Lycée Pierre-Gilles de Gennes **BCPST2**
Mathématiques **2019-2020**

Révisions/Formulaire 05

Révisions Algèbre: Systèmes, vecteurs et matrices

Vecteurs dans $\mathbb{K}^n$, opérations

Cette section doit se lire deux fois. Une première fois en remplaçant $\mathbb{K}$ par $\mathbb{R}$, une deuxième fois en remplaçant $\mathbb{K}$ par $\mathbb{C}$. Les éléments de $\mathbb{K}$ se nomment les *scalaires*. Ce sont des nombres.

Fixons $n \in \mathbb{N}^*$. Un élément x de $\mathbb{K}^n$ est un n-uplet d'éléments de $\mathbb{K}$

$$x = \underbrace{(x_1, \ldots, x_n)}_{n \text{ termes}} \text{ ou } x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

L'égalité de deux n-uplets est équivalente à l'égalité de chacune des *composantes* (ou de chacune des *entrées*) des n-uplets.

$$(x_1, \ldots, x_n) = (y_1, \ldots, y_n) \Leftrightarrow \forall k \in \{1, \ldots, n\}, x_k = y_k$$

La première notation est la notation mathématique standard pour un n-uplet. La deuxième, officiellement au programme, est utile lorsque l'on utilise un logiciel de calcul standard (MATLAB, module Python `numpy`...) et est, d'un point de vue typographique assez pratique.

Cette deuxième notation identifie implicitement un n-uplet de $\mathbb{K}^n$ avec une matrice « colonne » $n \times 1$ à entrées dans $\mathbb{K}$. On **n**'identifie **pas** automatiquement [1] un n-uplet $(x_1, \ldots, x_n)$ avec la matrice « ligne » $\begin{pmatrix} x_1 & \ldots & x_n \end{pmatrix}$.

On note $\mathbb{K}^n$ l'ensemble des n-uplets d'éléments de $\mathbb{K}$.

$$\mathbb{K}^n = \{x = (x_1, \ldots, x_n), x_1 \in \mathbb{K}, \ldots, x_n \in \mathbb{K}\}$$

Les deux notations (en ligne avec des ,, en colonne) traduit l'identification quasi automatique entre $\mathbb{K}^n$ et $\mathscr{M}_{n,1}(\mathbb{K})$ l'ensemble des matrices possèdant une colonne et n lignes.

Sur $\mathbb{K}^n$, on définit deux opérations :

— L'une, *interne*, l'*addition*, notée $+$, définie par

$$\forall x, y, z \in \mathbb{K}^n, (z = x + y \Leftrightarrow \forall k \in \{1, \ldots, n\}, z_k = x_k + y_k)$$

On notera que les deux arguments et le résultat d'une addition dans $\mathbb{K}^n$ sont des éléments de $\mathbb{K}^n$

— L'autre, *externe*, la *multiplication par un scalaire*, notée ., définie par

$$\forall \lambda \in \mathbb{K}, \forall x, y \in \mathbb{K}^n, (y = \lambda . x \Leftrightarrow \forall k \in \{1, \ldots, n\}, y_k = \lambda . x_k)$$

On notera que dans une telle multiplication, le premier argument est un élément de $\mathbb{K}$, le second et le résultat sont des éléments de $\mathbb{K}^n$

1. Noter la différence subtile de notation avec présence ou pas de virgule

— On pose, pour $k \in \{1,\dots,n\}$, $e_k = (0,\dots,0,1,0,\dots,0)$ où le 1 apparait en position k.
Ceci permet de réécrire, à l'aide des opérations tout juste définies

$$(x_1,\dots,x_n) = \sum_{k=1}^{n} x_k.e_k$$

ou, de manière équivalente, en utilisant la notation en colonne

$$\begin{pmatrix} x_1 \\ \vdots \\ . \\ . \\ . \\ \vdots \\ x_n \end{pmatrix} = \sum_{k=1}^{n} x_k.e_k, \ (e_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \rightarrow \text{ligne } k, \ k \in \{1,\dots,n\})$$

— Pour $x, y \in \mathbb{K}^n$, $\lambda, \mu \in \mathbb{K}$, on a

$$\begin{aligned} \lambda.x + \mu.y &= \lambda.\left(\sum_k x_k.e_k\right) + \mu.\left(\sum_k y_k.e_k\right) \\ (\text{déf. } \lambda.x,\ \mu.y) &= \sum_k (\lambda.x_k).e_k + \sum_k (\mu.y_k).e_k \\ (\text{déf. } \lambda.x + \mu.y) &= \sum_k (\lambda.x_k + \mu.y_k).e_k \end{aligned}$$

Dans ce petit jeu d'écriture, il faut identifier la nature des opérations : $+$ c'est une addition entre vecteurs ou entre scalaires ? . c'est une multiplication scalaire.vecteur ou scalaire.scalaire ? A vous de mettre des couleurs !

On peut (et on le fait !, notamment lors de l'implémentation de programmes de calcul en Python) utiliser l'ensemble d'indices $\{0,\dots,n-1\}$ à la place de $\{1,\dots,n\}$. Cela ne change rien aux grands principes mais peut mener à de petits décalages dans les formules.

Une équation linéaire homogène

Une équation linéaire scalaire [2] homogène d'inconnue $x = (x_1,\dots,x_n) \in \mathbb{K}^n$ a la forme

$$(L) : a_1.x_1 + \cdots + a_n.x_n = 0$$

pour un certain n-uplet $(a_1,\dots,a_n) \in \mathbb{K}^n$. L'ensemble des solutions de (L) est

$$L = \{x = (x_1,\dots,x_n) \in \mathbb{K}_n,\ a_1.x_1 + \cdots + a_n.x_n = 0\}$$

Donner un ensemble $L \subset \mathbb{K}^n$ de cette manière, c'est en donner une *équation cartésienne i.e.* , pour une certaine fonction $f : \mathbb{K}^n \rightarrow \mathbb{K}$,

$$L = \{x \in \mathbb{K}^n,\ f(x) = 0\}$$

Une équation cartésienne est pratique pour tester qu'un élément donné x appartient ou pas à l'ensemble L. Elle ne nous permet pas directement de trouver au moins un x (et encore moins *tous* les x) dans L.

2. Par ceci, on entend que le signe $=$ de l'équation est une égalité entre nombres, *i.e.* entre scalaires

A cette fin, on peut résoudre cette équation [3], c'est à dire déterminer une *forme paramétrique* de l'ensemble L, *i.e.* déterminer un ensemble Λ de paramètres (ici de la forme $\mathbb{R}^d$) et une fonction $g : \Lambda \to \mathbb{R}^n$ tels que

$$L = g(\Lambda) = \{g(\lambda), \lambda \in \Lambda\} = \{x \in \mathbb{R}^n, \exists \lambda \in \Lambda, x = g(\lambda)\}$$

Commençons par trois exemples

1. Résoudre $(L) : 2x+3y = 0$ d'inconnue $(x,y) \in \mathbb{R}^2$. On a
   (x,y) satisfait (L) si et seulement si $x = -\frac{2}{3}.y$, *i.e.* (passage logique délicat), si et seulement si

   $$(P_L) : \text{il existe } \lambda \in \mathbb{R} \text{ tel que } y = \lambda \text{ et } x = -\frac{2}{3}.\lambda$$

   On a donc démontré que

   $$L = \{(-\frac{2}{3}\lambda, \lambda), \lambda \in \mathbb{R}\} = \{\lambda.(-\frac{2}{3}, 1), \lambda \in \mathbb{R}\}$$

   Un élément de L est obtenu en prenant $\lambda = 1 : (-\frac{2}{3}, 1) \in L$ ou en prenant $\lambda = -\frac{3}{2} : (1, -\frac{3}{2}) \in L$. Il y a une infinité de possibilités pour λ, chacune mène à une solution différente de (L).
   Un point crucial est l'interprétation géométrique de la résolution : si on se place dans le plan muni d'un repère $\mathscr{R} = (O, \vec{i}, \vec{j})$ alors l'ensemble des points M dont les coordonnées (x,y) sont dans l'ensemble L est une droite D_L passant par O. L'équation (L) est une équation cartésienne de D_L dans le repère $\mathscr{R}$, la réprésentation (P_L) est un système d'équations paramétriques de D_L dans le repère $\mathscr{R}$. Le vecteur de coordonnées $(-\frac{2}{3}, 1)$ dans la base $(\vec{i}, \vec{j})$ du plan est un vecteur directeur de D_L
2. Résoudre $(L) : 2x+3y = 0$ d'inconnue $(x,y,z) \in \mathbb{R}^3$. On a
   (x,y,z) satisfait (L) si et seulement si $x = -\frac{2}{3}.y$, *i.e.* (passage logique délicat), si et seulement si

   $$(P_L) : \text{il existe } \lambda, \mu \in \mathbb{R} \text{ tels que } z = \mu, y = \lambda \text{ et } x = -\frac{2}{3}.\lambda$$

   On a donc démontré que

   $$L = \{(-\frac{2}{3}\lambda, \lambda, \mu), \lambda, \mu \in \mathbb{R}\} = \{\lambda.(-\frac{2}{3}, 1, 0) + \mu.(0,0,1), \lambda, \mu \in \mathbb{R}\}$$

   Un élément de L est obtenu en prenant $\lambda = 1, \mu = 0 : (-\frac{2}{3}, 1, 0) \in L$ ou en prenant $\lambda = -\frac{3}{2}, \mu = 1 : (1, -\frac{3}{2}, 1) \in L$. Il y a une infinité de possibilités pour le couple (λ, μ), chacune mène à une solution différente de (L).

3. Résoudre complètement une équation, c'est le passage « cartésien vers paramétrique ». C'est évidemment une problématique fondamentale des mathématiques, elle n'est pas limitée au cas linéaire. Le cas linéaire a ceci de singulier que l'on obtient une réponse complète (il n'y a pas de zone d'ombre), cohérente (seules des fonctions linéaires interviennent) et effective (on dispose d'algorithmes de calcul) et *relativement* simple. Pour un exemple non linéaire simple, on peut penser au cas du cercle unité $\mathbb{S}^1$ dans le plan. On a

$$\begin{aligned}
\mathbb{S}^1 &= \{(x,y) \in \mathbb{R}^2, x^2+y^2-1 = 0\} && \text{(forme cartésienne)}\\
&= \{(\cos\theta, \sin\theta), \theta \in \mathbb{R}\} && \text{(forme paramétrique)}\\
&= \{(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, t \in \mathbb{R}\} \cup \{(-1,0)\} && \text{(forme paramétrique bis)}
\end{aligned}$$

L'interprétation géométrique de la résolution est le suivant : si on se place dans l'espace muni d'un repère $\mathscr{R} = (O, \vec{i}, \vec{j}, \vec{k})$ alors l'ensemble des points M dont les coordonnées (x, y, z) sont dans l'ensemble L est un plan P_L passant par O. L'équation (L) est une équation cartésienne de P_L dans le repère $\mathscr{R}$, la réprésentation (P_L) est un système d'équations paramétriques de P_L dans le repère $\mathscr{R}$. Les vecteurs $\vec{u}$ et $\vec{v}$ de coordonnées respectives $(-\frac{2}{3}, 1, 0)$ et $(0, 0, 1)$ dans la base de l'espace $(\vec{i}, \vec{j}, \vec{k})$ forment une base de P_L. On peut munir P_L du repère $(O, \vec{u}, \vec{v})$

3. Résoudre $(L) : 2x + 3y = 0$ d'inconnue $(x, y, z) \in \mathbb{C}^3$. On a (x, y, z) satisfait (L) si et seulement si $x = -\frac{2}{3}.y$, *i.e.* (passage logique délicat), si et seulement si il existe $\lambda, \mu \in \mathbb{C}$ tels que

$$z = \mu,\ y = \lambda \text{ et } x = -\frac{2}{3}.\lambda$$

On a donc démontré que

$$L = \{(-\frac{2}{3}\lambda, \lambda, \mu), \lambda, \mu \in \mathbb{C}\} = \{\lambda.(-\frac{2}{3}, 1, 0) + \mu.(0, 0, 1), \lambda, \mu \in \mathbb{C}\}$$

Un élément de L est obtenu en prenant $\lambda = 1, \mu = 0 : (-\frac{2}{3}, 1, 0) \in L$ ou en prenant $\lambda = -\frac{3}{2}i, \mu = 1 : (i, -\frac{3}{2}.i, 1) \in L$. Il y a une infinité de possibilités pour le couple (λ, μ), chacune mène à une solution différente de (L).

Ici, l'interprétation géométrique n'est pas immédiate, ou tout au moins, beaucoup moins familière, on dira, par analogie avec le cas réel, que L est un plan vectoriel dans $\mathbb{C}^3$. Dans cette analogie, $\mathbb{C}^2$ est un plan vectoriel, $\mathbb{C}$ une droite vectorielle, ce qui entre en conflit avec l'expression habituelle « $\mathbb{C}$ est le plan complexe »

Ces exemples montrent qu'il faut introduire un certain nombre de paramètres pour décrire toutes les solutions.

Passons maintenant au cas général

— Si $(a_1, \ldots, a_n) = (0, \ldots, 0) = 0$, l'équation (L) est *triviale*

$$L = \mathbb{K}^n = \{(x_1, \ldots, x_n), x_1 \in \mathbb{K}, \ldots, x_n \in \mathbb{K}\} = \{\sum_{k=1}^{n} \lambda_k.e_k,\ (\lambda_1, \ldots, \lambda_n) \in \mathbb{K}^n\}$$

— Si $(a_1, \ldots, a_n) \neq (0, \ldots, 0)$, l'équation est *non triviale* au sens où il existe au moins un n-uplet qui n'est pas solution [4] de (L)

Supposons que i est tel que $a_i \neq 0$; x est solution de (L) si et seulement si

$$x_i = -\sum_{k \neq i} \frac{a_k}{a_i}.x_k$$

i.e. (et c'est à cet endroit qu'il y a le pas logique important à franchir), si et seulement si il existe

4. Que les a_i soient réels où complexes, on peut constater que $x = (\overline{a}_1, \ldots, \overline{a}_n)$, n'est pas solution de (L) ; Une autre « non solution » s'obtient en supposant que $a_i \neq 0$–il y a au moins un tel i– et en prenant $x = a_i.e_i$.

$\lambda_1,\ldots,\lambda_n \in \mathbb{K}$ (λ_i **n'apparait pas** dans cette liste ! ! !) tels que

$$\begin{array}{rcl} x_1 & = & \lambda_1 \\ \vdots & = & \ldots \\ & & \text{ces lignes jusque la } i-1 \text{ sont semblables} \\ \vdots & = & \ldots \\ x_i & = & -\sum_{k\neq i} \frac{a_k}{a_i}\lambda_k \\ \vdots & = & \ldots \\ & & \text{ces lignes à partir de la } i+1 \text{ sont semblables} \\ \vdots & = & \ldots \\ x_n & = & \lambda_n \end{array}$$

Si on note (u_i **n'apparait pas** dans cette liste ! ! !)

$$u_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ -\frac{a_1}{a_i} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, u_{i-1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ -\frac{a_{i-1}}{a_i} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, u_{i+1} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -\frac{a_{i+1}}{a_i} \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, u_n = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -\frac{a_{i+1}}{a_i} \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \rightarrow \text{ligne } i$$

alors

$$L = \{ \sum_{k=1,k\neq i}^{n} \lambda_k.u_k, (\lambda_1, \underbrace{\ldots}_{\text{pas de } i\,!\,!}, \lambda_n) \in \mathbb{K}^{n-1} \}$$

La signification « forme paramétrique de l'ensemble L » pour ce type d'écriture provient du fait que

1. Pour trouver **un** élément de L, il suffit de donner une valeur au n ou $n-1$ uplet de paramètres λ
2. **Tout** élément de L s'obtient de cette manière ;

Systèmes homogènes

Un système (de p équations scalaires) linéaire(s) homogène(s) d'inconnue $x = (x_1,\ldots,x_n) \in \mathbb{K}^n$ a la forme

$$(S) : \begin{cases} a_{11}.x_1 + \cdots + a_{1n}.x_n & = & 0 & (L_1) \\ \vdots & & \vdots & \vdots \\ a_{p1}.x_1 + \cdots + a_{pn}.x_n & = & 0 & (L_p) \end{cases}$$

où les $p.n$ coefficients $a_{ij}, 1 \leq i \leq p, 1 \leq j \leq n$ sont fixés dans $\mathbb{K}$.

L'ensemble des solutions de (S) est

$$S = \{x = (x_1,\ldots,x_n) \in \mathbb{K}_n, a_{11}.x_1 + \cdots + a_n.x_{1n} = 0 \text{ et } \ldots \text{ et } a_{p1}.x_1 + \cdots + a_n.x_{pn} = 0\}$$

Le système (S) est issu de la *conjonction* des p équations (L_i). Si L_i est l'ensemble des solutions de (L_i) alors

$$S = L_1 \cap \cdots \cap L_p$$

Donner un ensemble $S \subset \mathbb{K}^n$ de cette manière, c'est encore en donner une *équation cartésienne*. Si on pose

$$\begin{array}{rccc} f: & \mathbb{K}^n & \to & \mathbb{K}^p \\ & x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} & \mapsto & f(x) = \begin{pmatrix} a_{11}.x_1 + \cdots + a_n.x_{1n} \\ \vdots \\ a_{p1}.x_1 + \cdots + a_n.x_{pn} \end{pmatrix} \end{array}$$

i.e.

$$\forall y = \begin{pmatrix} y_1 \\ \vdots \\ y_p \end{pmatrix} \in \mathbb{K}^p, \forall x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{K}^n, \left(y = f(x) \Leftrightarrow \left\{ \begin{array}{cccc} & a_{11}.x_1 + \cdots + a_{1n}.x_n & = & y_1 \\ \text{et} & \vdots & \vdots & \vdots \\ \text{et} & a_{p1}.x_1 + \cdots + a_{pn}.x_n & = & y_p \end{array} \right. \right)$$

alors

$$S = \{x \in \mathbb{K}^n, f(x) = 0\}$$

Résolution par élimination de GAUSS : exemples

On commence par trois exemples

1. Résoudre
$$(S): \left\{ \begin{array}{rcll} x+y & = & 0 & (L_1) \\ 2x+3y & = & 0 & (L_2) \end{array} \right. \text{ d'inconnue } (x,y) \in \mathbb{R}^2$$
On a
   (a) Etape d'élimination (x,y) satisfait (S) si et seulement si (x,y) satisfait $(S') = (L_1)$ et $(L_2) - 2(L_1)$ *i.e.*
   $$(S'): \left\{ \begin{array}{rcll} x+y & = & 0 & (L_1' \leftarrow L_1) \\ y & = & 0 & (L_2' \leftarrow L_2 - 2.L_1) \end{array} \right.$$
   L'explication fondamentale de l'équivalence et que
   — Sachant que (x,y) vérifie (S), on en déduit qu'il vérifie (S') car « (S') se déduit de (S) »
   — Sachant que (x,y) vérifie (S'), on en déduit qu'il vérifie (S) car « (S) se déduit de (S') » via les opérations
   $$L_1 \leftarrow L_1' \text{ et } L_2 \leftarrow L_2' + 2L_1'$$
   — On voit que le fait d'avoir conservé une ligne entre les deux systèmes et d'avoir modifié les autres en se servant de cette ligne conservée est crucial pour conserver le même ensemble de solutions.

   (b) Etape de remontée : Le gain de cette opération d'élimination est que l'on se retrouve avec un système (S') *échelonné* : on peut le résoudre en « remontant ». Si le système (S') est échelonné, c'est qu'on a utilisé L_1 pour *éliminer* x_1 des autres équations. On a :
   $$(S'): \left\{ \begin{array}{rcll} x+0 & = & 0 & (L_1' \leftarrow L_1) \\ y & = & 0 & (L_2' \leftarrow L_2 - 2.L_1) \end{array} \right.$$
   et donc la seule solution de (S') (et de (S)) est $(x,y) = (0,0)$

(c) Interprétation géométrique : Le point crucial de l'interprétation géométrique de la résolution est la suivante : on se place dans le plan muni d'un repère $\mathscr{R} = (O, \vec{i}, \vec{j})$ et on identifie [5] un M de coordonnées (x,y) avec le couple (x,y). M vérifie (S) si et seulement si il vérifie (L_1) et (L_2), *i.e.* M est dans l'intersection des droites L_1 et L_2. Comme O est dans chacune de ces droites, que ces droites ne sont pas parallèles, l'intersection est réduite au point O.

2. Résoudre

$$(S) : \left\{ \begin{array}{rcll} x+y & = & 0 & (L_1) \\ 2x+2y+z & = & 0 & (L_2) \end{array} \right. \text{ d'inconnue } (x,y,z) \in \mathbb{R}^3$$

On a

(a) Etape d'élimination : (x,y,z) satisfait (S) si et seulement si (x,y,z) satisfait $(S') = (L_1)$ et $(L_2) - 2(L_1)$ *i.e.*

$$(S') : \left\{ \begin{array}{rcll} x+y & = & 0 & (L'_1 \leftarrow L_1) \\ z & = & 0 & (L'_2 \leftarrow L_2 - 2.L_1) \end{array} \right.$$

L'explication fondamentale de l'équivalence est la même que dans l'exemple précédent.

(b) Etape de remontée, introduction de paramètres : Là encore, le gain de cet opération est que l'on se retrouve avec un système (S') *échelonné* : on peut le résoudre en « remontant ». Il y a cependant, comme dans les exemples avec une seule équation, le pas logique important [6] à passer : (x,y,z) est solution de (S') si et seulement si

$$\text{il existe } \lambda \in \mathbb{R}, \left\{ \begin{array}{rcl} x+y & = & 0 \\ y & = & \lambda \\ z & = & 0 \end{array} \right.$$

et, donc (x,y,z) est solution de (S') si et seulement si

$$(P_S) : \text{il existe } \lambda \in \mathbb{R}, \left\{ \begin{array}{rcl} x & = & -\lambda \\ y & = & \lambda \\ z & = & 0 \end{array} \right.$$

L'ensemble S des solutions de (S') (et donc de (S)) s'écrit donc

$$S = \{x \in \mathbb{R}^3, \exists \lambda \in \mathbb{R}, x = (-\lambda, \lambda, 0)\} = \{\lambda.(-1,1,0), \lambda \in \mathbb{R}\}$$

En posant $\Lambda = \mathbb{R}$, $\forall \lambda \in \Lambda$, $g(\lambda) = \lambda.(-1,1,0)$, on a donc obtenu la représentation paramétrique de S :

$$S = g(\Lambda) = \operatorname{Im} g$$

(c) Vocabulaire des systèmes échelonnés : on voit dans cet exemple que les variables x,y,z ne jouent pas exactement le même rôle dans la remontée :

i. z est fixé (valant zero). Il s'agit d'une *inconnue principale* du système échelonné.

ii. y n'est pas fixé par les valeurs des inconnues « suivantes », ici, la seule inconnue z, on doit introduire le paramètre λ quelconque représentant la valeur de y. y est une *inconnue secondaire* du système échelonné.

iii. x est fixé par les valeurs des inconnues « suivantes », ici, y et z. L'inconnue x est une *inconnue principale* du système échelonné.

5. Cette opération permet de raccourcir considérablement les phrases
6. pour effectuer la remontée, on dit que la variable y vaut quelque chose, que l'on nomme λ

(d) Interprétation géométrique : L'interprétation géométrique de la résolution est la suivante : on se place dans l'espace muni d'un repère $\mathscr{R} = (O, \vec{i}, \vec{j}, \vec{k})$ et on identifie [7] un point M de coordonnées (x, y, z) avec le triplet (x, y, z). M vérifie (S) si et seulement si il vérifie (L_1) et (L_2), *i.e.* M est dans l'intersection des plans L_1 et L_2. Comme O est dans chacun de ces plans, que ces plans ne sont pas confondus, l'intersection S est une droite dont (P_S) est une représentation paramétrique. Le vecteur $(-1, 1, 1)$ est un vecteur directeur de cette droite D.

3. Résoudre

$$(S) : \left\{ \begin{array}{rcll} x+y & = & 0 & (L_1) \\ 2x+2y+z & = & 0 & (L_2) \\ 3x+3y-z & = & 0 & (L_3) \end{array} \right. \text{ d'inconnue } (x, y, z) \in \mathbb{R}^3$$

On a

(a) Etape d'élimination : (x, y, z) satisfait (S) si et seulement si (x, y, z) satisfait $(S') = (L_1)$ et $(L_2) - 2(L_1)$ et $(L_3) - 3(L_1)$ *i.e.*

$$(S') : \left\{ \begin{array}{rcll} x+y & = & 0 & (L'_1 \leftarrow L_1) \\ z & = & 0 & (L'_2 \leftarrow L_2 - 2.L_1) \\ -z & = & 0 & (L'_3 \leftarrow L_3 - 3.L_1) \end{array} \right.$$

Ce système n'est pas échelonné, la preuve, c'est qu'on peut encore faire une élimination : (x, y, z) satisfait (S') si et seulement si (x, y, z) satisfait $(S'') = (L'_1)$ et (L'_2) et $(L'_3) + (L'_2)$ *i.e.*

$$(S'') : \left\{ \begin{array}{rcll} x+y & = & 0 & (L''_1 \leftarrow L'_1) \\ z & = & 0 & (L''_2 \leftarrow L'_2) \\ 0 & = & 0 & (L''_3 \leftarrow L'_3 + L'_2) \end{array} \right.$$

L'explication fondamentale du fait que les système (S), (S') et (S'') sont équivalents au sens où ils ont même ensemble de solutions est la même que dans l'exemple précédent, on traite d'abord l'équivalence de (S) et (S'), puis celle de (S') et (S'').

(b) Etape d'élimination des lignes inutiles : dans (S'') l'équation (L''_3) n'impose aucune contrainte sur la solution car elle est vérifiée quoi que soit la valeur de (x, y, z). On peut l'enlever.

(c) Etape de remontée : c'est la même que dans l'exemple précédent et on trouve que

$$S = \{\lambda.(-1, 1, 0), \lambda \in \mathbb{R}\} = \operatorname{Im} g$$

(d) Interprétation géométrique : L'interprétation géométrique de la résolution est la suivante : on se place dans l'espace muni d'un repère $\mathscr{R} = (O, \vec{i}, \vec{j}, \vec{k})$ identifié à $\mathbb{R}^3$ par ce moyen. M vérifie (S) si et seulement si il vérifie (L_1), (L_2) et (L_3), *i.e.* M est dans l'intersection des TROIS plans L_1, L_2 et L_3.

Si on prend trois plans passant par O, dans une situation « générique », on se convainc assez facilement par des dessins que leur intersection est réduite à O, ce qui n'est pas le cas dans notre exemple.

La situation géométrique dans notre cas est la suivante : l'intersection des plans L_1 et L_2 est une droite D. Cette droite D est contenue dans le plan L_3 et donc l'intersection S de L_1, L_2 et L_3 est cette droite D. On peut remarquer que les trois plans L_1, L_2 et L_3 sont deux à deux distincts.

7. Cette opération permet là encore de raccourcir considérablement les phrases

Résolution par élimination de GAUSS : systématisation

Pour résoudre un système linéaire homogène, on peut *poser* sa résolution par élimination de GAUSS. Prenons par exemple le système

$$(S):\left\{\begin{array}{rcll} y+z+t & = & 0 & (L_1)\\ x+y-2t & = & 0 & (L_2)\\ 2x+2y+z+t & = & 0 & (L_3)\\ 3x+3y+z-t & = & 0 & (L_4)\end{array}\right. \text{ d'inconnue } (x,y,z,t)\in\mathbb{R}^4$$

L'idée est de simplifier les écritures et d'avoir une feuille de calcul normalisée :

1. On commence par une phrase de principe : « résolvons le système (S) par la méthode du pivot de GAUSS ; Les systèmes représentés par les tableaux suivants sont tous équivalents [8] ».
2. On écrit le système (S) sous forme de tableau :l'inconnue x correspond à la première colonne, y la deuxième,..., Il n'est donc plus besoin d'écrire les inconnues. La dernière colonne est reservée au second membre, *i.e.* des zéros dans le cas homogène, les lignes du tableau représentent les lignes du système. Pour chaque ligne, on place les coefficients en facteur de chaque inconnue dans la colonne correspondante, on met un zéro lorsque l'inconnue n'apparait pas.

$$(S):\left\{\begin{array}{cccc|c} 0 & 1 & 1 & 1 & 0\\ 1 & 1 & 0 & -2 & 0\\ 2 & 2 & 1 & 1 & 0\\ 3 & 3 & 1 & -1 & 0\end{array}\right.$$

3. On écrit sans justification particulière, sans signe $\Leftrightarrow$, la suite de systèmes équivalents jusqu'à obtenir un système échelonné ou, notamment lorsqu'il y a des paramètres qui peuvent mener à discussion, jusqu'à l'étape où on commence la discussion. On indique les opérations faites à chaque étape d'une manière ou d'une autre. Les opérations autorisées (un type d'opération par étape) sont
   (a) Un échange de lignes ou plus généralement une permutation
   (b) Une multiplication de lignes par des constantes **non nulles**.
   (c) Après choix d'une ligne de travail L_t, modification de chacune des autres lignes par ajout de $c.L_t$

$$\left\{\begin{array}{cccc|c} 0 & 1 & 1 & 1 & 0\\ 1 & 1 & 0 & -2 & 0\\ 2 & 2 & 1 & 1 & 0\\ 3 & 3 & 1 & -1 & 0\end{array}\right. \xrightarrow{L_1\leftrightarrow L_2} \left\{\begin{array}{cccc|c} 1 & 1 & 0 & -2 & 0\\ 0 & 1 & 1 & 1 & 0\\ 2 & 2 & 1 & 1 & 0\\ 3 & 3 & 1 & -1 & 0\end{array}\right.$$

$$\xrightarrow[L_4\leftarrow L_4-3.L_1]{L_3\leftarrow L_3-2.L_1} \left\{\begin{array}{cccc|c} 1 & 1 & 0 & -2 & 0\\ 0 & 1 & 1 & 1 & 0\\ 0 & 0 & 1 & 5 & 0\\ 0 & 0 & 1 & 5 & 0\end{array}\right. \xrightarrow{L_4\leftarrow L_4-L_3} \left\{\begin{array}{cccc|c} 1 & 1 & 0 & -2 & 0\\ 0 & 1 & 1 & 1 & 0\\ 0 & 0 & 1 & 5 & 0\\ 0 & 0 & 0 & 0 & 0\end{array}\right.$$

Le dernier système est échelonné. Il y a une ligne inutile, une inconnue secondaire (t) et trois inconnues principales (x, y, z). On peut effectuer la remontée. Deux possibilités

8. Deux systèmes sont équivalents s'ils ont le même ensemble de solutions : la méthode consiste à passer d'un système d'équations à un autre, plus simple, en conservant l'ensemble des solutions

1. On repasse en notation complète : Le dernier système se réécrit

$$\left\{\begin{array}{rcl} x+y-2t & = & 0 \\ y+z+t & = & 0 \\ z+5t & = & 0 \\ 0 & = & 0 \end{array}\right. \text{ d'inconnue } (x,y,z,t)\in\mathbb{R}^4$$

On introduit maintenant les paramètres correspondant aux inconnues secondaires–ici, il n'y en a qu'une, t– : $(x,y,z,t)\in\mathbb{R}^4$ est solution de (S) si et seulement si

$$\text{il existe } \lambda\in\mathbb{R} \text{ tel que } \left\{\begin{array}{rcl} x+4\lambda-2\lambda & = & 0 \\ y-5\lambda+\lambda & = & 0 (y=4\lambda) \\ z & = & -5\lambda \\ t & = & \lambda \end{array}\right.$$

i.e.

$$\text{il existe } \lambda\in\mathbb{R} \text{ tel que } \left\{\begin{array}{rcl} x & = & -2\lambda \\ y & = & 4\lambda \\ z & = & -5\lambda \\ t & = & \lambda \end{array}\right.$$

Le système (S) admet une infinité de solutions et, en posant $u=\begin{pmatrix}-2\\4\\-5\\1\end{pmatrix}$, on a la représentation paramétrique de (S) :

$$S=\{\lambda.u, \lambda\in\mathbb{R}\}$$

2. On reste en notation tableau (variante dite de GAUSS–JORDAN) : on élimine les lignes devenues inutiles. Pour chaque ligne correspondant à une inconnue principale, on annule le reste de la colonne correspondant à l'inconnue.

$$\left\{\begin{array}{cccc|c} 1 & 1 & 0 & -2 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 5 & 0 \end{array}\right. \xrightarrow[\text{annul. col. } z]{L_2\leftarrow L_2-L_3} \left\{\begin{array}{cccc|c} 1 & 1 & 0 & -2 & 0 \\ 0 & 1 & 0 & -4 & 0 \\ 0 & 0 & 1 & 5 & 0 \end{array}\right.$$

$$\xrightarrow[\text{annul. col. } y]{L_1\leftarrow L_1-L_2} \left\{\begin{array}{cccc|c} 1 & 0 & 0 & +2 & 0 \\ 0 & 1 & 0 & -4 & 0 \\ 0 & 0 & 1 & 5 & 0 \end{array}\right.$$

On repasse en notation complète avec introduction des paramètres pour les inconnues secondaires. $(x,y,z,t)\in\mathbb{R}^4$ est solution de (S) si et seulement si

$$\text{il existe } \lambda\in\mathbb{R} \text{ tel que } \left\{\begin{array}{rcl} x+2\lambda & = & 0 \\ y-4\lambda & = & 0 \\ z+5\lambda & = & 0 \\ t & = & \lambda \end{array}\right. \quad \textit{i.e.} \quad \left\{\begin{array}{rcl} x & = & -2\lambda \\ y & = & 4\lambda \\ z & = & -5\lambda \\ t & = & \lambda \end{array}\right.$$

Ce qui donne la même conclusion qu'avec l'autre méthode.

Le cas général homogène/ Vocabulaire

Considérons le système

$$(S):\left\{\begin{array}{rcccl} a_{11}.x_1+\cdots+a_{1n}.x_n & = & 0 & (L_1) & \\ \vdots & & \vdots & \vdots & \vdots \\ a_{p1}.x_1+\cdots+a_{pn}.x_n & = & 0 & (L_p) & \end{array}\right. \text{ d'inconnue } (x_1,\ldots,x_n)\in\mathbb{K}^n$$

où les $p.n$ coefficients $a_{ij}, 1\leq i\leq p, 1\leq j\leq n$ sont fixés dans $\mathbb{K}$.

Le système (S) se représente sous la forme du tableau

$$\left\{\begin{array}{cccc|c} a_{11} & a_{21} & \ldots & a_{1n} & 0 \\ a_{21} & a_{22} & \ldots & a_{2n} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \ldots & a_{pn} & 0 \end{array}\right.$$

L'étape principale, *échelonnement par rapport à la première variable* est la suivante : il y a deux possibilités

1. $a_{11}=a_{21}=\cdots=a_{p1}=0$, *i.e.* toute la colonne correspondant à l'inconnue x_1 est nulle. x_1 sera une inconnue secondaire à l'issue de l'échelonnement et $(x_1,x_2,\ldots,x_n)$ est solution de (S) si et seulement si
$$\exists\lambda\in\mathbb{K}, x_1=\lambda \text{ et } \left\{\begin{array}{rcccl} a_{12}.x_2+\cdots+a_{1n}.x_n & = & 0 & (L_1) & \\ \vdots & & \vdots & \vdots & \vdots \\ a_{p2}.x_2+\cdots+a_{pn}.x_n & = & 0 & (L_p) & \end{array}\right.$$
i.e. $\exists\lambda\in\mathbb{K}, x_1=\lambda$ et $(x_2,\ldots,x_n)$ est solution du système
$$(S')\left\{\begin{array}{rcccl} a_{12}.x_2+\cdots+a_{1n}.x_n & = & 0 & (L_1) & \\ \vdots & & \vdots & \vdots & \vdots \\ a_{p2}.x_2+\cdots+a_{pn}.x_n & = & 0 & (L_p) & \end{array}\right. \text{ d'inconnue } (x_2,\ldots,x_n)\in\mathbb{K}^{n-1}$$
2. Il existe $k\in\{1,\ldots,p\}$ tel que $a_{k1}\neq 0$ *i.e.* la colonne correspondant à l'inconnue x_1 n'est pas nulle.
   (a) On échange s'il y a lieu les lignes 1 et k pour obtenir, sous forme de tableau,
$$\left\{\begin{array}{cccc|c} a_{11} & a_{21} & \ldots & a_{1n} & 0 \\ a_{21} & a_{22} & \ldots & a_{2n} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \ldots & a_{pn} & 0 \end{array}\right. \xrightarrow{L_1\leftrightarrow L_k} \left\{\begin{array}{cccc|c} a'_{11} & a'_{21} & \ldots & a'_{1n} & 0 \\ a'_{21} & a'_{22} & \ldots & a'_{2n} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a'_{p1} & a'_{p2} & \ldots & a'_{pn} & 0 \end{array}\right.$$
   où $a'_{11}\neq 0$
   (b) On conserve L_1 et on élimine, par combinaisons, l'inconnue x_1 des autres lignes
$$\left\{\begin{array}{cccc|c} a'_{11} & a'_{21} & \ldots & a'_{1n} & 0 \\ a'_{21} & a'_{22} & \ldots & a'_{2n} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a'_{p1} & a'_{p2} & \ldots & a'_{pn} & 0 \end{array}\right. \xrightarrow[L_k\leftarrow L_k-\frac{a'_{k1}}{a'_{11}}L_1]{L_2\leftarrow L_2-\frac{a'_{21}}{a'_{11}}L_1} \left\{\begin{array}{cccc|c} a'_{11} & a'_{21} & \ldots & a'_{1n} & 0 \\ 0 & a''_{22} & \ldots & a''_{2n} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a''_{p2} & \ldots & a''_{pn} & 0 \end{array}\right.$$

x_1 sera une inconnue principale à l'issue de l'échelonnement complet de (S) et $(x_1, x_2, \ldots, x_n)$ est solution de (S) si et seulement si $(x_2, \ldots, x_n)$ est solution du système de $p-1$ équations

$$(S') \begin{cases} a''_{22}.x_2 + \cdots + a''_{2n}.x_n &= 0 \\ \vdots & \vdots \\ a''_{p2}.x_2 + \cdots + a''_{pn}.x_n &= 0 \end{cases} \quad \vdots \quad \text{d'inconnue } (x_2, \ldots, x_n) \in \mathbb{K}^{n-1}$$

et

$$x_1 = -\left(\frac{a'_{12}}{a'_{11}}x_2 + \cdots + \frac{a'_{1n}}{a'_{11}}x_n\right)$$

3. On voit qu'on s'est ramenés dans les deux cas à résoudre un système de p ou $p-1$ équations en $n-1$ inconnues scalaires (*i.e.* une inconnue dans $\mathbb{K}^{n-1}$). Si on applique la **même** méthode [9] à ce nouveau système en l'échelonnant par rapport à la variable x_2 , on sera amenés à résoudre un système de p, $p-1$ ou $p-2$ équations à $n-2$ inconnues scalaires et, de proche en proche, après élimination de lignes inutiles, à un système n'ayant plus qu'une seule équation (qui peut-être $0=0$).
   Comme on sait traiter ce cas ultime, on a résolu le système.

D'un point de vue pratique, on sépare l'étape d'échelonnement et l'étape de résolution :

1. Echelonnement : on peut utiliser pour résoudre les systèmes successifs le même tableau, sans toucher aux lignes déjà traitées. Le tableau final sera celui d'un système échelonné, *i.e.* un système qui ne change pas lorsque l'on applique l'algorithme d'échelonnement juste décrit.
2. Résolution du système échelonné : On repère inconnues secondaires et inconnues principales. On introduit un paramètre par inconnue secondaire et on effectue la remontée.

A l'issue de l'échelonnement, il reste un certain nombre d'équations utiles, une par inconnue principale.

Le nombre de ces équations et donc le nombre d'inconnues principales s'appelle le *rang du système échelonné*.

Le nombre de paramètres introduits est le nombre d'inconnues secondaires.

On a les relations, pour un système échelonné (une version primitive du théorème du rang) :

$$\text{nbre de paramètres} + \text{rang du système} = \text{nbre d'inconnues} = n$$

et

$$\text{nbre d'équations devenues inutiles} + \text{rang du système} = \text{nbre d'équations initiales} = p$$

Systèmes non homogènes

Considérons le système

$$(S) : \begin{cases} a_{11}.x_1 + \cdots + a_{1n}.x_n &= b_1 \quad (L_1) \\ \vdots & \vdots \quad \vdots \quad \vdots \\ a_{p1}.x_1 + \cdots + a_{pn}.x_n &= b_p \quad (L_p) \end{cases} \quad \text{d'inconnue } (x_1, \ldots, x_n) \in \mathbb{K}^n$$

où les $p.n$ coefficients $a_{ij}, 1 \leq i \leq p, 1 \leq j \leq n$, ainsi que les p coefficients b_i, $1 \leq i \leq p$ sont fixés dans $\mathbb{K}$.

Le système (S) se représente sous la forme du tableau

$$\left\{\begin{array}{cccc|c} a_{11} & a_{21} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & \ldots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \ldots & a_{pn} & b_p \end{array}\right.$$

9. Il s'agit donc d'une méthode *récursive*

On applique le même principe d'échelonnement. Pour l'étape principale, *échelonnement par rapport à la première variable*, il y a deux possibilités

1. $a_{11} = a_{21} = \cdots = a_{p1} = 0$, *i.e.* toute la colonne correspondant à l'inconnue x_1 est nulle. x_1 sera une inconnue secondaire à l'issue de l'échelonnement et $(x_1, x_2, \ldots, x_n)$ est solution de (S) si et seulement si
$$\exists \lambda \in \mathbb{K}, x_1 = \lambda \text{ et } \left\{ \begin{array}{rcccc} a_{12}.x_2 + \cdots + a_{1n}.x_n & = & b_1 & (L_1) \\ \vdots & & \vdots & \vdots & \vdots \\ a_{p2}.x_2 + \cdots + a_{pn}.x_n & = & b_p & (L_p) \end{array} \right.$$
i.e. $\exists \lambda \in \mathbb{K}, x_1 = \lambda$ et $(x_2, \ldots, x_n)$ est solution du système
$$(S') \left\{ \begin{array}{rcccc} a_{12}.x_2 + \cdots + a_{1n}.x_n & = & b_1 & (L_1) \\ \vdots & & \vdots & \vdots & \vdots \\ a_{p2}.x_2 + \cdots + a_{pn}.x_n & = & b_p & (L_p) \end{array} \right. \text{ d'inconnue } (x_2, \ldots, x_n) \in \mathbb{K}^{n-1}$$

2. Il existe $k \in \{1, \ldots, p\}$ tel que $a_{k1} \neq 0$ *i.e.* la colonne correspondant à l'inconnue x_1 n'est pas nulle.
   (a) On échange s'il y a lieu les lignes 1 et k pour obtenir, sous forme de tableau,
$$\left\{ \begin{array}{cccc|c} a_{11} & a_{21} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & \ldots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \ldots & a_{pn} & b_p \end{array} \right. \xrightarrow{L_1 \leftrightarrow L_k} \left\{ \begin{array}{cccc|c} a'_{11} & a'_{21} & \ldots & a'_{1n} & b'_1 \\ a'_{21} & a'_{22} & \ldots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a'_{p1} & a'_{p2} & \ldots & a'_{pn} & b'_p \end{array} \right.$$
   où $a'_{11} \neq 0$
   (b) On conserve L_1 et on élimine, par combinaisons, l'inconnue x_1 des autres lignes
$$\left\{ \begin{array}{cccc|c} a'_{11} & a'_{21} & \ldots & a'_{1n} & b'_1 \\ a'_{21} & a'_{22} & \ldots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a'_{p1} & a'_{p2} & \ldots & a'_{pn} & b'_p \end{array} \right. \begin{array}{l} L_2 \leftarrow L_2 - \frac{a'_{21}}{a'_{11}} L_1 \\ L_k \leftarrow L_k - \frac{a'_{k1}}{a'_{11}} L_1 \end{array} \longrightarrow \left\{ \begin{array}{cccc|c} a'_{11} & a'_{21} & \ldots & a'_{1n} & b'_1 \\ 0 & a''_{22} & \ldots & a''_{2n} & b'_2 - \frac{a'_{21}}{a'_{11}} b'_1 = b''_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a''_{p2} & \ldots & a''_{pn} & b'_p - \frac{a'_{p1}}{a'_{11}} b'_1 = b''_p \end{array} \right.$$
   x_1 sera une inconnue principale à l'issue de l'échelonnement complet de (S) et $(x_1, x_2, \ldots, x_n)$ est solution de (S) si et seulement si $(x_2, \ldots, x_n)$ est solution du système de $p-1$ équations
$$(S') \left\{ \begin{array}{rcccc} a''_{22}.x_2 + \cdots + a''_{2n}.x_n & = & b''_2 \\ \vdots & & \vdots & \vdots \\ a''_{p2}.x_2 + \cdots + a''_{pn}.x_n & = & b''_p \end{array} \right. \text{ d'inconnue } (x_2, \ldots, x_n) \in \mathbb{K}^{n-1}$$
   et
$$x_1 = b'_1 - \left(\frac{a'_{12}}{a'_{11}} x_2 + \cdots + \frac{a'_{1n}}{a'_{11}} x_n \right)$$

3. Comme dans le cas homogène, on voit qu'on s'est ramenés dans les deux cas à résoudre un système de p ou $p-1$ équations en $n-1$ inconnues scalaires (*i.e.* une inconnue dans $\mathbb{K}^{n-1}$). Si on applique la méthode à ce nouveau système en l'échelonnant par rapport à la variable x_2 , on sera amenés à résoudre un système de p, $p-1$, ou $p-2$ équations à $n-2$ inconnues scalaires.

4. La présence de coefficients b_i possiblement non nuls induit la possibilité d'un nouveau phénomène. On peut se retrouver, après un certain nombre d'étapes à un système composé de lignes du type

$$0 = b_i$$

i.e. un système où toutes les inconnues ont disparu.

De telles lignes sont appelées des *lignes de compatibilité*. De deux choses l'une

(a) Soit tous les b_i sont nuls, auquel cas ces lignes d'équations sont vraies, indépendamment des valeurs de l'inconnue : elles sont inutiles et on peut les ignorer.

(b) Soit l'un des b_i est *non nul* ; dans ce cas la ligne d'équation corresponante est fausse, indépendamment des valeurs de l'inconnue : le système n'a pas de solution, il est dit *incompatible*.

Lorsque le second membre comporte des paramètres, la présence de ce type de ligne implique une discussion sur les paramètres : il y a des cas où le système n'a pas de solution et d'autres où il y a des solutions.

A l'issue de l'échelonnement, imaginons que nous ayons conservé toutes les lignes. La présence d'équations de compatibilité (vraies ou fausses) est possible.

Le *rang du système échelonné* est encore le nombre d'inconnues principales. C'est aussi le rang du système homogène échelonné obtenu en remplaçant les b_i par 0

Le nombre de paramètres introduit est le nombre d'inconnues secondaires.

On a les relations, pour un système échelonné (une version primitive du théorème du rang) :

$$\text{nbre de paramètres} + \text{rang du système} = \text{nbre d'inconnues} = n$$

et

$$\text{nbre d'équations de compatibilité} + \text{rang du système} = \text{nbre d'équations initiales} = p$$

et donc

$$\text{nbre d'équations de compatibilité} - \text{nbre de paramètres} = p - n$$

Implémentation informatique

Utiliser le script `python/Gauss.py`.

Les fonctions Python `Gauss(A,B)` et `Jordan(E,C)` présentées dans le script implémentent informatiquement respectivement l'étape d'échelonnement et l'étape de remontée pour un système échelonné.

Il y a deux fonctions complémentaires :

1. `Echelonnement(E)` qui retourne si la matrice `E` est échelonnée ou pas, ainsi que les listes d'inconnues principales, secondaires et la liste des équations de compatibilité. La connaissance d'une liste d'inconnues principales donne une base de l'image d'une matrice (extraction de base d'une famille génératrice).
2. `KerIm(A)` qui retourne, pour la matrice `A`, une base de son noyau, un système d'équations cartésiennes de son image.

Noter que les matrices de second membre `B` et `C` peuvent comporter plusieurs colonnes, ce qui permet la résolution simultanée de système ayant même membre de gauche, *i.e.* même matrice `A` ou `E`.

La programmation de l'algorithme de GAUSS ou de GAUSS–JORDAN n'est pas stricto sensu au programme de BCPST2, néanmoins, la fonction de recherche des inconnues principales et secondaires d'un système échelonné tombe en plein dans le programme d'informatique.

Le théorème du rang

On peut démontrer le point fondamental suivant :

Deux systèmes échelonnés ayant même ensemble de solutions ont même rang.

En conséquence, on peut définir le *r*ang d'un système linéaire homogène comme étant le rang d'un système échelonné ayant même ensemble de solutions. L'algorithme de GAUSS nous fournit l'existence d'un tel système échelonné, le point sus-mentionné, garantit que le rang trouvé ne dépend pas du système échelonné (et donc, *a fortiori* , de la méthode d'échelonnement [10]).

Ce rang a d'autres interprétations que nous ne rappelerons pas ici, à l'exception du théorème du rang :

Théorème 1. *Si $f : \mathbb{K}^n \to \mathbb{K}^p$ est une application linéaire, alors*

$$\dim Ker\, f + rg\, f = n$$

Rappelons que Ker f est l'ensemble donné de façon cartésienne par

$$\text{Ker}\, f = \{x \in \mathbb{K}^n,\, f(x) = 0\}$$

Il s'agit donc de l'ensemble des solutions d'un certain système linéaire homogène, de rang r au sens précédent. Sa dimension est le nombre minimal de paramètres scalaires nécessaires à une description paramétrique linéaire de cet ensemble.

Le rang de f, rg$f$ est défini comme la dimension de Im f, *i.e.* le nombre minimal de paramètres scalaires nécessaires à une description paramétrique linéaire de cet ensemble.

On a rg$f = r$.

Si $A \in \mathscr{M}_{p,n}(\mathbb{K})$ et f est l'application linéaire définie par

$$\forall x \in \mathbb{K}^n,\, f(x) = A.x$$

où le produit $A.x$ est compris comme la mulitplication de la matrice A par le vecteur x représenté sous forme de matrice colonne.

Le noyau de la matrice A est le noyau de f, le rang de la matrice A est le rang de f et

$$\dim \text{Ker}\, A + \dim \text{Im}\, A = n$$

Après échelonnement de A,

1. dim Ker A est le nombre d'inconnues secondaires
2. dim Im A est le nombre d'inconnues principales

Discussion de systèmes à paramètres

L'exercice est le suivant : étant donné un paramètre $m \in \mathbb{R}$ et la matrice

$$M = \begin{pmatrix} 1 & 1 & 1 & m \\ 1 & 1 & m & 1 \\ 1 & m & 1 & 1 \\ m & 1 & 1 & 0 \end{pmatrix}$$

Déterminer, en discutant suivant les valeurs de m, le rang de la matrice M.

10. On peut remarquer qu'il y a plusieurs choix au moment de l'échange des lignes 1 et k

Pour cela on entame la résolution du système linéaire homogène de matrice M, que l'on écrit sous forme de tableaux.

Par application de l'algorithme de GAUSS, les tableaux suivants, représentent des systèmes linéaire équivalents et donc, de même rang :

$$\left\{\begin{array}{cccc|c} 1 & 1 & 1 & m & 0 \\ 1 & 1 & m & 1 & 0 \\ 1 & m & 1 & 1 & 0 \\ m & 1 & 1 & 0 & 0 \end{array}\right. \xrightarrow[L_3 \leftarrow L_3 - L_1]{L_2 \leftarrow L_2 - L_1 \\ L_4 \leftarrow L_4 - m.L_1} \left\{\begin{array}{cccc|c} 1 & 1 & 1 & m & 0 \\ 0 & 0 & m-1 & 1-m & 0 \\ 0 & m-1 & 0 & 1-m & 0 \\ 0 & 1-m & 1-m & -m^2 & 0 \end{array}\right.$$

$$\xrightarrow{L_3 \leftrightarrow L_2} \left\{\begin{array}{cccc|c} 1 & 1 & 1 & m & 0 \\ 0 & m-1 & 0 & 1-m & 0 \\ 0 & 0 & m-1 & 1-m & 0 \\ 0 & 1-m & 1-m & 0-m^2 & 0 \end{array}\right. \xrightarrow{L_4 \leftarrow L_4 + L_2} \left\{\begin{array}{cccc|c} 1 & 1 & 1 & m & 0 \\ 0 & m-1 & 0 & 1-m & 0 \\ 0 & 0 & m-1 & 1-m & 0 \\ 0 & 0 & 1-m & 1-m-m^2 & 0 \end{array}\right.$$

$$\xrightarrow{L_4 \leftarrow L_4 + L_3} \left\{\begin{array}{cccc|c} 1 & 1 & 1 & m & 0 \\ 0 & m-1 & 0 & 1-m & 0 \\ 0 & 0 & m-1 & 1-m & 0 \\ 0 & 0 & 0 & 2-2m-m^2 & 0 \end{array}\right.$$

Le système final a l'air échelonné : il faut se méfier des annulations !

1. Si $m-1 \neq 0$ et $m^2+2m-2 \neq 0$, alors le système est échelonné et son rang est 4.
2. Si $m^2-2m-2=0$, *i.e.* $m = 1 \pm \sqrt{3}$, le système est échelonné et son rang est 3.
3. Si $m = 1$, le système N'est PAS échelonné, il faut encore faire une opération de l'algorithme : représenté par le tableau

$$\left\{\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{array}\right. \xrightarrow{L_2 \leftrightarrow L_4} \left\{\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right.$$

Le système est de rang 2.

Exercices

Exercice 1.—On considère les systèmes suivants :

$$\text{(I)} \left\{\begin{array}{rcl} 2x+3y-z & = & -2 \\ x+2y+5z & = & 5 \\ 5x+8y+3z & = & 1 \end{array}\right. \qquad \text{(II)} \left\{\begin{array}{rcl} 3x-2y+2z & = & \lambda x \\ 3x-2y+3z & = & \lambda y \\ 2x-2y+3z & = & \lambda z \end{array}\right. (\lambda \text{ est un paramètre}).$$

1. Donner l'écriture matricielle de ces systèmes. Les interpréter en termes d'application linéaires.
2. Résoudre ces deux systèmes en utilisant la méthode du pivot de GAUSS (pour (II), commencer par échanger la ligne 3 et la ligne 1).

Exercice 2.—Soit $\lambda \in \mathbb{R}$. Appliquer la méthode du pivot de GAUSS pour résoudre le système d'inconnue $(x,y,z) \in \mathbb{R}^3$,

$$\left\{\begin{array}{rcrcrcl} (1-\lambda)x & & -y & & +z & = & 0 \\ -x & & +(1-\lambda)y & & -z & = & 0 \\ x & & -y & & +(1-\lambda)z & = & 0 \end{array}\right.$$

Exercice 3.—En utilisant la méthode du pivot de GAUSS, déterminer si les matrices suivantes sont inversibles et si c'est le cas, calculer leur inverse, sinon, donner leur rang :

$$A = \begin{pmatrix} 2 & 5 \\ 1 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 3 & -1 \\ 3 & -1 & 7 \end{pmatrix}.$$

Comment interpréter ceci sur les applications linéaires $\phi_A : \mathbb{R}^2 \to \mathbb{R}^2, X \mapsto A.X$ et $\phi_B : \mathbb{R}^3 \to \mathbb{R}^3, X \mapsto B.X$?

Exercice 4.—Résoudre les systèmes suivants d'inconnue $(x,y,z,t) \in \mathbb{R}^4$. Pour chacun de ces systèmes, on précisera la matrice du système, son rang et une base de son noyau.

$$\begin{cases} 2x+y+3z=1 \\ x+y+z=0 \\ x+2y+z=1 \end{cases} \quad \begin{cases} 2x+y+z+t=0 \\ 3x+y+z+t=1 \\ 2x+y+2z+2t=2 \\ 3x+y+2z+t=-1 \end{cases} \quad \begin{cases} 2x+y+z+t=0 \\ 3x+y+z+t=1 \\ 2x+y+2z+2t=2 \\ 3x+y+2z+2t=-1 \end{cases} \quad \begin{cases} x-2y+z+2t=0 \\ 2x+y+z+t=1 \\ x+y+2z+t=2 \end{cases}$$

Exercice 5.—Rappeler la formule du déterminant d'un système linéaire 2×2 et la formule de l'inverse d'une matrice 2×2.

Soit $\alpha, \beta \in \mathbb{R}$. Résoudre à l'aide de ces formules le système d'inconnue $(x,y) \in \mathbb{R}^2$.

$$\begin{cases} e^{\alpha}x & +e^{\beta}y & = & 1 \\ -e^{-\beta}x & +e^{-\alpha}y & = & 1 \end{cases}$$

Exercice 6.—Rappeler la formule du déterminant d'un système linéaire 2×2 et la formule de l'inverse d'une matrice 2×2.

Soit $f(x) = \lambda.e^x \cos x + \mu.e^x \sin x$. Soit $x_0 \in \mathbb{R}$. Déterminer $\lambda, \mu \in \mathbb{R}$ pour que

$$f(x_0) = 1 \text{ et } f'(x_0) = 1$$

Exercice 7.—Soit $m \in \mathbb{R}$ un réel fixé. Résoudre les systèmes suivants d'inconnue $(x,y,z) \in \mathbb{R}^3$ en discutant éventuellement suivant les valeurs du paramètre m. Préciser la matrice du système, son rang et une base de son noyau.

$$\begin{cases} x+y+mz=m \\ x+my-z=1 \\ x+y-z=1 \end{cases} \quad \begin{cases} x+y+z=1 \\ mx+y+z=m \\ x+my+z=1 \\ x+y+mz=m \end{cases} \quad \begin{cases} (m-1)x+my+z=m+1 \\ mx+2y+3z=3 \\ (m+1)x+my+(m-1)z=m-1 \end{cases}$$

Exercice 8.—Dans $\mathbb{R}^3$, considérons les points $A=(1,2,3)$ et $B=(2,4,8)$. Donner un système d'équations linéaires avec second membre admettant tous les points de la droite AB, et eux seuls, comme solutions.

Exercice 9.—Peut-on déterminer des réels x,y pour que le vecteur $v=(-2,x,y,3)$ appartienne au s.e.v. de $\mathbb{R}^4$ engendré par la famille (e_1,e_2) où $e_1=(1,-1,1,2)$ et $e_2=(-1,2,3,1)$?

Exercice 10.—Soient $e_1=(0,1,-2,1), e_2=(1,0,2,-1), e_3=(3,2,2,-1)$ et $e_4=(0,0,1,0)$ des vecteurs de $\mathbb{R}^4$. Les propositions suivantes sont-elles vraies ou fausses ? Justifier votre réponse.

1. $\text{Vect}\langle e_1,e_2,e_3\rangle = \text{Vect}\langle (1,1,0,0),(-1,1,-4,2)\rangle$.
2. $(1,1,0,0) \in \text{Vect}\langle e_1,e_2\rangle \cap \text{Vect}\langle e_2,e_3,e_4\rangle$.

Exercice 11.—On considère la matrice :

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

Calculer M^2 et montrer qu'on peut écrire M^2 comme combinaison linéaire de M et de I_3. En déduire que M est inversible et donner son inverse.

Exercice 12.—Pour la matrice suivante, calculer A^3 et en déduire que A n'est pas inversible.

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

Exercice 13.—Calculer A^n avec :

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Exercice 14.—Calculer, pour $n \geq 2$, A^n avec :

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{pmatrix}.$$

Exercice 15.—Pour $A \in \mathscr{M}_n(\mathbb{K})$, on appelle trace de A la somme des éléments diagonaux de A, c'est-à-dire :

$$\text{Si } A = \left(a_{i,j}\right)_{1 \leq i,j \leq n} \text{ alors } \mathrm{Tr}(A) = \sum_{i=1}^{n} a_{i,i}.$$

1. Montrer que pour toutes les matrices A et B carrées de taille n et tous les scalaires λ, $\mathrm{Tr}(\lambda A + B) = \lambda \mathrm{Tr}(A) + \mathrm{Tr}(B)$.
2. Montrer que pour toutes les matrices A et B carrées de taille n, $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$. Indication:Appeler $C = A.B$, $D = B.A$ et écrire les formules abstraites de produit matriciel donnant les coefficients C_{ij} et D_{ij}
3. En déduire qu'il n'existe pas de matrices A et B carrées de taille n telles que $AB - BA = I_n$.

Lycée Pierre-Gilles de Gennes **BCPST2**
Mathématiques **2019-2020**

Révisions/Formulaire 05bis

Algèbre Linéaire en Python

1 Listes, vecteurs et matrices

Un type d'objet fondamental en Python est la *liste*, du type `list`. Une telle liste se construit, par exemple en la nommant avec une construction du type

`MaListe = [` obj_1, obj_2,...,obj_n`]`

où obj_1, obj_2,...,obj_n sont des objets déjà existants. Ces objets peuvent être de tous types et de types variés. Certains peuvent être eux-mêmes des listes. La liste la plus simple est la liste vide, déclarée par

`MaListe = []` ou `MaListe = list()`

Un vecteur [11] ou une matrice peuvent être vus comme des listes (au sens commun du terme) de nombres ayant une *forme* (le nombre de composantes pour un vecteur, la taille de la matrice pour une matrice).

Quelques opérations simples sur les vecteurs et matrices peuvent être codées en Python mais certaines opérations déjà existantes (`builtin`) peuvent être déroutantes. Par exemple, la somme de deux listes `liste1+liste2` est simplement leur *concaténation* et le produit d'un entier `n` (de type `int`) par une liste `Maliste` donne une liste contenant les éléments de `MaListe` répétés `n` fois.

Ce n'est pas le comportement attendu pour des vecteurs ou des matrices.

Pour utiliser des vecteurs ou matrices en Python, le mieux est de se baser sur la bibliothèque `numpy` [12] qui définit un type très général d'objet : le tableau multi-dimensionnel de nombres de type `ndarray`.

Pour l'inversion de matrices (et d'autes opérations plus avancées, comme le calcul de valeurs propres), on utilisera la bibliothèque `scipy.linalg` [13]

Des documents assez complets (en anglais) se trouvent aux adresses

`http://wiki.scipy.org/Tentative_NumPy_Tutorial`

et `http://nbviewer.ipython.org/github/jrjohansson/scientific-python-lectures/blob/master/Lecture-2-Numpy.ipynb`

Les vecteurs sont représentés par des `ndarray` de dimension 1, les matrices par des `ndarray` de dimension 2, etc...les objets de dimension supérieure à 2 ne sont pas à oublier, ils peuvent par exemple servir à coder les valeurs d'une fonction de trois variables réelles discrétisée sur une grille parallélépipèdique.

Ces objets se comportent vis à vis des opérations usuelles de la façon mathématiquement attendue.

- si `V` est un `ndarray`, `a` est un `float` alors `a*V` est un `ndarray` dont les composantes sont celles de `V` chacune multipliée par la valeur de `a`.
- Si `V` et `W` sont deux `ndarray` de même forme (*i.e.* deux vecteurs de même longueur, deux matrices de même taille) alors `V+W` est un `ndarray` contenant la somme composante par composante de `V` et `W`.
- En bref, pour une forme de `ndarray` donnée, l'ensemble des `ndarray` muni de ces deux opérations se comporte comme un espace vectoriel dont le corps des scalaires est celui des `float` [14]

11. dans tout ce texte un vecteur est à prendre au sens de n-uplet dans $\mathbb{R}^n$
12. *Num*erical *Py*thon, à importer en début de script par `import numpy as np`
13. à importer en début de script par `import scipy.linalg as linalg`
14. On peut aussi faire des `ndarray` de `complex`, on travaille alors dans $\mathbb{C}^n$ ou $\mathscr{M}_{p,n}(\mathbb{C})$.

On déclare un `ndarray` par [15]

— `V = np.zeros(3)` : construit et désigne par `V` un vecteur de longueur 3 rempli de zéros
— `M = np.ones((3,2))` : construit, désigne par `M` une matrice de 3 lignes, 2 colonnes pleine de 1
— `M = np.random.rand((3,2))` : construit, désigne par `M` une matrice de 3 lignes et 2 colonnes remplie de nombres tirés au hasard uniformément sur $]0,1[$, indépendamment.
— `V=np.array([0.1,-2.3,3.4])` : prend la liste Python `[0.1,-2.3,3.4]`, construit le `ndarray` unidimensionnel contenant les éléments de cette liste et le désigne par `V`
— `M=np.array([[0.1,-2.3,3.4],[0.2,0.3,-np.pi]])` : prend la liste Python `[[0.1,-2.3,3.4],[0.2,0.3,-np.pi]]`, construit le `ndarray` 2-dimensionnel (*a.k.a* la matrice 2×3) contenant les éléments de cette liste et le désigne par `M`. Agit ligne par ligne.
— On peut finalement déclarer un `ndarray` comme simple vecteur et lui donner sa forme, *a posteriori* .

```
V=np.array([i for i range(15)]); V.reshape(3,5)
```

commence par fabriquer un vecteur de 15 composantes contenant les entiers de 0 à 14 et transforme ce vecteur en matrice 3×5.
— `np.arange(1,15)` fabrique un `ndarray` de 14 composantes `[1,2,...,14]`
— `np.linspace(-1,3.14,143)` fabrique un `ndarray` de 143 nombres flottants `float` équidistribués de -1 à 3.14, bornes comprises.
— `np.diag()`

1. Si `A` est une matrice (`ndarray` 2-dimensionnel), `np.diag(A)` fabrique un vecteur (`ndarray` 1-dimensionnel) dont les composantes sont les éléments diagonaux de `A`
2. Si `V` est un vecteur, `np.diag(V)` fabrique une matrice diagonale dont la diagonale est composée des composantes de `V`.

Utiliser le script `exemple-fil-rouge.py`.

Dans ce fichier, on voit comment se développe l'exemple de l'interpolation polynomiale, ce qui illustre bon nombre des opérations indiquées dans ce document.

2 Forme, accès aux composantes et slicing

La forme d'un `ndarray` `V` s'obtient grâce à l'attribut `V.shape` qui retourne un t-uple de nombres entiers : on récupère donc

— Le nombre de dimensions (*i.e.* identifier si on a affaire à un vecteur ou une matrice) par `len(V.shape)`
— le nombre de lignes par `V.shape[0]` (première composante de `V.shape`)
— le nombre de colonnes par `V.shape[1]` (deuxième composante de `V.shape`)
— le nombre de « tranches [16] » par `V.shape[2]` (troisième composante de `V.shape`)
— ....

On accède à une composante donnée d'un `ndarray` avec la syntaxe d'accès aux éléments d'une liste ou avec une syntaxe particulière aux `ndarray` :

— `V[i,j]` si `V` est une matrice donne la valeur de l'entrée indicée (i,j) dans la matrice `V`
— `V[i]` si `V` est un vecteur donne la valeur de la composante indicée i du vecteur `V`
— Il faut rappeler encore et encore que les indices commencent à 0. Dans le premier exemple, `i` doit être compris au sens large entre 0 et `V.shape[0]-1`, `j` doit être compris au sens large entre 0 et `V.shape[1]-1`

15. liste d'exemples non exhaustive
16. Comme on traite prioritairement vecteurs et matrices, ce cas ne devrait pas se présenter

— Une astuce utile est l'utilisation des indices négatifs, qui permettent de partir de la fin. Si `V` est un vecteur (un `ndarray` unidimensionnel), `V[-1]` désigne sa dernière composante, `V[-2]`, son avant-dernière composante.

Partant d'un matrice, on peut en extraire des sous-matrices, des vecteurs colonnes ou des lignes.

Il s'agit de la syntaxe de tranchage [17]. Par exemple, on peut accèder à la première colonne de la matrice `V` avec la syntaxe `V[:,0]`. Le `:` signifie « tous les indices dans la dimension considérée ».

La commande `V[1:3,2:]` permet de récupérer les élements des lignes indicées 1 et 2, aux colonnes à partir de celle indicée 2. `1:3` signifie « tous les indices i, $1 \leq i < 3$ », `2:` signifie « tous les indices j, $2 \leq j$ », `:-1` signifie « tous les indices sauf le dernier ».

3 Construction avancée de vecteurs ou matrices

Avant de manipuler des vecteurs ou matrices, il s'agit d'en construire. On a vu

— Comment construire une matrice/vecteur remplie de zeros.

— Comment utiliser une liste Python pour la transformer en vecteur/matrice.

Si on veut une matrice avec des entrées particulières (par exemple dépendant de l'indice de l'entrée), on peut partir d'une matrice nulle, et la remplir avec la formule souhaitée par deux boucles imbriquées. C'est la méthode de base, très robuste mais aussi *très lente* sur de *grosses matrices* [18]

Il peut être intéressant de ce point de vue de construire des vecteurs ou matrices comme s'il s'agissait de briques d'un jeu de construction. Les opérations sont alors confiées à des boucles internes invisibles bien plus rapides qu'une boucle `for`. Partant de deux matrices avec des dimensions compatibles (ça dépend des opérations), on peut les empiler, les juxtaposer, éliminer une colonne, placer l'une dans l'autre à un endroit précis.

A regarder dans le script, concernant les constructions, la fabrication d'une matrice de VANDERMONDE (lignes 14 à 32) à partir d'un vecteur par construction ligne à ligne et empilement vertical. On remarque qu'en fin de fonction, on retourne la matrice transposée de celle fabriquée, ce qui est la matrice de VANDERMONDE classique. Voici quelques opérations possibles

— Empiler verticalement : Si `A` et `B` sont deux `ndarray`, `np.vstack(A,B)` retourne le `ndarray` où on a placé `A` au-dessus de `B`

— Juxtaposer horizontalement : Si `A` et `B` sont deux `ndarray`, `np.hstack(A,B)` retourne le `ndarray` où `B` est placé à droite de `A`. **Cette méthode est utile pour tester si une famille de vecteurs est libre et/ou composer une matrice de passage par juxtaposition de vecteurs**

— `A=A[:,1:]` : Elimine la première colonne de la matrice `A` / `A=A[:,:-1]` : Elimine la dernière colonne de la matrice `A`

— `A=A[1:,:]` : Elimine la première ligne de la matrice `A` / `A=A[:-1,:]` : Elimine la dernière ligne de la matrice `A`

— `A=A[1:-1,1:-1]` : Elimine première et dernière ligne ainsi que première et dernière colonne [19] de la matrice `A`

— `V=V[2:6]` : Elimine les éléments indicés 0, 1, 6, 7, ... du vecteur `V`.

17. slicing en anglais

18. Il faut toujours penser qu'une matrice 1000×1000, qui selon les standards actuels n'est pas *très* grosse, contient 10^6 éléments et donc que la double boucle fait un million de tours pour la seule construction de la matrice. Une machine actuelle cadencée à 2GHz a un processeur qui effectue 2 milliards d'opérations très élémentaires en 1s. Python étant un langage interprété, un million d'opérations Python représente plusieurs dizaines de millions d'opérations processeur–Pour un langage compilé, comme `C` ou `C++`, le facteur est plutot de 1 à 10 que de 1 à 100 : un langage bien compilé est probablement 10 fois plus rapide qu'un langage interprété– Le remplissage de la matrice prendrait donc un centième de seconde, ce qui est beaucoup du point de vue d'une machine.

19. Si la matrice était $N \times N$, elle devient $N-2 \times N-2$

4 Opérations matricielles

— La multiplication matrice×matrice ou matrice×vecteur se fait via l'opération `np.dot(A,B)` ou `np.dot(A,V)`. L'utilisation de cette commande sur un couple de vecteurs donne leur *produit scalaire*, un nombre donc. Dans le script d'exemple voir la ligne 86 pour une utilisation de ce produit. **Le signe de multiplication `*` entre deux `ndarray` ne calcule pas un produit matriciel !!!**
— La transposée d'une matrice `A` s'obtient par `A.transpose()`.
— L'inverse d'une matrice `A` s'obtient [20] par `linalg.inv(A)`.Dans le script d'exemple, voir la ligne 85 pour l'utilisation de l'inversion.

5 Application d'une fonction à un `ndarray`

Les `ndarray` servent à représenter des vecteurs ou des matrices mais ils servent aussi à représenter des fonctions de une ou plusieurs variables. Ceci explique le comportement (non décrit par l'algèbre linéaire abstraite) de plusieurs opérateurs et fonctions.

— Multiplication, division `*` et `/` : ces opérations, placées entre deux `ndarray` de mêmes tailles, effectuent la multiplication ou la division **composante par composante**. Il ne s'agit pas de multiplication ou d'inversion matricielle ! !
— Mise à la puissance `**` : Si `A` est une matrice, `A**2` ne calcule pas le carré de la matrice `A`, il compose la matrice dont les composantes sont les composantes de `A` chacune mise au carré. Il en est de même des autres puissances.
— Fonctions algébriques et transcendantes. Les fonctions `np.sqrt()`,`np.cos()`, `np.exp()`,..., agissent aussi composantes par composantes ;
— Dans ce contexte, des opérations courantes sont `x=np.linspace(0,np.pi,100)`, *i.e.* la création d'un vecteur et le calcul des images de chacune des composantes par une fonction donnée `y=np.cos(x)`, l'application d'opérations algèbriques composante par composante du type `z=x*y`.

6 Sauvergarde sur disque, chargement, affichage

Quand le calcul d'une matrice à 1000000 d'entrées à pris 3 jours, il ne faut pas la perdre ! Sinon, a la session suivante, il faudra la recalculer et donc perdre 3 jours...

On peut sauver sur disque (au sens large : clé USB, réseau,...) un `ndarray` et le recharger

— En mode texte (fichier .csv, *c.f.* TP1, utilisable par exemple dans un tableur) par les commandes `np.savetxt(nom_fichier,M)` et `M=np.genfromtxt(nom_fichier)`
— En mode binaire (beaucoup plus rapide et efficace, mais utilisable via Python uniquement), par les commandes
`np.save(nom_fichier,M)` et `M=np.load(nom_fichier)`

Pour afficher un `ndarray` `A`, on peut l'imprimer à l'aide de `print(A)` ; Hormis pour de petites matrices ou de petits vecteurs, le résultat est assez illisible. Une option plus graphique consiste à utiliser `plt.imshow(A,interpolation='nearest')` qui affiche la matrice comme une image pixellisée, en *fausses couleurs*.

20. Pourvu que l'on ait chargé la bibliothèque `scipy.linalg` comme indiqué en première page