmatrix} 6 & 2 & 1 \\ -4 & 1 & -5 \end{bmatrix}$$

Toelichting

Met de optie `generator` in `RandomMatrix` kan men aangeven tussen welke grenzen de elementen moeten worden gekozen. ◇

13.4 Bewerkingen met matrices

`Dimension`

Afmetingen van matrices en vectoren. Met `Dimension` krijgen we de afmeting van een vector of matrix; als `A` een matrix is, dan geeft `Dimension(A)` *twee* getallen, namelijk respectievelijk het aantal rijen en kolommen. Desnoods kunt u met `RowDimension` of `ColumnDimension` afzonderlijk het aantal rijen of kolommen opvragen.

> **!** `RowDimension(A)` geeft dus niet de dimensie van de rijruimte van `A`, zie hiervoor Module 14.

Voorbeeldsessie

```
> with(LinearAlgebra):
> A := <<1,0,-4>|<-2,2,5>|<1,-8,9>|<0,1,-2>>:
> k,n := Dimension(A);
```

$$k,\, n := 3,\, 4$$

De tweede rij van A:

```
> b := A[2,1..-1]:
> Dimension(b);
```

$$4$$

`.`

Optelling, vermenigvuldiging. Voor het *optellen* van matrices of vectoren kunnen we de gewone `+` gebruiken. De *scalaire vermenigvuldiging*, dus vermenigvuldiging van een matrix of vector met een getal, gaat met het symbool `*`. Omdat de *matrixvermenigvuldiging* niet commutatief is (dat wil zeggen dat in het algemeen $AB \neq BA$ als A en B matrices zijn) wordt hiervoor een speciaal symbool gebruikt, namelijk de gewone punt (`.`). De uitdrukking $(A + 3B + CD)\,\mathbf{x}$, met A, B, C, D matrices en $\mathbf{x}$ een vector (alle van geschikte dimensies) wordt dan in Maple

```
(A + 3*B + C . D) . x;
```

Voorbeeldopgave

Bereken $(B + 3A)\,\mathbf{x}$ met

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Voorbeeldsessie

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

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

```
> x := <1,3>;
```

$$x := \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

```
> C := (B+3*A).x;
```

$$C := \begin{bmatrix} 28 \\ 27 \end{bmatrix}$$

Als A een *vierkante* matrix is, dan geeft `A^3` inplaats van `A.A.A` ook het gewenste resultaat.

Elementaire rij- en kolomoperaties. Maple kent voor de elementaire rij- en kolomoperaties de procedures `RowOperation` en `ColumnOperation`. Ze werken als volgt (`A` is een matrix):

`RowOperation`
`ColumnOperation`

- Optellen van p keer de k^{de} rij bij de i^{de} rij van A:

  `RowOperation(A, [i,k], p )`

- Vermenigvuldiging van de k^{de} rij met p:

  `RowOperation(A, k, p )`

- Verwisseling van de k^{de} rij met de i^{de} rij

  `RowOperation(A, [i,k] )`

Iets dergelijks geldt uiteraard voor kolomoperaties.

De procedure `RowOperation` kent nog een interessante extra optie, namelijk `inplace`. Met deze optie worden de wijzigingen direct in de matrix A doorgevoerd; als hij wordt weggelaten, dan verandert A *niet* door een `RowOperation`-opdracht.

`inplace`

Voorbeeldsessie

```
> with(LinearAlgebra):
> A := <<0,2,2,1,2>|<0,8,3,3,5>|<0,8,-2,2,2>|
       <2,-6,-1,-2,3>|<-2,4,9,3,1>>;
```

$$A := \begin{bmatrix} 0 & 0 & 0 & 2 & -2 \\ 2 & 8 & 8 & -6 & 4 \\ 2 & 3 & -2 & -1 & 9 \\ 1 & 3 & 2 & -2 & 3 \\ 2 & 5 & 2 & 3 & 1 \end{bmatrix}$$

```
> B := Copy(A):
```

Verwissel rij 1 met rij 4:

```
> RowOperation(B, [1,4], inplace );
```

$$\begin{bmatrix} 1 & 3 & 2 & -2 & 3 \\ 2 & 8 & 8 & -6 & 4 \\ 2 & 3 & -2 & -1 & 9 \\ 0 & 0 & 0 & 2 & -2 \\ 2 & 5 & 2 & 3 & 1 \end{bmatrix}$$

Deel rij 4 door 2:

```
> RowOperation(B, 4, 1/2, inplace );
```

$$\begin{bmatrix} 1 & 3 & 2 & -2 & 3 \\ 2 & 8 & 8 & -6 & 4 \\ 2 & 3 & -2 & -1 & 9 \\ 0 & 0 & 0 & 1 & -1 \\ 2 & 5 & 2 & 3 & 1 \end{bmatrix}$$

Tel het -2-voud van rij 1 op bij rij 5:

```
> RowOperation(B, [5,1], -2, inplace );
```

$$\begin{bmatrix} 1 & 3 & 2 & -2 & 3 \\ 2 & 8 & 8 & -6 & 4 \\ 2 & 3 & -2 & -1 & 9 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & -2 & 7 & -5 \end{bmatrix}$$

```
> A,B;
```

$$\begin{bmatrix} 0 & 0 & 0 & 2 & -2 \\ 2 & 8 & 8 & -6 & 4 \\ 2 & 3 & -2 & -1 & 9 \\ 1 & 3 & 2 & -2 & 3 \\ 2 & 5 & 2 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 2 & -2 & 3 \\ 2 & 8 & 8 & -6 & 4 \\ 2 & 3 & -2 & -1 & 9 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & -2 & 7 & -5 \end{bmatrix}$$

Toelichting

In §13.5 wordt uitgelegd waarom `B:=Copy(A)` nodig is, inplaats van `B:=A` om een nieuwe matrix B te maken, die gelijk is aan de (oorspronkelijke) matrix A.
Door steeds `inplace` te kiezen, zorgen we ervoor dat alle veranderingen direct in B worden bijgehouden. ◇

Overigens kunnen elementaire rijoperaties ook mooi worden uitgevoerd met een *tutor*: Tools→ Tutors → Linear Algebra → Gaussian Elimination.

13.5 Substitutie in matrices; vergelijken en kopiëren van matrices

Een matrix (of vector) hoeft niet alleen getallen bevatten, er mogen nog allerlei variabelen in voorkomen. Omdat deze variabelen min of meer verborgen kunnen raken, kan dit aanleiding geven tot onverwachte complicaties.[33]

Voorbeeldsessie

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

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

```
> B := A:
> B[1,1] := 5:
> A;
```

$$\begin{bmatrix} 5 & 2 \\ 3 & 4 \end{bmatrix}$$

Nu is $A_{1,1}$ ook veranderd!

```
> restart;
> A := <1,2;3,4>:
> B := LinearAlgebra:-Copy(A):
> B[1,1] := 5:
> A,B;
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 5 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> restart;
> A := <a,b;c,d>:
> a := 2:  A;
```

$$\begin{bmatrix} 2 & b \\ c & d \end{bmatrix}$$

```
> a := 9:  A;
```

$$\begin{bmatrix} 9 & b \\ c & d \end{bmatrix}$$

```
> B := subs( {b=4}, A );
```

$$B := \begin{bmatrix} 9 & 4 \\ c & d \end{bmatrix}$$

```
> A; #is ongewijzigd
```

[33] Vergelijkbaar met als `f := x -> ...` gedefinieerde functies; zie §7.3.

$$\begin{bmatrix} 9 & b \\ c & d \end{bmatrix}$$

```
> b := 11:  B; # is nu onafhankelijk van de waarde van b
```

$$\begin{bmatrix} 9 & 4 \\ c & d \end{bmatrix}$$

Toelichting

In de eerste plaats constateren we hier het merkwaardige verschijnsel, dat als we een 'nieuwe' matrix B, gelijk aan A maken, dat de veranderingen die we in B aanbrengen, óók in A doorwerken. Dat komt omdat met `B := A` in feite helemaal geen *nieuwe* matrix wordt gemaakt! Maple onthoudt alleen dat het de matrix A moet gebruiken als om B wordt gevraagd.

`Copy` Met het `Copy`-commando (uit de `LinearAlgebra`-bibliotheek) wordt wél een geheel nieuwe matrix aangemaakt, met dezelfde inhoud als A.

Als een matrix variabelen bevat, kunnen er vergelijkbare dingen gebeuren. Het veiligste is om *altijd* expliciet waarden voor de variabelen te substitueren. De oorspronkelijke matrix blijft dan in elk geval ook intact. ◇

Gelijkheid van matrices of vectoren. Een dergelijk probleem bestaat als we willen controleren of twee matrices (of vectoren) aan elkaar *gelijk* zijn. Het commando `is(A=B)` of `evalb(A=B)` geeft altijd *false* als `A` en `B` matrices of vectoren zijn. Speciaal voor vectoren en

`Equal` matrices bevat `LinearAlgebra` daarom de procedure `Equal`. Het moet worden gebruikt als

```
Equal(A,B)
```

(dus *niet* als `Equal(A=B)`).

Vergelijkingen met matrices of vectoren. Als we bijvoorbeeld α en β zouden willen oplossen uit de vergelijking $\alpha\mathbf{v_1} + \beta\mathbf{v_2} = \mathbf{a}$, met $\mathbf{v_1}$, $\mathbf{v_2}$ en $\mathbf{a}$ gegeven vectoren, dan kan dat niet (direct) met `solve`.

Voorbeeldopgave

Gegeven:

$$\mathbf{v_1} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \qquad \mathbf{v_2} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \qquad \mathbf{a} = \begin{bmatrix} 7 \\ 3 \end{bmatrix}.$$

Bepaal α en β uit de vergelijking $\alpha\mathbf{v_1} + \beta\mathbf{v_2} = \mathbf{a}$.

Voorbeeldsessie

```
> v1 := <3,1>:    v2 := <2,3>:    a := <7,3>:
> vgl := alpha*v1 + beta*v2 = a;
```

$$vgl := \begin{bmatrix} 3\,\alpha + 2\,\beta \\ \alpha + 3\,\beta \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \end{bmatrix}$$

```
> solve(vgl, {alpha,beta} );
```

Error, invalid input: solve expects its 1st argument, eqs, to be of type {'and', 'not', 'or', algebraic, relation(algebraic), ({set, list})({'and', 'not', 'or', algebraic, relation(algebraic)})}, but received (Vector(2, {(1) = 3*alpha+2*beta, (2) = alpha+3*beta})) = (Vector(2, {(1) = 7, (2) = 3}))

Op nul herleiden:

```
> vgl := alpha*v1 + beta*v2 - a;
```

$$vgl := \begin{bmatrix} 3\,\alpha + 2\,\beta - 7 \\ \alpha + 3\,\beta - 3 \end{bmatrix}$$

```
> stelsel := convert(vgl,set);
```

$$stelsel := \{\alpha + 3\,\beta - 3, 3\,\alpha + 2\,\beta - 7\}$$

```
> s := solve( stelsel, {alpha,beta} );
```

$$s := \left\{\alpha = \frac{15}{7}, \beta = \frac{2}{7}\right\}$$

Controle

```
> subs( s, alpha*v1 + beta*v2 );
```

$$\begin{bmatrix} 7 \\ 3 \end{bmatrix}$$

Alternatief:

```
> vgl := zip( '=', alpha*v1 + beta*v2, a );
```

$$vgl := \begin{bmatrix} 3\,\alpha + 2\,\beta = 7 \\ \alpha + 3\,\beta = 3 \end{bmatrix}$$

```
> solve( convert(vgl,set), {alpha,beta} );
```

$$\left\{\alpha = \frac{15}{7}, \beta = -\frac{2}{7}\right\}$$

Toelichting

Bij een vergelijking met vectoren moet u dus eerst alles op **0** (nul-vector of -matrix) herleiden (zodat de vergelijkingen geen rechterlid meer nodig hebben), en daarna naar een verzameling converteren. Dat hoeft niet per se, met een `zip`-commando kun je ook van een vergelijking van vectoren een vector van vergelijkingen maken.

Zie verder Module 14 voor een andere aanpak van dergelijke stelsels lineaire vergelijkingen. ◇

13.6 Inproduct en norm

Inproduct. Ook voor het *inproduct* $\mathbf{v} \cdot \mathbf{w}$ van twee vectoren $\mathbf{v}$ en $\mathbf{w}$ kan de gewone punt worden gebruikt.

Voorbeeldsessie

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

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

Het inproduct van twee vectoren:

```
> v.w;
```

$$11$$

Nog eens inproduct:

```
> v := <a,b>:    w := <p,q>:
> v.w;
```

$$\overline{a}\,p + \overline{b}\,q$$

Maple veronderstelt dat a, b, p en q complex zijn.
Het gebruik van `RealDomain` helpt niet:

```
> use RealDomain in v.w end use;
```

$$\overline{a}\,p + \overline{b}\,q$$

maar de `assume`-faciliteit wel:

```
> v.w assuming real;
```

$$p\,a + q\,b$$

Andere mogelijkheid: matrixproduct van een rijvector en een kolomvector

```
> v^%T.w;
```

$$p\,a + q\,b$$

geeft inderdaad het 'reële' inproduct.
Andersom:

```
> v.w^%T;
```

$$\begin{bmatrix} p\,a & a\,q \\ b\,p & q\,b \end{bmatrix}$$

```
> v*w;

Error, (in rtable/Product) invalid arguments
```

Toelichting

Als in de vectoren namen voorkomen in plaats van getallen blijkt dat Maple een `vector(n)` niet als element van $\mathbb{R}^n$, maar als element van $\mathbb{C}^n$ opvat.[34]
Als we ervan verzekerd willen zijn dat het 'reële' inproduct wordt berekend, dan moeten we expliciet vermelden dat alle gebruikte variabelen reëel verondersteld worden. Een ander mogelijkheid is om het matrixproduct $\mathbf{v}^{\mathrm{T}}\mathbf{w}$ te berekenen, waarbij de rijvector $\mathbf{v}^{\mathrm{T}}$ als $1 \times n$-matrix wordt opgevat.
De 'gewone' vermenigvuldigingsoperator "`*`" is uitsluitend bedoeld voor de 'gewone' scalaire vermenigvuldiging: twee getallen, of een getal met een vector of matrix. ◇

`Norm`
`Normalize`

Norm van een vector. Voor de (Euclidische) *norm* van vectoren kent Maple het commando `Norm`. Met `Normalize` kunnen we een vector *normeren*, dat wil zeggen dat alle coördinaten van $\mathbf{v}$ door $|\mathbf{v}|$ worden gedeeld. `Normalize` kan ook met de optie `inplace` worden gebruikt (zie blz. 172).
Aan het gebruik van `Norm` (en `Normalize`) zitten echter een paar haken en ogen:

Voorbeeldsessie

```
> with(LinearAlgebra):
> v := <-1,5,3>:    w := <a,b,c>:
> Norm(v);
```

$$5$$

```
> Norm(w);
```

$$\max\left(|a|, |b|, |c|\right)$$

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

$$\sqrt{35}$$

```
> sqrt(v.v) - %;
```

$$0$$

```
> vdak := Normalize(v,2);
```

$$vdak := \begin{bmatrix} -1/35\sqrt{35} \\ 1/7\sqrt{35} \\ \frac{3}{35}\sqrt{35} \end{bmatrix}$$

Controle:

[34] Dit wordt gedaan om ervoor te zorgen dat ook de norm van een complexe vector een positief reëel getal blijft, zie verderop.

```
> vdak.vdak;
```

$$1$$

```
> w1 := Norm(w,2);
```

$$w1 := \sqrt{(|a|)^2 + (|b|)^2 + (|c|)^2}$$

```
> w2 := Norm(w,2,conjugate=false);
```

$$w2 := \sqrt{a^2 + b^2 + c^2}$$

```
> w3 := Norm(w,2) assuming real;
```

$$w3 := \sqrt{(|a|)^2 + (|b|)^2 + (|c|)^2}$$

```
> simplify(w2-w3);
```

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

```
> simplify(w2-w3) assuming real;
```

$$0$$

Bruikbaar is ook (vooral als alle $w_i \geq 0$):

```
> Norm(w,1);
```

$$|a| + |b| + |c|$$

Toelichting

Voor de gewone norm in $\mathbb{R}^n$ hebben we `Norm(v,2)` nodig.[35] Zonder tweede argument berekent `Norm(v)` de grootste absolute waarde van de coördinaten. Ook in het complexe geval geldt $|\mathbf{v}|^2 = \mathbf{v}\cdot\mathbf{v}$. Om aan te geven dat eventueel gebruikte variabelen reëel verondersteld worden kunnen we de optie `conjugate=false` gebruiken; `assuming real` werkt hier niet.
Met `Norm(w,1)` wordt $\sum_{i=1}^{n} |w_i|$ berekend. Als alle elementen van $\mathbf{w}$ positief zijn, kun je dat gebruiken om $\mathbf{w}$ zó te schalen dat de som van de elementen gelijk aan 1 wordt. ◇

13.7 Overige berekeningen

Er is nog een aantal berekeningen met matrices en vectoren die iets minder toelichting behoeven. We sommen ze hier op (`A` is een vierkante matrix, en `v` en `w` zijn vectoren):

`Determinant`
- De *determinant* van A: `Determinant(A)`

`Trace`
- Het *spoor* van A: `Trace(A)`
  (dit is de som van de diagonaalelementen)

[35] De 2 als tweede argument slaat op het feit dat de *tweede*machtswortel van de som van de *kwadraten* genomen moet worden.

`MatrixInverse`

- De *inverse matrix* van A: `MatrixInverse(A)`
  De inverse kan ook met `A^(-1)` worden berekend.

`&x`

- Het *vectorproduct* (of *uitproduct*) $\mathbf{v} \times \mathbf{v}$ van de vectoren $\mathbf{v}$ en $\mathbf{w} \in \mathbb{R}^3$ wordt berekend als `v &x w`.

Opgave 13.1

`Diagonal`

Als A een (vierkante) matrix is, dan geeft `Diagonal(A)` (in de bibliotheek `LinearAlgebra`) de kolomvector, bestaande uit de elementen van de *hoofddiagonaal* van A, dus $a_{11}, a_{22}, \ldots, a_{nn}$).
Gebruik dit om van een willekeurige vierkante matrix (gemaakt met `RandomMatrix`) een diagonaalmatrix te maken met dezelfde hoofddiagonaal als A.

Opgave 13.2

De parametervoorstelling van een vlak is

$$\mathbf{r} = \mathbf{a} + \alpha\mathbf{u} + \beta\mathbf{v}, \quad \alpha, \beta \in \mathbb{R}\,,$$

waarbij $\mathbf{a}, \mathbf{u}$ en $\mathbf{v}$ gegeven vectoren zijn, en $\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ de plaatsvector van een punt in het vlak is.
Elimineer α en β uit de parametervoorstelling om een *vergelijking* voor het vlak te vinden, als gegeven is:

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} -3 \\ 1 \\ -2 \end{bmatrix}.$$

Opgave 13.3

Bepaal de afstand en de hoek tussen de vectoren $\mathbf{v}$ en $\mathbf{w}$ waarbij

(a) $\mathbf{v} = (-3, 1, 1, 5)$ en $\mathbf{w} = (1, 2, 0, 2)$;
(b) $\mathbf{v} = (0, 2, 1, 1, 0)$ en $\mathbf{w} = (1, -1, 0, 0, 1)$.

Opgave 13.4

Bepaal alle vectoren $\mathbf{u} = (u_1, u_2, u_3, u_4) \in \mathbb{R}^4$ die een hoek van $\frac{\pi}{3}$ of $\frac{2\pi}{3}$ maken met de vector $(0, 1, 0, 1)$ door een vergelijking te geven in de componenten van $\mathbf{u}$.

Opgave 13.5

Gebruik Maple om te bewijzen dat voor alle $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^3$ geldt:
(a) $\mathbf{u} \cdot (\mathbf{v} \times \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) \cdot \mathbf{w}$;
(b) $\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) = (\mathbf{u} \cdot \mathbf{w})\mathbf{v} - (\mathbf{u} \cdot \mathbf{v})\mathbf{w}$.
Aanwijzing: Definieer `u := <u1,u2,u3>:` enzovoort.

Opgave 13.6

(a) $A = \begin{bmatrix} 3 & 4 \\ 5 & 2 \end{bmatrix}$. Bereken de matrix $A^2 - 5A - 14I_2$.

(b) $B = \begin{bmatrix} 2 & 1 & 0 \\ 3 & 2 & 0 \\ 0 & 0 & \alpha \end{bmatrix}$. Bepaal α zo dat $(B - 4I_3)(B^2 - 4B + I_3)$ de nulmatrix is.

Opgave 13.7

Gegeven is dat A een $n \times n$-matrix is. Bereken de determinant van de matrix $A - \lambda I_n$ en verifieer dat deze een polynoom van de graad n in λ is.
(Maak voor verschillende waarden van n een aantal verschillende random-matrices A.)

Opgave 13.8

$C[0, 2\pi]$ is de verzameling van continue functies op het interval $[0, 2\pi]$. In $C[0, 2\pi]$ is een inwendig product gedefinieerd door

$$f \cdot g = \frac{1}{\pi} \int_0^{2\pi} f(x)g(x)\,dx.$$

Verder is gegeven de verzameling S van functies in $C[0, 2\pi]$ door

$$S = \left\{ \frac{1}{\sqrt{2}},\ \sin(x),\ \cos(x),\ \sin(2x),\ \cos(2x) \right\}.$$

Laat zien dat voor alle functies in S geldt dat $f \cdot g = 0$ als $f \neq g$ en $f \cdot f = 1$. Doe dit door een matrix te construeren met het (i, j)-de element het inproduct van de i^{de} en j^{de} functie uit S.