ALGÈBRE LINÉAIRE

Bernard Dupont

Bernard.Dupont@univ-lille1.fr

Pour des raisons intrinséques - qui tiennent au caractère "mécanique" de ses calculs, l'algèbre linéaire est sans aucun doute le domaine mathématique qui se prête le mieux au calcul formel. L'informatique construit et manipule facilement des vecteurs et des matrices. Elle exécute des opérations à une vitesse bluffante. Des commandes basiques sont disponibles dans le noyau de Maple mais ce chapitre va aussi explorer les ressources très solides du paquetage dédié **LinearAlgebra**. Il convient donc de charger celui-ci en premier lieu :

```
> restart;
  with(LinearAlgebra);
```

```
[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm,
    BilinearForm, CharacteristicMatrix, CharacteristicPolynomial, Column,
    ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix,
    ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation,
    CrossProduct, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix,
    Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors,
    Equal, ForwardSubstitute, FrobeniusForm, GaussianElimination, GenerateEquations,
    GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix,
    GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm,
    HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite,
    IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, KroneckerProduct,
    LA_Main, LUDecomposition, LeastSquares, LinearSolve, LyapunovSolve, Map, Map2,
    MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply,
    MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply,
    MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize,
    NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, QRDecomposition,
    RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm,
    Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply,
    ScalarVector, SchurForm, SingularValues, SmithForm, StronglyConnectedBlocks,
    SubMatrix, SubVector, SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace,
    Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle,
    VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip]
```

Les fonctions de cette bibliothèque sont si nombreuses que seules celles qui répondent aux besoins courants des économistes seront évoquées.

Les sections 1 et 2 traitent respectivement des vecteurs et des matrices au triple point de vue de leur construction, de leur manipulation et des opérations. La section 3 montre comment s'informer sur les caractéristiques d'une matrice, carrée ou non. La section 4 est consacrée aux tests sur les matrices car Maple a développé des programmes puissants capables de comparer l'égalité ou la similarité de deux matrices ou encore de vérifier si une matrice est définie. La section 5 montre comment passer d'un

système d'équations linéaires à sa forme matricielle puis le résoudre.

Vecteurs

Conséquence de la succession des versions dans le temps et de la difficulté à rompre radicalement avec le passé, Maple propose plusieurs façons de construire des vecteurs (et des matrices). La première sous-section retient la dernière en date, qui est la plus performante. La seconde sous-section montre comment manipuler les vecteurs, en particulier accéder à ses composantes et les modifier. La sous-section 3 donne les règles opératoires sur les vecteurs.

Création de vecteurs : `Vector`

L'instruction longue à écrire mais non ambigue **`Vector([a,b,...,z])`** renvoie un vecteur-colonne dont les coordonnées sont *a*, *b*, ..., *z*. Notez le "V majuscule" de **`Vector`**. Le fait d'écrire **`vector`** au lieu de **`Vector`** n'entraîne pas de message d'avertissement car Maple accepte toujours cette écriture d'une version antérieure du logiciel; mais la structure informatique du vecteur n'est pas la même. Un vecteur-ligne s'obtient avec l'option **`orientation=row`** ou plus simplement en tapant **`Vector[row]([coordonnées])`**.

```
> v1:=Vector([a,b,c]);whattype(v1);
  v2:=Vector([1,alpha,2,beta]);whattype(v2);
  v3:=Vector([a,b,c],orientation=row);whattype(v3);
  v4:=Vector[row]([1,alpha,2,beta]);whattype(v4);
```

$$v1 := \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$Vector_{column}$$

$$v2 := \begin{bmatrix} 1 \\ \alpha \\ 2 \\ \beta \end{bmatrix}$$

$$Vector_{column}$$

$$v3 := \begin{bmatrix} a & b & c \end{bmatrix}$$

$$Vector_{row}$$

$$v4 := \begin{bmatrix} 1 & \alpha & 2 & \beta \end{bmatrix}$$

$$Vector_{row}$$

La seconde possibilité de création d'un vecteur est d'utiliser le racourci **`<a,b,...,z>`** pour obtenir un vecteur-colonne et le racourci **`<a|b|...|z>`** pour obtenir un vecteur-ligne :

```
> v5:=<a,b,c>;whattype(v5);
  v6:=<1,alpha,2,beta>;whattype(v6);
```

```
v7:=<a|b|c>;whattype(v7);
v8:=<1|alpha|2|beta>;whattype(v8);
```

$$v5 := \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$Vector_{column}$$

$$v6 := \begin{bmatrix} 1 \\ \alpha \\ 2 \\ \beta \end{bmatrix}$$

$$Vector_{column}$$

$$v7 := \begin{bmatrix} a & b & c \end{bmatrix}$$

$$Vector_{row}$$

$$v8 := \begin{bmatrix} 1 & \alpha & 2 & \beta \end{bmatrix}$$

$$Vector_{row}$$

Malgrè les apparences, ces deux méthodes de génération de vecteurs ne sont pas équivalentes. La seconde sera utilisée si on connaît à la fois la dimension et toutes les composantes. La première ne nécessite pas forcément la connaissance de toutes les composantes. Ainsi, **Vector** peut avoir pour seul argument une dimension **d**, ce qui crée un vecteur à *d* composantes nulles. Il peut aussi avoir pour premier argument une dimension **d** et pour second argument un nombre **n**, ce qui crée un vecteur avec *d* composantes égales à *n*. Il peut encore avoir pour premier argument une dimension **d** et pour second argument une liste de **p** nombres ($p \leq n$), ce qui crée un vecteur à *d* composantes dont les *p* premières sont les éléments de la liste.

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

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

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

$$\begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} \qquad \textbf{(1.1.1)}$$

Certains vecteurs particuliers, tels que le vecteur nul, un vecteur unitaire ou un vecteur aléatoire, peuvent être créés par des commandes spécifiques du paquetage **LinearAlgebra**.

Un vecteur à *n* composantes nulles s'écrit en colonne avec **ZeroVector(n)** et en ligne avec **ZeroVector[row](n)** :

```
> ZeroVector(5);
  ZeroVector[row](5);
```

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

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

La commande **UnitVector(i,d)** retourne un vecteur d'un espace de dimension **d** dont la **i**-ème composante est égale à 1 et les autres égales à 0.

```
> UnitVector(3,5);#vecteur unitaire colonne
  UnitVector[row](3,5);#vecteur unitaire ligne
```

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

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

La commande **RandomVector(d)** retourne un vecteur de dimension **d** dont les composantes sont des entiers compris entre -99 et +99. L'option **generator=a..b** où *a* et *b* sont des réels permet d'obtenir des nombres aléatoires réels dans l'intervalle [*a*,*b*].

```
> RandomVector(5);#vecteur-colonne à 5 composantes aléatoires
  prises entre -99 et +99
  RandomVector[row](5,generator=0..1.);#vecteur-ligne à 5
  composantes aléatoires prises entre 0 et 1
```

$$\begin{bmatrix} 29 \\ 44 \\ 92 \\ -31 \\ 67 \end{bmatrix}$$

$$[0.188381974003326236, 0.547220597255961728, 0.308167048525366140, 0.221034047248655829, 0.968867770682095398]$$

Manipulations sur les vecteurs

Un vecteur **v** étant créé, sa transposée est donnée par la commande **Transpose(v)** du paquetage **LinearAlgebra**.

```
> v1:=<C,I,Y>;Transpose(v1);#transposée d'un vecteur-colonne
  v2:=<P1|P2|P3>;Transpose(v2);#transposée d'un vecteur-ligne
```

$$v1 := \begin{bmatrix} C \\ \mathrm{I} \\ Y \end{bmatrix}$$

$$\begin{bmatrix} C & \mathrm{I} & Y \end{bmatrix}$$

$$v2 := \begin{bmatrix} P1 & P2 & P3 \end{bmatrix}$$

$$\begin{bmatrix} P1 \\ P2 \\ P3 \end{bmatrix} \qquad \textbf{(1.2.1)}$$

Pour extraire une composante d'un vecteur, on procédera comme s'il s'agit d'une liste. Ainsi, si **v** est un vecteur à n composantes, la sélection de la p-ième composante ($p \leq n$) se fait par **v[p]** qui renvoie en écho l'élément concerné. La sélection des composantes p, $p+1$, ..., q ($p \leq q \leq n$) se fait par **v[p..q]** qui renvoie un vecteur à $q-p+1$ composantes.

```
> v:=<a,1,b,2*x^6,exp(Pi*t),0.02>;
  v[5];
  v[2..4];
```

$$v := \begin{bmatrix} a \\ 1 \\ b \\ 2\,x^{6} \\ \mathrm{e}^{\pi t} \\ 0.02 \end{bmatrix}$$

$$\mathrm{e}^{\pi t}$$

$$\begin{bmatrix} 1 \\ b \\ 2\,x^{6} \end{bmatrix} \qquad \textbf{(1.2.2)}$$

La modification des composantes d'un vecteur devient alors possible par réassignation d'une ou de plusieurs composantes :

```
> v[5]:=log(Pi*t);#modification de la 5-ième composante de v
  v;#v est modifié
  v[2..4]:=<0,0,0>;#modification des composantes 2,3 et 4.
  v;#v est modifié
```

$$v_5 := \ln(\pi\, t)$$

$$\begin{bmatrix} a \\ 1 \\ b \\ 2\,x^6 \\ \ln(\pi\, t) \\ 0.02 \end{bmatrix}$$

$$v_{2..4} := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} a \\ 0 \\ 0 \\ 0 \\ \ln(\pi\, t) \\ 0.02 \end{bmatrix} \quad \textbf{(1.2.3)}$$

Opérations sur les vecteurs

L'addition de vecteurs se fait en utilisant la touche **+**, la multiplication par un scalaire par la touche ***** ou par la commande dédiée **VectorScalarMultiply(v,t)** du paquetage **LinearAlgebra**, **v** étant un vecteur et **t** un scalaire.

```
> v1:=<a,b,c>:v2:=<1,2,3>:v3:=<alpha,beta,gamma>:#vecteurs-colonnes
  v1+v2;v2+v3;v1+v2+v3;#addition de vecteurs
  2*v1;5*v2;2*v1+5*v2;lambda*v3;#multiplication par un scalaire
```

$$\begin{bmatrix} a+1 \\ b+2 \\ c+3 \end{bmatrix}$$

$$\begin{bmatrix} 1+\alpha \\ 2+\beta \\ 3+\gamma \end{bmatrix}$$

$$\begin{bmatrix} a+1+\alpha \\ b+2+\beta \\ c+3+\gamma \end{bmatrix}$$

$$\begin{bmatrix} 2\,a \\ 2\,b \\ 2\,c \end{bmatrix}$$

$$\begin{bmatrix} 5 \\ 10 \\ 15 \end{bmatrix}$$

$$\begin{bmatrix} 2\,a+5 \\ 2\,b+10 \\ 2\,c+15 \end{bmatrix}$$

$$\begin{bmatrix} \lambda\,\alpha \\ \lambda\,\beta \\ \lambda\,\gamma \end{bmatrix}$$

```
> v1:=<a|b|c>:v2:=<1|2|3>:v3:=<alpha|beta|gamma>:#vecteurs-
  lignes
  v1+v2;v2+v3;v1+v2+v3;#addition de vecteurs
  2*v1;5*v2;2*v1+5*v2;lambda*v3;#multiplication par un
  scalaire
```

$$\begin{bmatrix} a+1 & b+2 & c+3 \end{bmatrix}$$

$$\begin{bmatrix} 1+\alpha & 2+\beta & 3+\gamma \end{bmatrix}$$

$$\begin{bmatrix} a+1+\alpha & b+2+\beta & c+3+\gamma \end{bmatrix}$$

$$\begin{bmatrix} 2\,a & 2\,b & 2\,c \end{bmatrix}$$

$$\begin{bmatrix} 5 & 10 & 15 \end{bmatrix}$$

$$\begin{bmatrix} 2\,a+5 & 2\,b+10 & 2\,c+15 \end{bmatrix}$$

$$\begin{bmatrix} \lambda\,\alpha & \lambda\,\beta & \lambda\,\gamma \end{bmatrix}$$

```
> VectorScalarMultiply(<a,b,c>,1/2);#multiplication d'un
  vecteur-colonne par un scalaire
  VectorScalarMultiply(<a|b|c>,1/2);#multiplication d'un
  vecteur-ligne par un scalaire
```

$$\begin{bmatrix} \frac{1}{2}\,a \\ \frac{1}{2}\,b \\ \frac{1}{2}\,c \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{2}\,a & \frac{1}{2}\,b & \frac{1}{2}\,c \end{bmatrix}$$

Le produit scalaire des vecteurs de même dimension **v1** et **v2** est donné par la commande **DotProduct(v1,v2,conjugate=false)** :

```
> v1:=<x1,x2,x3>;v2:=<y1,y2,y3>;DotProduct(v1,v2,conjugate=
  false);
```

$$v1 := \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix}$$

$$v2 := \begin{bmatrix} y1 \\ y2 \\ y3 \end{bmatrix}$$

$$x1\,y1 + x2\,y2 + x3\,y3$$

Matrices

Les matrices s'approchent sur le même plan d'étude que les vecteurs : création; manipulation; opérations.

Création de matrices : Matrix

Soit M une matrice à m lignes et n colonnes. On note aij l'élément situé sur la i-ème ligne et la j-ème colonne. Comme pour les vecteurs, il y a deux manières de la créer en Maple.

On peut utiliser la commande **Matrix(1..m,1..n,[[a11,a12,..,a1n],[a21,a22,..,a2n],..,[am1,..,amn])**, par exemple :

```
> M1:=Matrix(1..3,1..4,[[a,b,c,d],[1,2,3,4],[alpha,beta,
  gamma,delta]]);whattype(M1);
```

$$M1 := \begin{bmatrix} a & b & c & d \\ 1 & 2 & 3 & 4 \\ \alpha & \beta & \gamma & \delta \end{bmatrix}$$

Matrix

ou, plus simplement en précisant le nombre de lignes et le nombre de colonnes par **Matrix(m,n,[[a11,a12,..,a1n],[a21,a22,..,a2n],..,[am1,..,amn])**.

```
> M2:=Matrix(3,4,[[a,b,c,d],[1,2,3,4],[alpha,beta,gamma,
  delta]]);whattype(M2);
```

$$M2 := \begin{bmatrix} a & b & c & d \\ 1 & 2 & 3 & 4 \\ \alpha & \beta & \gamma & \delta \end{bmatrix}$$

Matrix

Quand on connaît toutes les composantes d'une matrice, il est plus simple d'avoir recours au raccourci **<v1|v2|..|vn>** si les **vi** sont des vecteurs colonnes ou **<v1,v2,..,vm>** si les **vj** sont des vecteurs-lignes.

```
> v1:=<a,1,alpha>;v2:=<b,2,beta>;v3:=<c,3,gamma>;v4:=<d,4,
  gamma>;#création de 4 vecteurs-colonnes
  M3:=<v1|v2|v3|v4>;#concaténation de vecteurs-colonnes
```

$$v1 := \begin{bmatrix} a \\ 1 \\ \alpha \end{bmatrix}$$

$$v2 := \begin{bmatrix} b \\ 2 \\ \beta \end{bmatrix}$$

$$v3 := \begin{bmatrix} c \\ 3 \\ \gamma \end{bmatrix}$$

$$v4 := \begin{bmatrix} d \\ 4 \\ \gamma \end{bmatrix}$$

$$M3 := \begin{bmatrix} a & b & c & d \\ 1 & 2 & 3 & 4 \\ \alpha & \beta & \gamma & \gamma \end{bmatrix}$$

```
> v5:=<a|b|c|d>;v6:=<1|2|3|4>;v7:=<alpha|beta|gamma|delta>;
  #création de 3 vecteurs-lignes
  M4:=<v5,v6,v7>;#concaténation de vecteurs-lignes
```

$$v5 := \begin{bmatrix} a & b & c & d \end{bmatrix}$$

$$v6 := \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

$$v7 := \begin{bmatrix} \alpha & \beta & \gamma & \delta \end{bmatrix}$$

$$M4 := \begin{bmatrix} a & b & c & d \\ 1 & 2 & 3 & 4 \\ \alpha & \beta & \gamma & \delta \end{bmatrix}$$

Les deux méthodes ne sont équivalentes que si toutes les composantes sont explicitées. Si ce n'est pas le cas ou si ce n'est pas nécessaire, l'instruction **`Matrix`** est la seule option. Ainsi **`Matrix(m,n)`** génère une matrice à *m* lignes et *n* colonnes qu'elle remplit de 0 par défaut; **`Matrix(m,n,C)`** génère une matrice à *m* lignes et *n* colonnes dont toutes les composantes valent *C*; **`Matrix(m,n,[[liste1],[liste2],...,[listek]])`** où $k \leq m$ et où le nombre d'éléments de chaque liste est inférieur ou égal à *n*, donne une matrice à *m* lignes et *n* colonnes dont les premières composantes pour chaque ligne sont les éléments des listes correspondantes et les composantes restantes des 0.

```
> Matrix(3,3);
  Matrix(3,3,exp(2));
  Matrix(3,3,[[exp(1),exp(2)],[exp(3)],[exp(1.25)]]);
```

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

$$\begin{bmatrix} e^2 & e^2 & e^2 \\ e^2 & e^2 & e^2 \\ e^2 & e^2 & e^2 \end{bmatrix}$$

$$\begin{bmatrix} e & e^2 & 0 \\ e^3 & 0 & 0 \\ 3.490342957 & 0 & 0 \end{bmatrix} \quad \textbf{(2.1.1)}$$

Certaines matrices particulières s'écrivent à l'aide de commandes spécifiques disponibles dans le paquetage **`LinearAlgebra`**.

Matrice nulle. La commande **`ZeroMatrix(m, n)`** retourne une matrice de format $m \times n$ dont toutes les composantes sont nulles.

```
> ZeroMatrix(5,4);
```

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

Matrice identité. La commande **`IdentityMatrix(m)`** construit une matrice carrée de format *m* dont les composantes de la diagonale principale valent 1 et toutes les autres composantes sont nulles.

```
> IdentityMatrix(3);
```

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

Matrice aléatoire. La commande **`RandomMatrix(m,n)`** retourne une matrice de format $m \times n$ dont les composantes sont des entiers aléatoirement choisis entre -99 et $+99$. L'option **`generator=a..b`** où a et b sont des réels permet d'obtenir des nombres aléatoires réels dans l'intervalle $[a, b]$.

```
> RandomMatrix(3,4);
  RandomMatrix(3,4,generator=0..1.);
```

$$\begin{bmatrix} -22 & -38 & 33 & 57 \\ 45 & -18 & -98 & 27 \\ -81 & 87 & -77 & -93 \end{bmatrix}$$

```
[[0.39873852400288079, 0.36129400766737241, 0.87843064933637094,
  0.00478348450184817866],
 [0.68135954200279081, 0.79792860922002551, 0.63976336115753141,
  0.29702944414232924],
 [0.21192433635885444, 0.50366268050477425, 0.11246451672714586,
  0.79810585677832713]]
```

Matrice diagonale. La commande **`DiagonalMatrix(v)`** où **`v`** est un vecteur de dimension m ou une liste de m scalaires retourne une matrice carrée de format m dont les composantes de la diagonale principale sont les composantes de **`v`** et toutes les autres composantes sont nulles.

```
> DiagonalMatrix([1,2,3]);DiagonalMatrix(<alpha,beta,gamma>);
  DiagonalMatrix([%%,%]);
```

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

$$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 & 0 & \gamma \end{bmatrix}$$

Manipulations sur les matrices

Cette section expose les principales commandes permettant de manipuler les éléments d'une matrice : extraction d'un élément, d'une colonne, d'une ligne, d'une sous-matrice, "augmentation" d'une matrice.

Extraction d'un élément

Soit M une matrice de format $m \times n$. L'extraction de l'élément situé à la ligne i ($1 \leq i \leq m$) et la colonne j ($1 \leq j \leq n$) se fait par l'instruction **`M[i,j]`**.

```
> M:=<<-2,3,10>|<2,7,-4>|<3,-8,-3>>;M[1,1];M[1,2];M[2,3];
```

$$M := \begin{bmatrix} -2 & 2 & 3 \\ 3 & 7 & -8 \\ 10 & -4 & -3 \end{bmatrix}$$

$$-2$$

$$2$$

$$-8$$

Extraction d'une colonne et d'une ligne

Soit M une matrice de format $m \times n$. L'extraction de la j-ième colonne se fait par la commande **`Column(M,j)`** du paquetage **`LinearAlgebra`**.

```
> Column(M,2);
```

$$\begin{bmatrix} 2 \\ 7 \\ -4 \end{bmatrix} \qquad \textbf{(2.2.2.1)}$$

Dans le même ordre d'idée, l'extraction de la i-ème ligne se fait par la commaande **`Row(M,i)`** du paquetage **`LinearAlgebra`**.

```
> Row(M,3);
```

$$\begin{bmatrix} 10 & -4 & -3 \end{bmatrix} \qquad \textbf{(2.2.2.2)}$$

Extraction d'une sous-matrice

On appelle bloc de la matrice M de format $p \times q$ une sous-matrice formée d'éléments contigus.

Il y a deux manières d'extraire un bloc de format $p \times q$ ($1 \leq p \leq m$ et $1 \leq q \leq n$).

La première s'inspire de la syntaxe d'extraction d'un élément et consiste à expliciter les lignes et les colonnes retenues au moyen d'intervalles de valeurs prises par les indices des lignes et des colonnes :

```
> M[2..3,2..3];
```

$$\begin{bmatrix} 7 & -8 \\ -4 & -3 \end{bmatrix}$$

La seconde utilise la commande dédiée du paquetage **LinearAlgebra**, soit **SubMatrix(M,[i1..i2],[j1..j2])** ou plus simplement **SubMatrix(M,i1..i2,j1..j2)** :

```
> SubMatrix(M,[2..3],[2..3]);SubMatrix(M,2..3,2..3);
```

$$\begin{bmatrix} 7 & -8 \\ -4 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 7 & -8 \\ -4 & -3 \end{bmatrix}$$

Il est alors simple d'extraire des matrices uni-ligne ou uni-colonne, le résultat étant reconnu par Maple comme un vecteur :

```
> M1:=M[1,1..3];M2:=M[1..2,3];whattype(M1);whattype(M2);
```

$$M1 := \begin{bmatrix} -2 & 2 & 3 \end{bmatrix}$$

$$M2 := \begin{bmatrix} 3 \\ -8 \end{bmatrix}$$

$$Vector_{row}$$

$$Vector_{column}$$

Si on veut que le résultat soit une matrice et non un vecteur, il faut le signaler en plaçant l'indice entre crochets :

```
> M1:=M[[1],1..3];M2:=M[1..2,[3]];whattype(M1);whattype(M2)
  ;
```

$$M1 := \begin{bmatrix} -2 & 2 & 3 \end{bmatrix}$$

$$M2 := \begin{bmatrix} 3 \\ -8 \end{bmatrix}$$

$$Matrix$$

$$Matrix$$

Pour former une sous-matrice formée d'éléments non contigus au départ, il faut donner dans l'ordre la liste des lignes retenues puis la liste des colonnes retenues :

```
> M;M[[1,3],[1,3]];SubMatrix(M,[1,3],[1,3]);
```

$$\begin{bmatrix} -2 & 2 & 3 \\ 3 & 7 & -8 \\ 10 & -4 & -3 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 3 \\ 10 & -3 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 3 \\ 10 & -3 \end{bmatrix}$$

En "mélangeant" les deux principes précédents, on peut intervertir deux lignes entre elles ou deux colonnes entre elles :

```
> M[[3,2,1],1..3]; #interversion entre les lignes 1 et 3
  M[1..3,[1,3,2]]; #interversion des colonnes 2 et 4
```

$$\begin{bmatrix} 10 & -4 & -3 \\ 3 & 7 & -8 \\ -2 & 2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 3 & 2 \\ 3 & -8 & 7 \\ 10 & -3 & -4 \end{bmatrix}$$

Matrice augmentée

Soit A, une matrice de format $m \times n$, B, une matrice de format $m \times p$, C, une matrice de format $q \times n$, et M, une matrice de format $q \times p$. On peut former par concaténation de A et B une matrice de format $m \times (n+p)$, par concaténation de A et C une matrice de format $(m+q) \times n$, par concaténation de A, B, C et M une matrice de format $(m+q) \times (n+p)$, etc.

```
> A:=<<a|b>,<c|d>>;B:=<<1|2|3>,<4|5|6>>;C:=<<alpha|beta>,
  <gamma|delta>,<rho|omicron>>;M:=<<sqrt(2)|exp(2)|ln(3)>,
  <cos(Pi)|sin(Pi)|tan(4)>,<x^2|x^3|x^4>>;#création des
  matrices
  mat1:=<A|B>;#concaténation horizontale
  mat2:=<A,C>;#concaténation verticale
  mat3:=<<A|B>,<C|M>>;#concaténation horizontale et
  verticale
```

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

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

$$C := \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \\ \rho & o \end{bmatrix}$$

$$M := \begin{bmatrix} \sqrt{2} & e^2 & \ln(3) \\ -1 & 0 & \tan(4) \\ x^2 & x^3 & x^4 \end{bmatrix}$$

$$mat1 := \begin{bmatrix} a & b & 1 & 2 & 3 \\ c & d & 4 & 5 & 6 \end{bmatrix}$$

$$mat2 := \begin{bmatrix} a & b \\ c & d \\ \alpha & \beta \\ \gamma & \delta \\ \rho & o \end{bmatrix}$$

$$mat3 := \begin{bmatrix} a & b & 1 & 2 & 3 \\ c & d & 4 & 5 & 6 \\ \alpha & \beta & \sqrt{2} & e^2 & \ln(3) \\ \gamma & \delta & -1 & 0 & \tan(4) \\ \rho & o & x^2 & x^3 & x^4 \end{bmatrix}$$

Opérations sur les matrices

On passe en revue l'addition des matrices, la multiplication par un scalaire, la "pseudo" addition d'une matrice et d'un scalaire, le produit matriciel et son corollaire la puissance d'une matrice.

Addition des matrices

Soit A et B, deux matrices de même format. L'addition $A + B$ se fait simplement en Maple en utilisant le signe **+** ou en invoquant la commande dédiée **MatrixAdd** du paquetage **LinearAlgebra** :

```
> restart;
  with(LinearAlgebra):
  A:=DiagonalMatrix([1,2,3]);B:=<<a,b,c>|<C,I,G>|<alpha,
  beta,gamma>>;#création de deux matrices 3*3
  'A+B'=A+B;#les apostrophes empêchent l'évaluation
  'A+B'=MatrixAdd(A,B);
```

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

$$B := \begin{bmatrix} a & C & \alpha \\ b & \mathrm{I} & \beta \\ c & G & \gamma \end{bmatrix}$$

$$A+B = \begin{bmatrix} 1+a & C & \alpha \\ b & 2+\mathrm{I} & \beta \\ c & G & 3+\gamma \end{bmatrix}$$

$$A+B = \begin{bmatrix} 1+a & C & \alpha \\ b & 2+\mathrm{I} & \beta \\ c & G & 3+\gamma \end{bmatrix}$$

Multiplication par un scalaire

Soit A une matrice et λ un scalaire (réel ou complexe). La multiplication de A par λ se fait par **`*`** ou par la commande dédiée **`ScalarMultiply(A,lambda)`** du paquetage **`LinearAlgebra`**. Si λ est un paramètre numérique, alors les deux méthodes sont équivalentes. Si λ est un paramètre symbolique, il est prudent de passer par la deuxième méthode (il arrive que la première méthode ne fasse pas entrer le paramètre dans la matrice).

```
> A:=<<a|b>,<c|d>>;
  '2*A'=2*A;
  'lambda*A'=lambda*A;
  'lambda*A'=ScalarMultiply(A,lambda);
```

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

$$2\,A = \begin{bmatrix} 2\,a & 2\,b \\ 2\,c & 2\,d \end{bmatrix}$$

$$\lambda\,A = \begin{bmatrix} \lambda\,a & \lambda\,b \\ \lambda\,c & \lambda\,d \end{bmatrix}$$

$$\lambda\,A = \begin{bmatrix} \lambda\,a & \lambda\,b \\ \lambda\,c & \lambda\,d \end{bmatrix}$$

Addition d'une matrice et d'un scalaire

Evidemment, le titre est trompeur d'un point de vue mathématique. Pourtant Maple accepte la commande **`A+1`** ou **`A+lambda`** à condition de préciser le statut de **`lambda`** avec **`assuming`**.

```
> A+1;
```

$$\begin{bmatrix} 1+a & b \\ c & d+1 \end{bmatrix}$$

```
> A+lambda assuming lambda::scalar;simplify(%) assuming
  lambda::scalar;
```

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

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

Il est sous-entendu que le scalaire est multiplié par la matrice identité de même format que A. Maple calcule en fait $A + Id$ dans le premier exemple et $A + \lambda\, Id$ dans le second.

Multiplication matricielle

Soit A une matrice de format $m \times n$ et B une matrice de format $n \times p$. Le produit de A par B peut être demandé par la commande **A.B** ou par la commande dédiée **MatrixMatrixMultiply(A,B)** du paquetage **LinearAlgebra**. La multiplication matricielle ne doit jamais faire intervenir la touche produit du pavé numérique ***** qui est réservée pour les produits commutatifs et on sait que le produit matriciel n'a pas cette propriété.

```
> A:=Matrix(1..2,1..3,[[1.2,2.3,3.4],[4.5,5.6,6.7]]);B:=
  Matrix(1..3,1..4,[[a,b,c,d],[e,f,g,h],[i,j,k,l]]);
  A.B;
  MatrixMatrixMultiply(A,B);
```

$$A := \begin{bmatrix} 1.2 & 2.3 & 3.4 \\ 4.5 & 5.6 & 6.7 \end{bmatrix}$$

$$B := \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix}$$

$$[[1.2\,a + 2.3\,e + 3.4\,i,\ 1.2\,b + 2.3\,f + 3.4\,j,\ 1.2\,c + 2.3\,g + 3.4\,k,\ 1.2\,d + 2.3\,h + 3.4\,l],$$
$$[4.5\,a + 5.6\,e + 6.7\,i,\ 4.5\,b + 5.6\,f + 6.7\,j,\ 4.5\,c + 5.6\,g + 6.7\,k,\ 4.5\,d + 5.6\,h + 6.7\,l]]$$

$$[[1.2\,a + 2.3\,e + 3.4\,i,\ 1.2\,b + 2.3\,f + 3.4\,j,\ 1.2\,c + 2.3\,g + 3.4\,k,\ 1.2\,d + 2.3\,h + 3.4\,l],$$
$$[4.5\,a + 5.6\,e + 6.7\,i,\ 4.5\,b + 5.6\,f + 6.7\,j,\ 4.5\,c + 5.6\,g + 6.7\,k,\ 4.5\,d + 5.6\,h + 6.7\,l]]$$

Un message d'erreur sanctionne une tentative qui ne respecte pas la cohérence lignes-colonnes :

```
> B.A;
Error, (in LinearAlgebra:-MatrixMatrixMultiply) first
```

matrix column dimension (4) <> second matrix row dimension (2)

On vérifie aisément que la multiplication matricielle n'est pas commutative :

```
> C:=<<a,b>|<c,d>>;E:=<<alpha,beta>|<gamma,delta>>;C.E;E.C;
  Equal(C.E,E.C);#test d'égalité de matrices (voir plus
  bas)
```

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

$$E := \begin{bmatrix} \alpha & \gamma \\ \beta & \delta \end{bmatrix}$$

$$\begin{bmatrix} a\,\alpha + c\,\beta & a\,\gamma + c\,\delta \\ b\,\alpha + d\,\beta & b\,\gamma + d\,\delta \end{bmatrix}$$

$$\begin{bmatrix} a\,\alpha + b\,\gamma & \alpha\,c + \gamma\,d \\ \beta\,a + \delta\,b & c\,\beta + d\,\delta \end{bmatrix}$$

false

Produit d'une matrice et d'un vecteur

A condition de respecter les règles de cohérence lignes-colonnes, la multiplication de la matrice *A* par le vecteur *V* est marquée par un point (et non par la touche de multiplication du pavé numérique) :

```
> A:=Matrix(1..2,1..3,[[1.2,2.3,3.4],[4.5,5.6,6.7]]);V:=<x,
  y,z>;A.V;
```

$$A := \begin{bmatrix} 1.2 & 2.3 & 3.4 \\ 4.5 & 5.6 & 6.7 \end{bmatrix}$$

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

$$\begin{bmatrix} 1.2\,x + 2.3\,y + 3.4\,z \\ 4.5\,x + 5.6\,y + 6.7\,z \end{bmatrix}$$

On peut alors écrire aisément la forme quadratique associée à une matrice :

```
> Q:=<<a11,a21,a31>|<a12,a22,a32>|<a13,a23,a33>>;
  Transpose(V).Q.V;expand(%);fq:=collect(%,[x,y,z]);
```

$$Q := \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}$$

$$x\,(x\,a11 + y\,a21 + z\,a31) + y\,(x\,a12 + y\,a22 + z\,a32) + z\,(x\,a13 + y\,a23 + z\,a33)$$

$$x^2\,a11 + x\,y\,a21 + x\,z\,a31 + y\,x\,a12 + y^2\,a22 + y\,z\,a32 + z\,x\,a13 + z\,y\,a23 + z^2\,a33$$

$$fq := x^2\,a11 + ((a21 + a12)\,y + (a31 + a13)\,z)\,x + y^2\,a22 + (a32 + a23)\,z\,y + z^2\,a33$$

Puissance d'une matrice

Maple calcule les puissances entières positives d'une matrice carrée. Il faut utiliser le symbole d'exponentiation **^.**

```
> A:=<<a|b>,<c|d>>;'A^1'=A^1;'A^2'=A^2;'A^3'=A^3;
```

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

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

$$A^2 = \begin{bmatrix} a^2 + b\,c & a\,b + b\,d \\ c\,a + d\,c & b\,c + d^2 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} (a^2 + b\,c)\,a + (a\,b + b\,d)\,c & (a^2 + b\,c)\,b + (a\,b + b\,d)\,d \\ (c\,a + d\,c)\,a + (b\,c + d^2)\,c & (c\,a + d\,c)\,b + (b\,c + d^2)\,d \end{bmatrix}$$

Tant que la matrice est non singulière, Maple calcule les puissances entières négatives d'une matrice carrée :

```
> 'A^(-1)'=A^(-1);'A^(-2)'=A^(-2);
```

$$\frac{1}{A} = \begin{bmatrix} \frac{d}{a\,d - b\,c} & -\frac{b}{a\,d - b\,c} \\ -\frac{c}{a\,d - b\,c} & \frac{a}{a\,d - b\,c} \end{bmatrix}$$

$$\frac{1}{A^2} = \begin{bmatrix} \frac{b\,c + d^2}{a^2\,d^2 + b^2\,c^2 - 2\,a\,b\,d\,c} & -\frac{b\,(a + d)}{a^2\,d^2 + b^2\,c^2 - 2\,a\,b\,d\,c} \\ -\frac{c\,(a + d)}{a^2\,d^2 + b^2\,c^2 - 2\,a\,b\,d\,c} & \frac{a^2 + b\,c}{a^2\,d^2 + b^2\,c^2 - 2\,a\,b\,d\,c} \end{bmatrix}$$

Ainsi, on dispose d'un moyen commode pour obtenir l'inverse d'une matrice :

```
> A.A^(-1);simplify(%);
```

$$\begin{bmatrix} \frac{a\,d}{a\,d - b\,c} - \frac{b\,c}{a\,d - b\,c} & 0 \\ 0 & \frac{a\,d}{a\,d - b\,c} - \frac{b\,c}{a\,d - b\,c} \end{bmatrix}$$

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

Caractéristiques d'une matrice

Le paquetage **LinearAlgebra** offre un certain nombre de commandes relatives aux caractéristiques d'une matrice : trace, déterminant, inverse, noyau, rang, polynôme caractéristique, valeurs propres, vecteurs propres, etc.

Trace; déterminant; inverse

La trace est donnée par ... **Trace** et le déterminant par ... **Determinant**. La matrice inverse est donnée sans surprise par **MatrixInverse**.

```
> restart;with(LinearAlgebra):
  M:=<<a|b|c>,<d|e|f>,<g|h|i>>;
  trM:=Trace(M);
  detM:=Determinant(M);
  invM:=MatrixInverse(M);
```

$$M := \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$trM := a + e + i$$

$$detM := a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c$$

$$invM := \begin{bmatrix} \left[\frac{e\,i - f\,h}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c}, -\frac{-h\,c + b\,i}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c}, \frac{b\,f - e\,c}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c} \right], \\ \left[-\frac{d\,i - g\,f}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c}, \frac{a\,i - g\,c}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c}, -\frac{a\,f - d\,c}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c} \right], \\ \left[\frac{d\,h - g\,e}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c}, -\frac{a\,h - g\,b}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c}, \frac{a\,e - d\,b}{a\,e\,i - a\,f\,h + d\,h\,c - d\,b\,i + g\,b\,f - g\,e\,c} \right] \end{bmatrix}$$

Noyau et rang

Le noyau de la matrice *A*, c'est à dire le sous-ensemble de l'ensemble de départ dont l'image est le vecteur nul de l'ensemble d'arrivée, a une base dont les vecteurs sont donnés par la commande **NullSpace(A)**.

```
> A:=<<1|0|0|1>,<1|0|1|1>,<0|0|1|0>>;
  KerA:=NullSpace(A);
```

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

$$KerA := \left\{ \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right\}$$

Le rang de A, c'est à dire la dimension du sous-espace de l'ensemble d'arrivée engendré par l'application linéaire dont A est une matrice représentative, est donné par **Rank(A)** :

```
> rgA:=Rank(A);
```

$$rgA := 2$$

Polynôme caractéristique; valeurs propres; vecteurs propres.

Soit A une matrice carrée, qu'on veut diagonaliser ou trigonaliser. Le polynôme caractéristique est le déterminant de la matrice $A - \lambda I$ que Maple renvoie avec la commande **CharacteristicPolynomial(A,lambda)** :

```
> A:=<<-2,3,10>|<2,7,-4>|<3,-8,-3>>;
  polcaracA:=CharacteristicPolynomial(A,lambda);
```

$$A := \begin{bmatrix} -2 & 2 & 3 \\ 3 & 7 & -8 \\ 10 & -4 & -3 \end{bmatrix}$$

$$polcaracA := 282 + \lambda^3 - 2\lambda^2 - 97\lambda$$

Les valeurs prores de A sont les valeurs de λ qui annulent le polynôme caractéristique. La commande **Eigenvalues(A)** renvoie en principe ces valeurs; encore faut-il savoir résoudre l'équation!!

```
> vpA:=Eigenvalues(A);#par défaut, l'affichage des valeurs
  propres se fait en colonne
  vpAl:=Eigenvalues(A,output=list);#les valeurs propres sont
  affichées dans une liste avec l'option output=list
```

$$vpA := \begin{bmatrix} 3 \\ -\frac{1}{2} + \frac{1}{2}\sqrt{377} \\ -\frac{1}{2} - \frac{1}{2}\sqrt{377} \end{bmatrix}$$

$$vpAl := \left[3, -\frac{1}{2} + \frac{1}{2}\sqrt{377}, -\frac{1}{2} - \frac{1}{2}\sqrt{377}\right]$$

A chaque valeur propre est associé un espace propre, engendré par un ou plusieurs vecteurs propres. La commande **`Eigenvectors(A)`** renvoie deux informations : la première est un vecteur colonne rappelant les valeurs propres de *A*; la seconde est une matrice dont la première colonne donne le vecteur propre associé à la première valeur propre, la seconde colonne donne le vecteur propre associé à la seconde valeur propre, etc..

```
> vepA:=Eigenvectors(A);
```

$$vepA := \left[\begin{array}{c} 3 \\ -\frac{1}{2}+\frac{1}{2}\sqrt{377} \\ -\frac{1}{2}-\frac{1}{2}\sqrt{377} \end{array}\right], \left[\left[\begin{array}{c} \frac{14}{13} \\ \frac{46\left(-\frac{77}{2}+\frac{3}{2}\sqrt{377}\right)}{\left(-\frac{681}{2}+\frac{43}{2}\sqrt{377}\right)\left(\frac{3}{2}+\frac{1}{2}\sqrt{377}\right)} \\ \frac{46\left(-\frac{77}{2}-\frac{3}{2}\sqrt{377}\right)}{\left(-\frac{681}{2}-\frac{43}{2}\sqrt{377}\right)\left(\frac{3}{2}-\frac{1}{2}\sqrt{377}\right)} \end{array}\right], \left[\begin{array}{c} \frac{31}{26} \\ \frac{1}{2}\,\frac{-\frac{1499}{2}+\frac{9}{2}\sqrt{377}}{-\frac{681}{2}+\frac{43}{2}\sqrt{377}} \\ \frac{1}{2}\,\frac{-\frac{1499}{2}-\frac{9}{2}\sqrt{377}}{-\frac{681}{2}-\frac{43}{2}\sqrt{377}} \end{array}\right], \left[\begin{array}{c} 1 \\ 1 \\ 1 \end{array}\right]\right]$$

L'option **`output=list`** fournit une liste de listes : chaque sous-liste affiche une valeur propre, indique son ordre de multiplicité et fournit les vecteurs de la base du sous-espace propre correspondant :

```
> vpspA:=Eigenvectors(A,output=list);
```

$$vpspA := \left[\left[-\frac{1}{2}+\frac{1}{2}\sqrt{377}, 1, \left\{\left[\begin{array}{c} \frac{46\left(-\frac{77}{2}+\frac{3}{2}\sqrt{377}\right)}{\left(-\frac{681}{2}+\frac{43}{2}\sqrt{377}\right)\left(\frac{3}{2}+\frac{1}{2}\sqrt{377}\right)} \\ \frac{1}{2}\,\frac{-\frac{1499}{2}+\frac{9}{2}\sqrt{377}}{-\frac{681}{2}+\frac{43}{2}\sqrt{377}} \\ 1 \end{array}\right]\right\}\right], \left[\right.$$

$$-\frac{1}{2}-\frac{1}{2}\sqrt{377},\ 1,\ \left\{\left[\begin{array}{c}\dfrac{46\left(-\frac{77}{2}-\frac{3}{2}\sqrt{377}\right)}{\left(-\frac{681}{2}-\frac{43}{2}\sqrt{377}\right)\left(\frac{3}{2}-\frac{1}{2}\sqrt{377}\right)}\\ \dfrac{1}{2}\\ 1\end{array}\right]\right\},\ \left[3,\ 1,\ \left\{\left[\begin{array}{c}\frac{14}{13}\\ \frac{31}{26}\\ 1\end{array}\right]\right\}\right]$$

Quand on y réfléchit et qu'on a en mémoire le temps passé à calculer - sans faire d'erreur- des valeurs propres et leurs vecteurs propres correspondants, les possibilités de Maple dans la domaine de l'algèbre linéaire laissent songeur.

Tests sur les matrices

Une batterie de tests sur les matrices est disponible dans le paquetage **`LinearAlgebra.`**

Soit *A* et *B* deux matrices.

La commande **`Equal(A,B)`** teste leur égalité; la réponse est true (vrai : les matrices sont égales) ou false (faux : les matrices sont différentes).

La commande **`IsSimilar`** teste si elles sont semblables; la réponse est true (vrai : il existe une matrice non singulière *T* telle que $B = T\,A\,T^{-1}$) ou false. L'option **`output=['query','C']`** permet d'afficher la réponse au test et, si la réponse est true, la matrice de passage *T* sous forme d'une liste de listes.

```
> restart;
  with(LinearAlgebra):#chargement du paquetage LinearAlgebra
  A:=Matrix([[(sqrt(5)+sqrt(2)*sqrt(5-sqrt(5))+1)/4,-sqrt(2)*
  sqrt(5-sqrt(5))/2],[sqrt(2)*sqrt(5-sqrt(5))/4,(sqrt(5)-sqrt
  (2)*sqrt(5-sqrt(5))+1)/4]]);
  B:=Matrix([[(sqrt(5)+1)/4,-sqrt(2)*sqrt(5-sqrt(5))/4],[sqrt
  (2)*sqrt(5-sqrt(5))/4,(sqrt(5)+1)/4]]);
```

$$A := \begin{bmatrix} \frac{1}{4}\sqrt{5} + \frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}} + \frac{1}{4} & -\frac{1}{2}\sqrt{2}\sqrt{5-\sqrt{5}} \\ \frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}} & \frac{1}{4}\sqrt{5} - \frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}} + \frac{1}{4} \end{bmatrix}$$

$$B := \begin{bmatrix} \frac{1}{4}\sqrt{5} + \frac{1}{4} & -\frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}} \\ \frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}} & \frac{1}{4}\sqrt{5} + \frac{1}{4} \end{bmatrix}$$

Les matrices **A** et **B** sont-elles égales?

```
> Equal(A,B);
```

false

Elles sont différentes. Sont-elles semblables?

```
> IsSimilar(A,B);
```

true **(4.1)**

Elles sont semblables. Quelle est la matrice de passage *T*?

```
> q,T:=IsSimilar(A,B,output=['query','C']);
```

$$q, T := true, \left[\left[1, 0\right], \left[\frac{3}{40}\sqrt{2}\sqrt{5-\sqrt{5}}\sqrt{5} - \frac{1}{2} + \frac{1}{8}\sqrt{2}\sqrt{5-\sqrt{5}} - \frac{1}{80}\sqrt{2}\sqrt{5-\sqrt{5}}\left(5 + \sqrt{5}\right)\left(\sqrt{5} + \sqrt{2}\sqrt{5-\sqrt{5}} + 1\right), 1 - \frac{1}{20}\left(-5 + \sqrt{5}\right)\left(5 + \sqrt{5}\right)\right]\right]$$

On cherche à simplifier la matrice *T* en mappant la commande **simplify** sur chaque composante de la matrice.

```
> T:=map(simplify,T);
```

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

Par précaution, on vérifie que $B = T.A.T^{-1}$.

```
> Equal(B,T.A.T^(-1));
```

true

Soit à présent une matrice *M*. La commande **IsUnitary(M)** teste si elle est unitaire (formée d'éléments tous égaux à 1). La commande **IsZero(M)** teste si elle est nulle (formée d'éléments tous égaux à 0). La commande **IsOrthogonal(M)** teste si elle est orthogonale, c'est à dire si elle vérifie $M.M^t = I$. Enfin, comble du raffinement, la commande **IsDefinite(M)** teste si la matrice est définie positive. En option, on peut préciser sa question sous la forme suivante : **query= 'positive_definite'** ou **'positive_semidefinite'** ou **'negative_definite'** ou **'negative_semidefinite'.**

```
> IsOrthogonal(B);simplify(B.Transpose(B));
```

true

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

```
> M1:=<<4,0>|<0,6>>;M2:=<<4,0>|<0,-2>>;M3:=<<-2,0>|<0,-6>>;
  IsDefinite(M1);IsDefinite(M2);IsDefinite(M3);
  IsDefinite(M3,query=negative_definite);
```

$$M1 := \begin{bmatrix} 4 & 0 \\ 0 & 6 \end{bmatrix}$$

$$M2 := \begin{bmatrix} 4 & 0 \\ 0 & -2 \end{bmatrix}$$

$$M3 := \begin{bmatrix} -2 & 0 \\ 0 & -6 \end{bmatrix}$$

true

false

false

true

Système matriciel associé à un système d'équations linéaires

Il est fréquent de transformer un système d'équations linéaires en une équation matricielle. Disponible dans le paquetage **LinearAlgebra**, l'instruction correspondante est **GenerateMatrix([equations],[inconnues])** où **equations** est une suite d'équations linéaires et **inconnues** une suite de variables. Maple se charge de tout : il affiche en sortie la matrice des coefficients et le vecteur colonne des seconds membres qu'on a intérêt à assigner.

```
> restart;with(LinearAlgebra):
  equations:=a11*x+a12*y+a13*z=b1,a21*x+a22*y+a23*z=b2,a31*x+
  a32*y+a33*z=b3;
  inconnues:=x,y,z;
  A,b:=GenerateMatrix([equations],[inconnues]);
```

$$equations := a11\,x + a12\,y + a13\,z = b1,\ a21\,x + a22\,y + a23\,z = b2,\ a31\,x + a32\,y + a33\,z = b3$$

$$inconnues := x, y, z$$

$$A, b := \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}, \begin{bmatrix} b1 \\ b2 \\ b3 \end{bmatrix}$$

La matrice A étant supposée non singulière, la solution du système initial est donnée par $\left(\frac{1}{A}\right).b$:

```
> sol:=A^(-1).b;
```

$$sol := \left[\right.$$

$$\left[((a22\,a33 - a23\,a32)\,b1) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32) \right.$$

$$- ((a12\,a33 - a13\,a32)\,b2) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32)$$

$$+ ((a12\,a23 - a13\,a22)\,b3) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32) \Big],$$

$$\left[\right.$$

$$- ((-a31\,a23 + a21\,a33)\,b1) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32)$$

$$+ ((-a31\,a13 + a11\,a33)\,b2) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32)$$

$$- ((-a21\,a13 + a11\,a23)\,b3) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32) \Big],$$

$$\left[((-a31\,a22 + a21\,a32)\,b1) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32) \right.$$

$$- ((-a31\,a12 + a11\,a32)\,b2) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32)$$

$$+ ((-a21\,a12 + a11\,a22)\,b3) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32) \Big]\Big]$$

Mais il est préférable de résoudre un système d'équations linéaires en invoquant la commande dédiée **`LinearSolve`** du paquetage **`LinearAlgebra`** parce que sa programmation est très efficace. La syntaxe minimale est **`LinearSolve(A,b)`** et le résultat est un vecteur colonne.

```
> LinearSolve(A,b);
```

$$\left[\left[(a12\,a23\,b3 - a12\,b2\,a33 + a13\,a32\,b2 - a13\,a22\,b3 + b1\,a22\,a33 - b1\,a23\,a32) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32 + a11\,a22\,a33 - a11\,a23\,a32) \right], \right. \quad \textbf{(5.1)}$$

$$\left[\right.$$

$$- (a11\,a23\,b3 - a11\,b2\,a33 - a21\,a13\,b3 - a23\,a31\,b1 + b2\,a31\,a13 + a21\,b1\,a33) \Big/ (a31\,a12\,a23 - a31\,a13\,a22 - a21\,a12\,a33 + a21\,a13\,a32$$

$$+ a11\ a22\ a33 - a11\ a23\ a32\ )\ ],$$
$$[\ (\ -a21\ a12\ b3 - a11\ a32\ b2 + a11\ a22\ b3 - a22\ a31\ b1 + a32\ a21\ b1$$
$$+ a31\ a12\ b2\ )\ /\ (a31\ a12\ a23 - a31\ a13\ a22 - a21\ a12\ a33 + a21\ a13\ a32$$
$$+ a11\ a22\ a33 - a11\ a23\ a32\ )\ ]\ ]$$

Quand les coefficients de la matrice **A** et du vecteur **b** sont des symboles et des nombres rationnels, l'output respecte le principe d'exactitude du résultat. Quand interviennent des nombres avec au moins l'un d'entre eux écrit sous forme décimale, le solveur affiche une solution approchée.

```
> Me:=Matrix(3,3,[[3/10,3/10,4/10],[1/10,4/10,5/10],[4/10,4/10,
  2/10]]):#Matrice à coefficients rationnels
  bo:=Vector([b1,b2,b3]):#Vecteur à coefficients symboliques
  be:=Vector([5/10,2/10,3/10]):#Vecteur à coefficients
  rationnels
  Ma:=Matrix(3,3,[[0.3,0.3,0.4],[0.1,0.4,0.5],[0.4,0.4,0.2]])
  :#Matrice à coefficients "réels". Noter que Me=Ma.
  ba:=Vector([0.5,0.2,0.3]):#Vecteur à coefficients "réels".
  Noter que ba=be.
  sol1:=LinearSolve(Me,bo);#cas d'une matrice à coefficients
  rationnels et des seconds membres symboliques
  sol2:=LinearSolve(Me,be);#cas d'une matrice à coefficients
  rationnels et des seconds membres rationnels
  sol3:=LinearSolve(Ma,bo);#cas d'une matrice à coefficients
  "réels" et des seconds membres symboliques
  sol4:=LinearSolve(Ma,ba);#cas d'une matrice à coefficients
  "réels" et des seconds membres "réels"
```

$$sol1 := \begin{bmatrix} \frac{1}{3}\ b3 + 4\ b1 - \frac{10}{3}\ b2 \\ \frac{11}{3}\ b3 - 6\ b1 + \frac{10}{3}\ b2 \\ -3\ b3 + 4\ b1 \end{bmatrix}$$

$$sol2 := \begin{bmatrix} \frac{43}{30} \\ -\frac{37}{30} \\ \frac{11}{10} \end{bmatrix}$$

$$sol3 := \begin{bmatrix} 0.3333333333\ b3 + 4.\ b1 - 3.333333333\ b2 \\ 3.666666667\ b3 - 6.\ b1 + 3.333333333\ b2 \\ -3.\ b3 + 4.\ b1 \end{bmatrix}$$

(5.2)

$$sol4 := \begin{bmatrix} 1.43333333333333290 \\ -1.23333333333333295 \\ 1.09999999999999986 \end{bmatrix} \quad \textbf{(5.2)}$$

Travailler avec des nombres décimaux plutôt que des rationnels offre l'avantage indéniable d'obtenir rapidement une solution facilement lisible mais cet avantage se paye par la perte probable d'un résultat exact. Ainsi, les deux tests suivants devraient normalement être positifs puisque ,d'une part, *Me.sol2 = be*, *Ma.sol4 = ba* et *be = ba*, et d'autre part, *Me.sol1 = bo*, *Ma.sol3 = bo*.

```
> Equal(Me.sol2,Ma.sol4);
  Equal(Me.sol1,Ma.sol3);
```

true

false **(5.3)**

Les experts des systèmes linéaires accèdent à des méthodes sophistiquées en signalant leur choix par l'option `method =nom`, nom étant `'subs'` (à utiliser pour les matrices *A* triangulaires), `'Cholesky'` (pour les matrices *A* symétriques définies positives), `'SparseIterative'` (pour les matrices *A* ayant de nombreux éléments nuls), etc.

Application d'opérateurs sur tous les éléments d'un vecteur ou d'une matrice

Il est souvent utile de simplifier ou, plus généralement, de transformer tous les éléments d'un vecteur ou d'une matrice (entre autres, passer de nombres écrits sous forme rationnelle à une forme décimale). Pour appliquer un opérateur aux éléments, on dispose de deux instructions proches : `map` et `Map`, `Map` étant disponible dans le paquetage `LinearSolve`. Dans les deux cas, on donne d'abord l'opérateur puis le vecteur ou la matrice à transformer. La syntaxe est par conséquent `map(operateur,vecteur/matrice)` et `Map(operateur,vecteur/matrice)`. Le résultat est le même à ceci près que `map` crée un nouveau vecteur/matrice alors que `Map` réassigne le vecteur/la matrice de départ.

```
> matr:=Matrix(2,2,[[4,1],[2,3]]);#définition d'une matrice
  matrinv:=MatrixInverse(matr);#assignation à matrinv de
  l'inverse de matr
  map(evalf,matrinv);#map applique l'évaluation décimale sur
  les termes de matrinv
  matrinv;#matrinv n'est pas modifiée par map
  Map(evalf,matrinv);#Map applique l'évaluation décimale sur
  les termes de matrinv
  matrinv;#matrinv est réassignée
```

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

$$matrinv := \begin{bmatrix} \frac{3}{10} & -\frac{1}{10} \\ -\frac{1}{5} & \frac{2}{5} \end{bmatrix}$$

$$\begin{bmatrix} 0.3000000000 & -0.1000000000 \\ -0.2000000000 & 0.4000000000 \end{bmatrix}$$

$$\begin{bmatrix} \frac{3}{10} & -\frac{1}{10} \\ -\frac{1}{5} & \frac{2}{5} \end{bmatrix}$$

$$\begin{bmatrix} 0.3000000000 & -0.1000000000 \\ -0.2000000000 & 0.4000000000 \end{bmatrix}$$

$$\begin{bmatrix} 0.3000000000 & -0.1000000000 \\ -0.2000000000 & 0.4000000000 \end{bmatrix} \quad \textbf{(6.1)}$$

Tous les opérateurs programmés par Maple sont utilisables (**simplify**, **expand**, **evalf**, **sqrt**, ...) mais on peut aussi créer ses propres opérateurs sous forme de fonctions procédures :

```
> Map(x->log(abs(x)),matrinv);
```

$$\begin{bmatrix} -1.203972804 & -2.302585093 \\ -1.609437912 & -0.9162907319 \end{bmatrix} \quad \textbf{(6.2)}$$

La commande **Zip** du paquetage **LinearAlgebra** permet, quant à elle, d'effectuer des opérations sur **deux** variables, la première étant un élément d'un vecteur ou d'une matrice, la seconde l'élément correspondant dans un autre vecteur ou une autre matrice de même dimension. En notant **oper** l'opérateur agissant sur deux variables, **obj1** le premier vecteur ou la première matrice et **obj2** le second vecteur ou la seconde matrice, la syntaxe est **Zip(oper,obj1,obj2)**. Par exemple :

```
> mat1:=Matrix(2,2,[[0.1,0.7],[-0.5,1]]);
  mat2:=Matrix(2,2,[[-0.3,0.8],[0.2,-0.9]]);
  Zip((x,y)->sqrt(x^2+y^2),mat1,mat2);
```

$$mat1 := \begin{bmatrix} 0.1 & 0.7 \\ -0.5 & 1 \end{bmatrix}$$

$$mat2 := \begin{bmatrix} -0.3 & 0.8 \\ 0.2 & -0.9 \end{bmatrix}$$

$$\begin{bmatrix} 0.3162277660 & 1.063014581 \\ 0.5385164807 & 1.345362405 \end{bmatrix} \quad \textbf{(6.3)}$$

Exercices

Exercice M1

Calculer "à la main" puis avec Maple les déterminants des matrices suivantes :

1. $M_1 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$

2. $M_2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

3. $M_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}$

4. $M_4 = \begin{bmatrix} 3 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 3 \end{bmatrix}$

5. $M_5 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix}$

6. $M_6 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 1 & -4 & 3 \\ 3 & -4 & -1 & 2 \\ 4 & 3 & -2 & -1 \end{bmatrix}$

7. $M_7 = \begin{bmatrix} 1 & 0 & 2 & 3 & 0 \\ -1 & 2 & 0 & 1 & 4 \\ -2 & 1 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 & -1 \\ 0 & 0 & -1 & 2 & 1 \end{bmatrix}$

8. $M_8 = \begin{bmatrix} 1 & 0 & -1 & 3 & 4 \\ 2 & 1 & 0 & -1 & 1 \\ 1 & 0 & 3 & 1 & 5 \\ -1 & 2 & 1 & 0 & 3 \\ 0 & 0 & 5 & 2 & 1 \end{bmatrix}$

Quelle méthode est la plus fiable?

Exercice M2

On considère la matrice carrée d'ordre n :

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

1. Construire une fonction-procédure qui, à tout n, associe la matrice M_n.

2. Calculer les déterminants des matrices M_n pour $n = 1,.., 10$.

3. Calculer le rapport $\frac{det(M_{n+1})}{det(M_n)}$ pour différentes valeurs de n. En déduire une relation entre $det(M_{n+1})$ et $det(M_n)$ puis donner une formule de calcul de $det(M_n)$ pour tout n.

Exercice M3

On considère la matrice carrée d'ordre n :

$$M_n = \begin{bmatrix} 2 & 1 & 0 & 0 & ... & 0 \\ 1 & 2 & 1 & 0 & ... & 0 \\ 0 & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & 0 \\ ... & ... & ... & ... & ... & 1 \\ 0 & 0 & ... & 0 & 1 & 2 \end{bmatrix}$$

1. Construire une fonction-procédure qui, à tout n, associe la matrice M_n.

2. Calculer les déterminants des matrices M_n pour $n = 1,.., 20$.

3. Enoncer une conjecture sur $det(M_n)$.

Exercice M4

Calculer le déterminant de la matrice :

$$\begin{bmatrix} \binom{3}{0} & \binom{4}{0} & \binom{5}{0} & \binom{6}{0} \\ \binom{3}{1} & \binom{4}{1} & \binom{5}{1} & \binom{6}{1} \\ \binom{3}{2} & \binom{4}{2} & \binom{5}{2} & \binom{6}{2} \\ \binom{3}{3} & \binom{4}{3} & \binom{5}{3} & \binom{6}{3} \end{bmatrix}$$

dont les coefficients sont les combinaisons de p ($p = 0, 1, 2, 3$) éléments parmi n ($n = 3, 4, 5, 6$).

Exercice M5

Calculer les déterminants des matrices suivantes :

$$M_1 = \begin{bmatrix} 1 & 1 & 1 \\ b+c & c+a & a+b \\ bc & ca & ab \end{bmatrix}$$

$$M_2 = \begin{bmatrix} a & a+1 & a+2 \\ a+1 & a+2 & a+3 \\ a+2 & a+3 & a+4 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 0 & a & b & c \\ a & 0 & c & b \\ b & c & 0 & a \\ c & b & a & 0 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} a & 0 & b & a \\ b & b & 0 & a \\ b & a & 0 & b \\ a & b & 0 & a \end{bmatrix}$$

On demande de présenter le résultat le plus simple possible.

Exercice M6

On considère la matrice suivante :

$$M = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

1. Montrer que M est inversible.
2. Calculer M^{-1}.
3. Vérifier que $M.M^{-1} = M^{-1}.M = I$.

Exercice M7

On considère la matrice suivante :

$$M = \begin{bmatrix} 1 & a \\ -a & 2 \end{bmatrix}$$

1. Montrer que M est inversible.
2. Calculer M^{-1}.
3. Vérifier que $M.M^{-1} = M^{-1}.M = I$.

Exercice M8

On considère la matrice suivante :

$$M = \begin{bmatrix} 2 & -1 & 1 & 2 \\ -1 & 2 & -1 & 1 \\ 1 & -1 & 2 & -1 \\ 2 & 1 & -1 & 2 \end{bmatrix}$$

1. Diagonaliser M. On notera G la matrice diagonale et P la matrice de passage.
2. Vérifier que $M = P.G.P^{-1}$.

Exercice M9

Résoudre les systèmes d'équations linéaires suivants :

1.
$$\begin{cases} 2x - y - z = 4 \\ 3x + 4y - 2z = 11 \\ 3x - 2y + 4z = 11 \end{cases}$$

2.
$$\begin{cases} 3x + y + z = 1 \\ x - y + 2z = 2 \\ x + 3y - 3z = -3 \end{cases}$$

3.
$$\begin{cases} 3x + 2y + z = 5 \\ 2x + 3y + z = 1 \\ 2x + y + 3z = 11 \\ x + 2y + 3z = 7 \end{cases}$$

4.

$$\begin{cases} x+y+z+t=0 \\ x+2y+3z+4t=0 \\ x+3y+6z+10t=0 \\ x+4y+10z+20t=0 \end{cases}$$

5.

$$\begin{cases} x+y-3z=1 \\ 2x+y-2z=31 \\ x+y+z=3 \\ x+2y-3z=1 \end{cases}$$

6.

$$\begin{cases} x+y+z+t+u=7 \\ 3x+2y+z+t-3u=-2 \\ y+2z+2t+6u=23 \\ 5x+4y+3z+3t-u=12 \end{cases}$$