informatique

DUT INFO / 2ème année / 2014 - 2015

Mis à jour le 17 mars 2015 à 10:14

Les nouvelles aventures mathématiques du programmeur

Guillaume CONNAN

UNIVERSITÉ DE NANTES

IUT NANTES Informatique

TABLE DES MATIÈRES

THÈME

1

Types, polymorphisme, monades et algèbre

1 Déterminant d'une matrice carrée

1 1 Déterminant d'une matrice de taille 2

Godfried W. Leibniz
(1646-1716)

En fait, vous savez depuis la Seconde que deux vecteurs du plan de coordonnées (a,b) et (a',b') sont colinéaires si, et seulement si, leurs coordonnées sont proportionnelles. Il existe donc un réel λ tel que $a=\lambda a'$ et $b=\lambda b'$ et alors

$$ab'-a'b=\lambda ab-\lambda ab=0$$

On appelle déterminant de la matrice $A=\begin{pmatrix}a & a'\\ b & b'\end{pmatrix}$ le nombre $ab'-a'b$ et on note :

$$\det(A)=\begin{vmatrix}a & a'\\ b & b'\end{vmatrix}=ab'-a'b$$

1 2 Propriétés des déterminants

Dans la suite, $A=(a_{ij})$, $B=(b_{ij})$ sont des matrices carrées de taille n, λ est un scalaire. L_i est une ligne quelconque de A. On note Δ_{ij} le déterminant de la matrice obtenue à partir de A en « barrant » la ligne i et la colonne j.

Propriété 1 - 1

Soient i et j deux entiers compris entre 1 et n.

$$\det A=\boxplus_{k=1}^{n} a_{1k}\boxdot(-1)^{1+k}\boxdot\Delta_{1k}=\boxplus_{k=1}^{n} a_{ik}\boxdot(-1)^{i+k}\boxdot\Delta_{ik}=\boxplus_{k=1}^{n} a_{kj}\boxdot(-1)^{k+j}\boxdot\Delta_{kj}$$

On admettra cette propriété. Voyons ce que cela donne pour une matrice de taille 3 :

$$(-1)^{(1+1)}\begin{vmatrix}\boxed{a_{11}} & a_{12} & a_{13}\\ a_{21} & \boxed{a_{22}} & \boxed{a_{23}}\\ a_{31} & \boxed{a_{32}} & \boxed{a_{33}}\end{vmatrix}+(-1)^{(2+1)}\begin{vmatrix}a_{11} & \boxed{a_{12}} & \boxed{a_{13}}\\ \boxed{a_{21}} & a_{22} & a_{23}\\ a_{31} & \boxed{a_{32}} & \boxed{a_{33}}\end{vmatrix}+(-1)^{(3+1)}\begin{vmatrix}a_{11} & \boxed{a_{12}} & \boxed{a_{13}}\\ a_{21} & \boxed{a_{22}} & \boxed{a_{23}}\\ \boxed{a_{31}} & a_{32} & a_{33}\end{vmatrix}$$

C'est-à-dire :

$$\det(A)=a_{11}\begin{vmatrix}a_{22} & a_{23}\\ a_{32} & a_{33}\end{vmatrix}-a_{21}\begin{vmatrix}a_{12} & a_{13}\\ a_{32} & a_{33}\end{vmatrix}+a_{31}\begin{vmatrix}a_{12} & a_{13}\\ a_{22} & a_{23}\end{vmatrix}$$

Par exemple, calculons le déterminant de $A=\begin{pmatrix}1 & 2 & 3\\ 2 & 1 & -2\\ -2 & 3 & 1\end{pmatrix}$

$$(-1)^{(1+1)}\begin{vmatrix}\boxed{1} & 2 & 3\\ 2 & \boxed{1} & \boxed{-2}\\ -2 & \boxed{3} & \boxed{1}\end{vmatrix}+(-1)^{(2+1)}\begin{vmatrix}1 & \boxed{2} & \boxed{3}\\ \boxed{2} & 1 & -2\\ -2 & \boxed{3} & \boxed{1}\end{vmatrix}+(-1)^{(3+1)}\begin{vmatrix}1 & \boxed{2} & \boxed{3}\\ 2 & \boxed{1} & \boxed{-2}\\ \boxed{-2} & 3 & 1\end{vmatrix}$$

C'est-à-dire :

$$\begin{aligned}\det(A)&=1\times\begin{vmatrix}1 & -2\\ 3 & 1\end{vmatrix}-2\times\begin{vmatrix}2 & 3\\ 3 & 1\end{vmatrix}+(-2)\times\begin{vmatrix}2 & 3\\ 1 & -2\end{vmatrix}\\ &=(1+6)-2(2-9)-2(-4-3)\\ &=7+14+14\\ &=35\end{aligned}$$

```
# det entier [[1;2;3];[2;1;-2];[-2;3;1]];;
- : int = 35
```

Recherche

Appliquez la même méthode pour calculer de tête le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 1 & 17 & 32 & 49 \\ 0 & 5 & 0 & 3 \\ 0 & 39 & 2 & -49 \\ 0 & 7 & 0 & 1 \end{pmatrix}$$

Propriété 1 - 2

Le déterminant d'une matrice triangulaire est égal au produit des éléments de sa diagonale principale.

Sans utiliser cette propriété, calculez de tête le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 2 & 32 & 45 & 87 & 987 \\ 0 & -1 & 568 & -542 & 712 \\ 0 & 0 & 5 & 741 & -12 \\ 0 & 0 & 0 & 1 & -789 \\ 0 & 0 & 0 & 0 & -10 \end{pmatrix}$$

Prouvez alors cette propriété dans le cas général.

Propriété 1 - 3

Si la matrice B résulte de l'échange de deux lignes ou de deux colonnes d'une matrice A alors $\det B = -\det A$

On admettra cette propriété.
Calculez de tête le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 45 & 32 & 2 & 987 & 87 \\ 568 & -1 & 0 & 712 & -542 \\ 5 & 0 & 0 & -12 & 741 \\ 0 & 0 & 0 & -789 & 1 \\ 0 & 0 & 0 & -10 & 0 \end{pmatrix}$$

Propriété 1 - 4

$$\det\left({}^{t}A\right) = \det A$$

Démontrez cette propriété.

Propriété 1 - 5

Le déterminant d'une matrice comportant une ligne (ou une colonne) de 0 est nul.

Démontrez cette propriété.

Propriété 1 - 6

Soit B la matrice obtenue à partir de A en effectuant $L_i \leftarrow \lambda L_i$, alors $\det B = \lambda \det A$.

Démontrez cette propriété.

Propriété 1 - 7

$$\det(\lambda A) = \lambda^n \det A$$

Démontrez cette propriété.
Si A est une matrice carrée d'ordre 5 dont le déterminant vaut 4, que vaut $\det(-2A)$?

Démontrez que le déterminant d'une matrice antisymétrique (i.e. ${}^tA = -A$) d'ordre 37 est nul.

Propriété 1 - 8 Le déterminant d'une matrice comportant deux lignes (ou colonnes) identiques est nul.

Démontrez cette propriété en utilisant la propriété 1 - 3 page précédente.

Propriété 1 - 9 Le déterminant d'une matrice qui comporte deux lignes (ou colonnes) dont l'une est multiple de l'autre est nul.

Démontrez cette propriété.

Propriété 1 - 10 Soit B la matrice obtenue à partir de A en effectuant $L_i \leftarrow L_i + \lambda L_j$, alors $\det B = \det A$.

Démontrez cette propriété. Utilisez-la pour calculer le déterminant de la matrice suivante :

$$M = \begin{pmatrix} 1 & 3 & 2 & -1 \\ -1 & -2 & 2 & 6 \\ 2 & 4 & -3 & 2 \\ -2 & -7 & 4 & 1 \end{pmatrix}$$

Estimez le nombre d'opérations nécessaires pour effectuer le calcul du déterminant d'une matrice de taille n en utilisant uniquement la définition.

Faites de même lorsqu'on utilise l'algorithme Fang Tcheng

En juin 2012, un ordinateur américain de la firme IBM a établi un nouveau record de vitesse de calcul de 16,32 péta-flops, soit $16,32 \times 10^{16}$ opérations mathématiques à virgule flottante par seconde. En comparaison, un ordinateur individuel standard a une puissance d'environ 100 giga-flops. L'ordinateur utilisé compte 1 572 864 cœurs. Combien de temps lui faudra-t-il environ pour effectuer le calcul d'un déterminant de taille 30 à l'aide de la définition ? Avec l'algorithme Fang Tcheng ?

Il est à noter que le test de performance servant à classer les superordinateurs est le LINPACK : il mesure le temps mis pour résoudre un système dense (sans zéros) de n équations à n inconnues en utilisant l'algorithme Fang Tcheng

Propriété 1 - 11 Soit A et B deux matrices carrées de même taille alors

$$\det(A \times B) = \det A \cdot \det B$$

2 Rotations

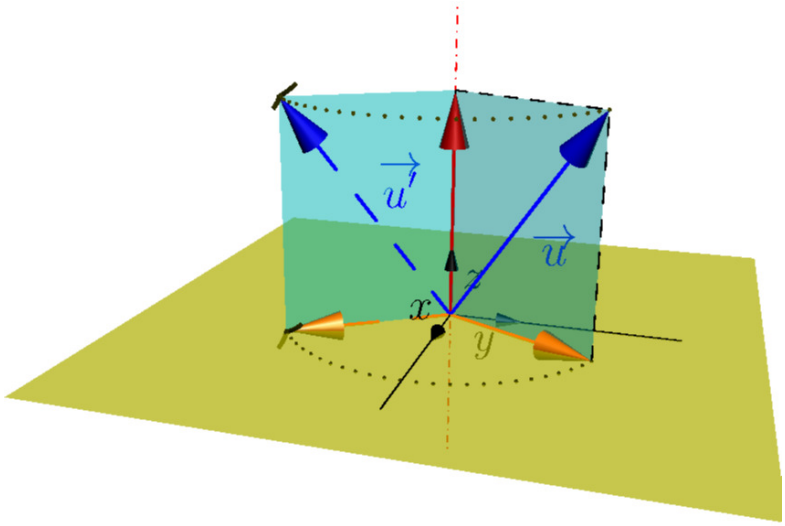
2 1 Approche intuitive

Une rotation vectorielle est une application linéaire particulière. Elle sert à décrire le mouvement qui correspond à ce que vous avez étudié dans le secondaire dans le plan affine mais le généralise en dimension quelconque et surtout, les objets « déplacés » sont des vecteurs, ce qui est un peu plus abstrait que les points ou les solides...

Plus étrange encore, nous ne définirons les rotations vectorielles qu'à la fin de ce chapitre. Nous partirons d'abord d'une approche intuitive.

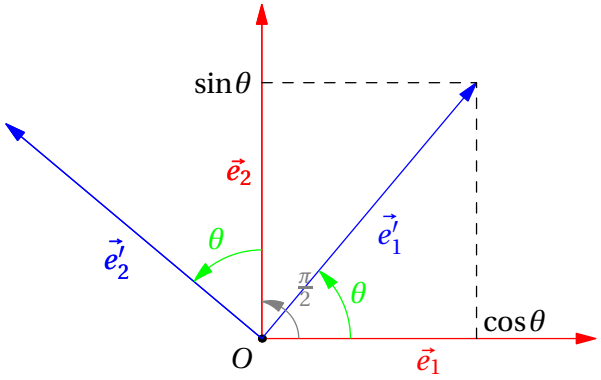
Nous travaillerons la plupart du temps dans $\mathbb{R}^3$, l'espace usuel de dimension 3. Une rotation sera alors caractérisée par un axe et un angle « autour » duquel les vecteurs « tournent »...

Dans un premier temps, nous étudierons une rotation d'axe $\overrightarrow{e_3}$.



Rotation d'un vecteur

Nous avons vu au module précédent que pour caractériser une application linéaire (ici une rotation), il suffit de déterminer les images de chaque vecteur de base.
On observe que e_3 est invariant car il a la même direction que l'axe de rotation.
Il suffit donc de se placer dans le plan (e_1, e_2) et de déterminer les images des deux vecteurs de base :



Projection orthogonale de la rotation d'un vecteur

Si vous avez quelques souvenirs trigonométriques de collège ou de lycée, vous ne serez pas surpris par les résultats suivants :

$$\overrightarrow{e'_1} = \cos\theta\overrightarrow{e_1} + \sin\theta\overrightarrow{e_2}$$
$$\overrightarrow{e'_2} = \cos\left(\theta + \tfrac{\pi}{2}\right)\overrightarrow{e_1} + \sin\left(\theta + \tfrac{\pi}{2}\right)\overrightarrow{e_2} = -\sin\theta\overrightarrow{e_1} + \cos\theta\overrightarrow{e_2}$$

Ce qui donne pour notre matrice, en se souvenant que $\overrightarrow{e'_3} = e_3$:

$$\begin{matrix} r(\overrightarrow{e_1}) & r(\overrightarrow{e_2}) & r(\overrightarrow{e_3}) & \\ \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} & & & \begin{matrix} \overrightarrow{e_1} \\ \overrightarrow{e_2} \\ \overrightarrow{e_3} \end{matrix} \end{matrix}$$

2 2 Détermination de l'image d'un vecteur

Soit $\vec{u}$ un vecteur de coordonnées (x, y, z). Cela signifie que $\vec{u} = x\overrightarrow{e_1} + y\overrightarrow{e_2} + z\overrightarrow{e_3}$.

Soit r la rotation d'axe $\overrightarrow{e_3}$ et d'angle θ.
On obtient alors que :

$$\begin{aligned} r(\overrightarrow{u}) &= xr(\overrightarrow{e_1}) + yr(\overrightarrow{e_2}) + zr(\overrightarrow{e_3}) \\ &= x\big(\cos(\theta)\overrightarrow{e_1} + \sin(\theta)\overrightarrow{e_2}\big) + y\big(-\sin(\theta)\overrightarrow{e_1} + \cos(\theta)\overrightarrow{e_2}\big) + z\overrightarrow{e_3} \\ &= (x\cos(\theta) - y\sin(\theta))\overrightarrow{e_1} + (x\sin(\theta) + y\cos(\theta))\overrightarrow{e_2} + z\overrightarrow{e_3} \end{aligned}$$

Notons R la matrice de r dans la base canonique et U la matrice colonne des coordonnées de $\overrightarrow{u}$ dans la base canonique. Calculons $R \times U$:

$$\underbrace{\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{R} \quad \underbrace{\begin{pmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \\ z \end{pmatrix}}_{R \times U} \qquad \begin{matrix} \\ \\ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \\ \\ \boxed{U} \end{matrix}$$

Miracle ! $R \times U$ est en fait la matrice colonne des coordonnées de $r(\overrightarrow{u})$. Un problème de transformation géométrique se règle par de simples multiplications : la puissance des mathématiques en marche grâce à une notion fondamentale, l'*isomorphie*. On trouve un « corps mathématiques » tout à fait similaire à celui étudié mais dans lequel il est plus aisé de travailler...

Remarque

Espaces isomorphes
Ainsi, il existe de telles *isomorphismes* entre l'Espace géométrique de dimension 3, l'ensemble $\mathbb{R}^3$ de leurs coordonnées et $\mathcal{M}_{3,1}(\mathbb{R})$, l'ensemble des matrices à coefficients réels ayant trois lignes et une colonne. On parlera souvent de l'un pour l'autre.

2 3 Généralisation

On peut alors admettre la généralisation suivante :

Propriété 1 - 12

Expression matricielle de l'image d'un vecteur
Soit M la matrice d'un déplacement dans une certaine base $\mathscr{B}$ de $\mathbb{R}^3$.
Soit U la matrice colonne des coordonnées d'un certain vecteur $\overrightarrow{u}$ dans cette base.
Alors la matrice colonne des coordonnées de $f(\overrightarrow{u})$ dans cette base vaut :

$$\text{mat}_{\mathscr{B}} f(\overrightarrow{u}) = M \times U$$

2 4 Matrice d'une composée de déplacements

On est souvent amené à décomposer le mouvement d'un système en mouvements simples. Pour obtenir les coordonnées finales à partir des initiales, il faut ensuite recomposer les mouvements. Étudions le cas de la composée de deux déplacements.

Remarque

composée d'applications
On note $\varphi \circ g$ l'application composée de g suivie de f (attention à l'ordre !).
Ainsi, $\varphi \circ g(x) = f\big(g(x)\big)$: on calcule $g(x)$ puis on calcule l'image par f du vecteur obtenu.

Considérons deux déplacements f et g. Notons $M = \text{mat}_{\mathscr{B}} f$, $N = \text{mat}_{\mathscr{B}} g$ et $U = \text{mat}_{\mathscr{B}} \overrightarrow{u}$. Or

$$f \circ g(\overrightarrow{u}) = f\Big(g(\overrightarrow{u})\Big)$$

donc, point de vue matrice :

$$\text{mat}_{\mathscr{B}} f \circ g(\vec{u}) = M \times (N \times U) = (M \times N) \times U$$

On en déduit la propriété suivante :

Propriété 1 - 13

Expression matricielle de la composée de deux applications linéaires

Soit M la matrice d'un déplacement dans une certaine base $\mathscr{B}$ de $\mathbb{R}^3$.
Soit N la matrice d'un déplacement dans la même base.
Alors :

$$\text{mat}_{\mathscr{B}} f \circ g = M \times N$$

En particulier, avec les notations habituelles,

$$\text{mat}_{\mathscr{B}} f \circ g(\vec{u}) = (M \times N) \times U$$

3 Produit scalaire - Matrices orthogonales

3 1 Produit scalaire de deux vecteurs

Nous rappellerons juste la définition vue en Terminale :

Définition 1 - 1

produit scalaire

Dans un repère orthonormé de $\mathbb{R}^3$, deux vecteurs $\vec{v}$ et $\vec{v'}$ ont pour coordonnées respectives (x, y, z) et (x', y', z').
On appelle *produit scalaire* de $\vec{v}$ et $\vec{v'}$ et on note $\left\langle \vec{v}, \vec{v'} \right\rangle$ le nombre réel :

$$\left\langle \vec{v}, \vec{v'} \right\rangle = xx' + yy' + zz'$$

3 2 Représentation matricielle

Nous aurons besoin pour le reste de notre étude la *transposée* d'une matrice.

Définition 1 - 2

matrice transposée

On appelle transposée de la matrice A et on note ${}^{t}A$ la matrice obtenue en échangeant lignes et colonnes.

Par exemple, si $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$, alors ${}^{t}A = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$
Soit $\vec{v}$ et $\vec{v'}$ deux vecteurs de représentations matricielles respectives V et V' dans une certaine base orthonormée.
Pour obtenir matriciellement le produit scalaire des deux vecteurs, on effectue le produit $V \times {}^{t}V'$.
Nous aurons besoin enfin de cette importante définition :

Définition 1 - 3

vecteurs orthogonaux

Deux vecteurs sont orthogonaux si, et seulement si, leur produit scalaire est nul.

3 3 Norme d'un vecteur

Il s'agit tout simplement de la racine carrée du produit scalaire d'un vecteur par lui-même.

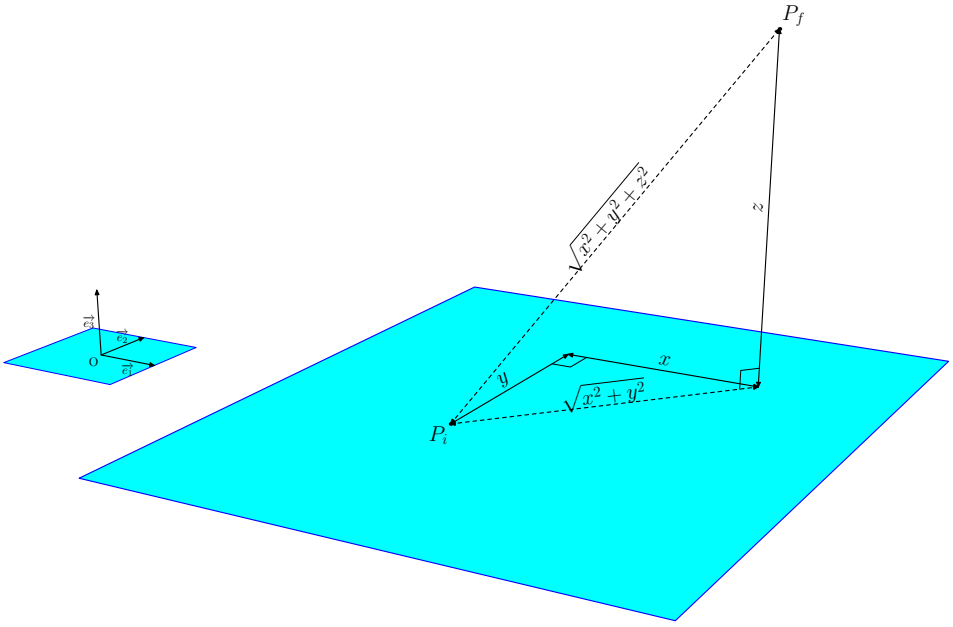
Définition 1 - 4

norme d'un vecteur

Soit $\vec{v}$ un vecteur. On appelle norme de $\vec{v}$ le nombre noté $n^\circ \vec{v}$ et défini par :

$$n^\circ \vec{v} = \sqrt{\left\langle \vec{v}, \vec{v} \right\rangle}$$

Ainsi, si $\vec{v}$ a pour coordonnées (x, y, z) dans un repère orthonormé, alors $n^\circ \vec{v} = \sqrt{x^2 + y^2 + z^2}$.
Cela vous rappelle sûrement un vieux théorème de collège...



norme d'un vecteur

3 4 Matrices orthogonales

Nous allons enfin pouvoir revenir à nos transformations de l'espace. Il nous faut d'abord donner une nouvelle définition...

Définition 1 - 5

matrice orthogonale

Une matrice de $\mathcal{M}_3(\mathbb{R})$ est orthogonale si, et seulement si, elle est inversible et $A^{-1} = {}^tA$

Voici qui est bien pratique : pour inverser ce genre de matrices, il suffit de les transposer. Il nous reste à découvrir qui sont vraiment ces matrices...

Théorème 1 - 1

caractérisation d'une matrice orthogonale

Une matrice de $\mathcal{M}_3(\mathbb{R})$ est orthogonale si, et seulement si, ses colonnes forment une base orthonormée de $\mathbb{R}^3$.

Une nouvelle fois, nous ne démontrerons pas ce théorème (car nous ne sommes pas vraiment mathématiciens...).

Utilisons-le cependant dès maintenant. Reprenons la matrice d'une rotation d'angle θ et d'axe (Oz) et appelons C_1, C_2 et C_3 ses colonnes :

$$R = \begin{matrix} C_1 & C_2 & C_3 \end{matrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Alors la norme du vecteur associé à C_1 vaut $\sqrt{\cos^2\theta + \sin^2\theta + 0^2} = 1$. Il en va de même pour C_2 et C_3.

Formons à présent les produits scalaires :

$$C_1 \times {}^tC_2 = -\cos\theta\sin\theta + \sin\theta\cos\theta + 0 = 0$$

Il en va de même pour C_1 et C_3 ainsi que pour C_2 et C_3.

La matrice de cette rotation est donc orthogonale.

3 5 Quelques propriétés des matrices orthogonales

Nous allons d'abord donner, une nouvelle fois sans preuve, deux propriétés importantes des déterminants :

Propriété 1 - 14

déterminant d'une transposée et d'un produit

$$\det {}^t A = \det A$$

$$\det A \times B = \det A \cdot \det B$$

cela va nous permettre d'établir le théorème suivant :

Théorème 1 - 2

déterminant d'une matrice orthogonale

Une matrice orthogonale a pour déterminant 1 ou -1

Allez, pour le plaisir, au moins une preuve dans ce cours....

On a d'une part $\det(A \times {}^t A) = \det A \cdot \det {}^t A = (\det A)^2$.

D'autre part $\det(A \times {}^t A) = \det(I_3) = 1$.

Finalement, $(\det A)^2 = 1$, c'est-à-dire $\det A = \pm 1$.

Bon, ce n'était pas méchant...

Théorème 1 - 3

Détermination de l'axe d'une rotation

Si A est une matrice orthogonale de $\mathcal{M}_3(\mathbb{R})$ différente de l'identité et dont le déterminant vaut 1, alors A est la matrice d'une rotation autour d'un axe.
La direction de l'axe est donnée en déterminant un vecteur invariant.

Voilà pour l'axe. En ce qui concerne l'angle, nous aurons besoin...d'une nouvelle définition !

Définition 1 - 6

trace d'une matrice carrée

La trace d'une matrice, notée $\mathrm{tr}(A)$ est la somme des éléments de sa diagonale.

Par exemple, la trace de la matrice $\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$ est $2\cos\theta + 1$.

Or il existe une belle propriété qui dit que la trace ne change pas lorsqu'on effectue des changements de base et donc...

Théorème 1 - 4

détermination du cosinus de l'angle d'une rotation

Soit r une rotation de matrice A dans une certaine base. Alors l'angle θ de la rotation vérifie :

$$\cos\theta = \frac{\mathrm{tr}(A) - 1}{2}$$

ce qui est « magique », c'est qu'on va ainsi pouvoir déterminer la rotation quelque soit la base.

3 6 Produit vectoriel

Nous avons parlé du produit scalaire qui à deux vecteurs associe un nombre réel (un scalaire). Nous allons maintenant introduire un nouveau produit qui à deux vecteurs associe un troisième

vecteur.

Définition 1 - 7

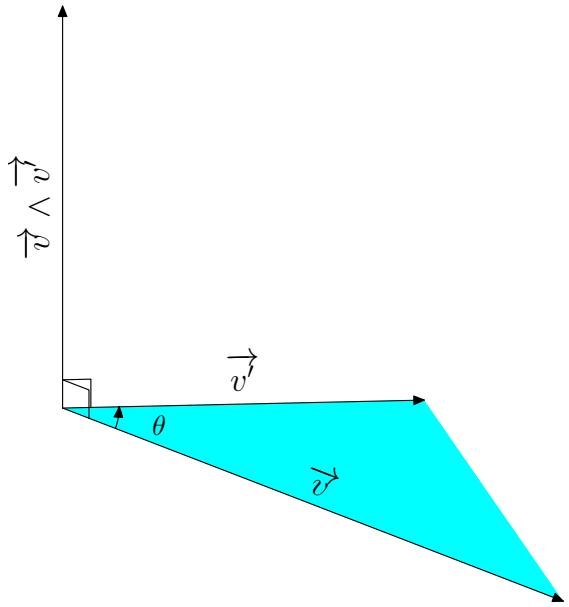
produit vectoriel

Soit $\overrightarrow{v}$ et $\overrightarrow{v'}$ deux vecteurs de coordonnées (x, y, z) et (x', y', z') dans une base orthonormée directe.

On appelle produit vectoriel de $\overrightarrow{v}$ et $\overrightarrow{v'}$ et on note $\overrightarrow{v} \wedge \overrightarrow{v'}$ le vecteur dont les coordonnées sont :

$$\left(\begin{vmatrix} y & z \\ y' & z' \end{vmatrix}, -\begin{vmatrix} x & z \\ x' & z' \end{vmatrix}, \begin{vmatrix} x & y \\ x' & y' \end{vmatrix} \right)$$

On peut vérifier assez facilement que le *vecteur* $v \wedge v'$ est orthogonal à la fois à $\overrightarrow{v}$ et à $\overrightarrow{v'}$.



produit vectoriel

On montre aussi le résultat important suivant :

$$\left\| \overrightarrow{v} \wedge \overrightarrow{v'} \right\| = n^{\circ}\overrightarrow{v} \times n^{\circ}\overrightarrow{v'} \times \left| \sin\left(\overrightarrow{v}, \overrightarrow{v'}\right) \right|$$

Remarque

disposition pratique

Une méthode peu orthodoxe mais pratique de retenir la formule de calcul du produit scalaire est de le disposer comme un déterminant d'une matrice de taille 3 :

$$\overrightarrow{v} \wedge \overrightarrow{v'} = \begin{vmatrix} \overrightarrow{e_1} & \overrightarrow{e_2} & \overrightarrow{e_3} \\ x & y & z \\ x' & y' & z' \end{vmatrix}$$

puis de développer par rapport à la première ligne :

$$\overrightarrow{v} \wedge \overrightarrow{v'} = \begin{vmatrix} y & z \\ y' & z' \end{vmatrix} \overrightarrow{e_1} - \begin{vmatrix} x & z \\ x' & z' \end{vmatrix} \overrightarrow{e_2} + \begin{vmatrix} x & y \\ x' & y' \end{vmatrix} \overrightarrow{e_3}$$

On peut donc montrer que $\overrightarrow{v} \wedge \overrightarrow{v'} = -\overrightarrow{v'} \wedge \overrightarrow{v}$

Remarque

règle du tire-bouchon

Très grossièrement, quand on « tourne » de $\overrightarrow{v}$ vers $\overrightarrow{v'}$, on « monte le long » de $\overrightarrow{v} \wedge \overrightarrow{v'}$ et ~~vis~~ vice-versa

3 7 Exemple de détermination d'une rotation

On connaît la matrice d'une certaine application dans une certaine base :

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

1. Vérifiez que la matrice est orthogonale.

2. Vérifiez que $\det(A) = 1$: il s'agit donc d'une rotation.

3. Pour déterminer l'axe, on va chercher un vecteur de représentation matricielle $V = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ invariant par la rotation donc tel que $A \times V - V$ soit égal à la matrice colonne nulle. On obtient après simplification (faites le calcul...) :

$$\begin{pmatrix} \frac{-x-y-2z}{3} \\ \frac{2x-y+z}{3} \\ \frac{x-2y-z}{3} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

cela revient à résoudre le système :

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

Ce système admet une infinité de solutions car en additionnant les deux premières lignes on obtient la troisième. On choisit une des inconnues comme paramètre (par exemple z) et on résout le système résultant ayant maintenant deux équations et deux inconnues.

On obtient $x = -z$ et $y = -z$.

L'axe de la rotation est donc la droite vectorielle dirigée par le vecteur $\vec{v}$ de coordonnées $(1, 1, -1)$.

4. Pour l'angle, on sait que $\cos\theta = \frac{\mathrm{tr}(A)-1}{2} = \frac{2-1}{2} = \frac{1}{2}$.

On obtient donc que $\theta = \pm\frac{\pi}{3}$ à un multiple de 2π près.

Pour déterminer le signe, il nous faut un autre renseignement : nous allons utiliser le produit vectoriel. Nous allons déterminer un vecteur $\vec{n}$ orthogonal à $\vec{v}$. Il faut donc s'arranger pour que le produit scalaire de $\vec{n}$ et $\vec{v}$ soit nul.

Si nous posons (x, y, z) les coordonnées de $\vec{n}$ alors $\langle \vec{v}, \vec{n} \rangle = x + y - z$.

Nous pouvons donc choisir $(1, 0, 1)$ pour les coordonnées de $\vec{n}$.

Déterminons à présent l'image de $\vec{n}$ par la rotation :

$$\underbrace{\begin{pmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-2}{3} \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{-2}{3} & \frac{2}{3} \end{pmatrix}}_{A} \quad \underbrace{\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}}_{N} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Donc $r(\vec{n})$ a pour coordonnées $(0, 1, 1)$.

Calculons à présent le produit vectoriel de $\vec{n}$ par $r(\vec{n})$:

$$\vec{n} \wedge r(\vec{n}) = \begin{vmatrix} \vec{e_1} & \vec{e_2} & \vec{e_3} \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix} = -\vec{e_1} - \vec{e_2} + \vec{e_3} = -\vec{v}$$

Si nous en avions le loisir, nous pourrions montrer le résultat suivant :

Propriété 1 - 15

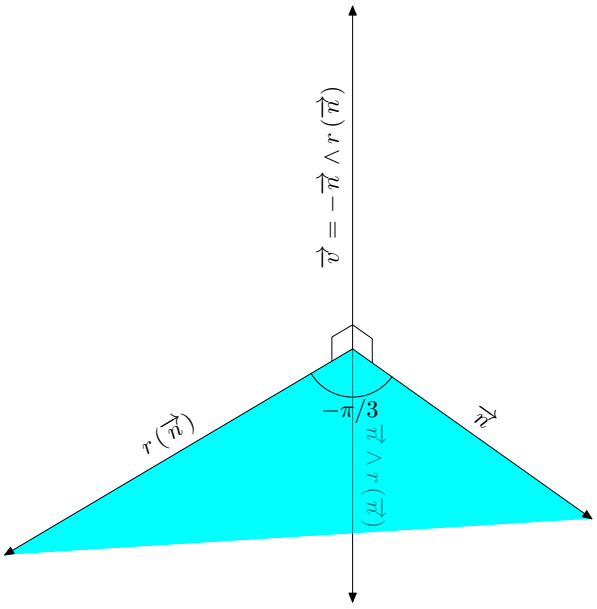
rotation et produit vectoriel

Si l'axe d'une rotation r est dirigé par un vecteur $\vec{v}$ et si $\vec{n}$ est un vecteur du plan orthogonal à $\vec{v}$ alors

$$\vec{n} \wedge r(\vec{n}) = \frac{n^\circ \vec{n} \times \|r(\vec{n})\|}{n^\circ \vec{v}} \times \sin(\theta) \cdot \vec{v}$$

Ainsi, $-\vec{v} = \frac{\sqrt{2}\times\sqrt{2}}{\sqrt{3}} \times \sin(\theta) \cdot \vec{v}$. Nous en déduisons que $\sin(\theta) = -\frac{\sqrt{3}}{2}$ or $\cos(\theta) = \frac{1}{2}$.

Finalement, A est la matrice dans la base canonique de la rotation d'axe dirigé par $\vec{v}$ de coordonnées $(1, 1, -1)$ et d'angle $-\frac{\pi}{3}$.



rotation de matrice A

4 Changement de base

4 1 Le problème

On connaît la matrice A d'une rotation relativement à une certaine base, généralement une base orthonormée dont un des vecteurs dirige l'axe de la rotation.

On voudrait alors connaître la matrice de cette rotation mais relativement à une autre base...

4 2 Matrice de passage

Définition 1 - 8

matrice de passage

Soit $\mathscr{B}$ et $\mathscr{B}'$ deux bases de $\mathbb{R}^3$. On appelle matrice de passage de $\mathscr{B}$ à $\mathscr{B}'$ et on note $\mathscr{P}_{\mathscr{B},\mathscr{B}'}$ la matrice de $M_3(\mathbb{R})$ dont les colonnes sont les vecteurs de $\mathscr{B}'$ exprimés dans $\mathscr{B}$:

$$\begin{array}{c} \begin{array}{ccc} \overrightarrow{e'_1} & \overrightarrow{e'_2} & \overrightarrow{e'_3} \end{array} \\ \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix} \begin{array}{c} \overrightarrow{e_1} \\ \overrightarrow{e_2} \\ \overrightarrow{e_3} \end{array} \end{array}$$

À noter : si les bases sont *orthonormées* alors les matrices de passages seront *orthogonales*.

Nous retiendrons alors la propriété importante suivante :

Propriété 1 - 16

matrice de passage inverse

$$\mathscr{P}_{\mathscr{B}',\mathscr{B}} = \mathscr{P}^{-1}_{\mathscr{B},\mathscr{B}'}$$

En particulier, si les bases sont *orthonormées* :

$$\mathscr{P}_{\mathscr{B}',\mathscr{B}} = {}^{\mathrm{t}}\mathscr{P}_{\mathscr{B},\mathscr{B}'}$$

4 3 Changement de coordonnées d'un vecteur

Soit $\overrightarrow{v}$ un vecteur dont les coordonnées dans une certaine base $\mathscr{B}$ sont (x, y, z).

On peut donc associer une matrice colonne V à ce vecteur : $V = \mathrm{mat}_{\mathscr{B}}\overrightarrow{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

ATTENTION ! À un même vecteur correspond une infinité de vecteurs colonnes différents selon la base choisie !

Notons maintenant $V' = \mathrm{mat}_{\mathscr{B}'}\overrightarrow{v}$ alors :

Propriété 1 - 17

changement de coordonnées d'un vecteur

$$V = P_{\mathscr{B},\mathscr{B}'} \times V'$$

Notez bien que cette formule donne les anciennes coordonnées en fonction des nouvelles !

Remarque

On peut en fait retenir que les « primes » sont du côté des « primes ».

Cette formule n'est donc pas pratique sous cette forme mais sachant que $V = \mathscr{P}_{\mathscr{B},\mathscr{B}'}V'$, alors

$$V' = \mathscr{P}^{-1}_{\mathscr{B},\mathscr{B}'} \times V$$

et comme nous travaillerons dans des bases orthonormées, nous utiliserons le plus souvent :

$$V' = {}^{\mathrm{t}}\mathscr{P}_{\mathscr{B},\mathscr{B}'} \times V$$

4 4 Relation de Chasles

Considérons 3 bases. Avec des notations standard on a

$$V'' = \mathscr{P}_{\mathscr{B}'',\mathscr{B}} \times V$$

Mais on a aussi $V' = \mathscr{P}_{\mathscr{B}',\mathscr{B}} \times V$ et $V'' = \mathscr{P}_{\mathscr{B}'',\mathscr{B}'} \times V'$ donc

$$V'' = \mathscr{P}_{\mathscr{B}'',\mathscr{B}'} \times \mathscr{P}_{\mathscr{B}',\mathscr{B}} \times V$$

Finalement :

Propriété 1 - 18

relation de Chasles

$$\mathscr{P}_{\mathscr{B}'',\mathscr{B}} = \mathscr{P}_{\mathscr{B}'',\mathscr{B}'} \times \mathscr{P}_{\mathscr{B}',\mathscr{B}}$$

4 5 Rotation et changement de base

Soit A la matrice d'une rotation r dans une certaine base. Notons toujours $V = \mathrm{mat}_{\mathscr{B}}\overrightarrow{v}$ et $W = \mathrm{mat}_{\mathscr{B}} r(\overrightarrow{v})$ et leurs pendants « primés ».
Nous avons vu que $W = A \times V$. D'après ce que nous avons vu au paragraphe précédent, cela donne :

$$\mathscr{P}_{\mathscr{B},\mathscr{B}'} \times W' = A \times \mathscr{P}_{\mathscr{B},\mathscr{B}'} \times V'$$

et donc

$$W' = \left({}^{\mathrm{t}}\mathscr{P}_{\mathscr{B},\mathscr{B}'} \times A \times \mathscr{P}_{\mathscr{B},\mathscr{B}'}\right) \times V'$$

On en déduit que $A' = {}^{\mathrm{t}}\mathscr{P}_{\mathscr{B},\mathscr{B}'} \times A \times \mathscr{P}_{\mathscr{B},\mathscr{B}'}$

Théorème 1 - 5

transformation orthogonale et changement de base

$$\mathrm{mat}_{\mathscr{B}'} f = {}^{\mathrm{t}}\mathscr{P}_{\mathscr{B},\mathscr{B}'} \times \mathrm{mat}_{\mathscr{B}} f \times \mathscr{P}_{\mathscr{B},\mathscr{B}'}$$

Il sera pratique de retenir le schéma suivant (attention au sens des flèches) :

Remarque

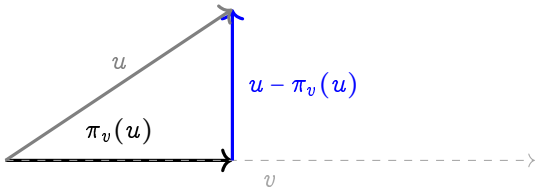
schéma du changement de base

$$\begin{array}{ccc} \mathscr{B}\ \mathbb{R}^3 & \xrightarrow[A]{f} & \mathbb{R}^3\ \mathscr{B} \\ \downarrow \mathscr{P}_{\mathscr{B},\mathscr{B}'} & & \downarrow \mathscr{P}_{\mathscr{B},\mathscr{B}'} \\ \mathscr{B}'\ \mathbb{R}^3 & \xrightarrow[A']{f} & \mathbb{R}^3\ \mathscr{B}' \end{array}$$

5 Projection orthogonale

5 1 Projection orthogonale sur une droite orthogonale

La figure clé qui guidera notre intuition est celle-ci :



Elle correspond à la définition suivante :

Définition 1 - 9

projection orthogonale sur une droite vectorielle

La projection orthogonale de u sur v est le vecteur noté $\pi_v(u)$ défini par :

$$\pi_v(u) = \frac{\langle u, v\rangle}{\|v\|^2} v$$

Nous démontrerons en exercice que :

Propriété 1 - 19

Le vecteur $u - \pi_v(u)$ est orthogonal à v.

et également :

Propriété 1 - 20

Inégalité de Cauchy-Schwarz

$$|\langle u, v\rangle| \leqslant \|u\| \cdot \|v\|$$

ce qui permet d'obtenir la confirmation de ce qui se voit sur le dessin :

Propriété 1 - 21

$$\|\pi_v(u) \leqslant \|u\|$$

5.2 Projection orthogonales sur un sev

On peut montrer que les composantes d'un vecteur dans une base orthogonale $(e_1, e_2, \ldots, e_n)$ est :

$$\begin{aligned} u &= \pi_{e_1} e_1 + \pi_{e_2} e_2 + \cdots + \pi_{e_n} e_n \\ &= \frac{\langle u, e_1\rangle}{\|e_1\|^2} e_1 + \frac{\langle u, e_2\rangle}{\|e_2\|^2} e_2 + \cdots \frac{\langle u, e_n\rangle}{\|e_n\|^2} e_n \end{aligned}$$

et si la base est orthonormale :

$$u = \rangle u, e_1 \langle e_1 + \rangle u, e_2 \langle e_2 + \cdots \rangle u, e_n \langle e_n$$

On admettra alors le théorème suivant :

Théorème 1 - 6

Soit W un sev de V et $(e_1, e_2, \cdots, e_n)$ une base orthonormale de W. Alors tout vecteur u de V peut s'écrire sous la forme $u = w_1 + w_2$ avec w_1 un vecteur de W et w_2 un vecteur orthogonal à W (i.e. orthogonal à tous les vecteurs de base de W). Alors :

$$w_1 = \rangle u, e_1 \langle e_1 + \rangle u, e_2 \langle e_2 + \cdots \rangle u, e_n \langle e_n$$

On appelle w_1 la projection orthogonale de u sur W.

5.3 Procédé d'orthonormalisation de Gram-Schmidt

Nous aurons besoin pour compresser nos images en niveaux de gris d'orthonormaliser une base donnée. Nous admettrons que le procédé suivant répond à notre problème.
On part d'une base $(e_1, \ldots, e_n)$ quelconque et on veut en déduire une base $(u_1, \ldots, u_n)$ orthonormale. Il suffit de prendre :

$$\begin{aligned} u_1 &= \frac{w_1}{\|w_1\|}, \text{ où } w_1 = e_1 \\ u_2 &= \frac{w_2}{\|w_2\|}, \text{ où } w_2 = e_2 - \frac{\langle e_2, w_1\rangle}{\|w_1\|^2} w_1 \\ &\vdots \\ u_n &= \frac{w_n}{\|w_n\|}, \text{ où } w_n = e_n - \frac{\langle e_n, w_1\rangle}{\|w_1\|^2} w_1 - \frac{\langle e_n, w_2\rangle}{\|w_1\|^2} w_2 - \cdots - \frac{\langle e_n, w_{n-1}\rangle}{\|w_{n-1}\|^2} w_{n-1} \end{aligned}$$

Les utilisations de ce procédé sont très nombreuses dans des domaines très divers.

6 Décomposition en valeurs singulières (SVD)

Nous allons travailler avec des images qui sont des matrices de niveaux de gris. Notre belle amie Léna sera représentée par une matrice carrée de taille 2^9 ce qui permet de reproduire Léna à l'aide de $2^{18} = 262\,144$ pixels. Léna prend alors beaucoup de place. Nous allons tenter de compresser la pauvre Léna sans pour cela qu'elle ne perde sa qualité graphique. Une des méthodes les plus abordables est d'utiliser la décomposition d'une matrice en valeurs singulières.

Nous passerons outre la plupart des démonstrations mais, en bon(ne) informaticien(ne) nous essaierons de comprendre le mécanisme général pour mieux appréhender son utilisation « dans la vraie vie ». C'est également une nouvelle preuve des liens étroits existants entre informatique et mathématique...
C'est un sujet extrêmement riche qui a de nombreuses applications. L'algorithme que nous utiliserons (mais que nous ne détaillerons pas) a été mis au point par deux très éminents chercheurs en 1965 (Gene GOLUB, états-unien et William KAHAN, canadien, père de la norme IEEE-754).

6 1 Valeurs et vecteurs propres

6 1 1 Généralités

Un *vecteur propre* v (*eigenvector* en anglais de spécialité ;-)) d'un endomorphisme f d'un $\mathbb{K}$-espace vectoriel E est un vecteur non nul satisfaisant l'équation :

$$f(v) = \lambda v$$

avec λ un élément de $\mathbb{K}$ appelé *valeur propre* (*eigenvalue*) associée au vecteur propre v.
Cela se traduit matriciellement dans une certaine base $\mathscr{B}$. Si A est la matrice de f dans $\mathscr{B}$ et V le vecteur colonne des coordonnées de v dans $\mathscr{B}$, on a :

$$A \times V = \lambda \cdot V$$

Comme f est un endomorphisme, A est une matrice carrée.

6 1 2 Un exemple

Considérons par exemple une matrice $A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$.

Posons $V = \begin{pmatrix} x \\ y \end{pmatrix}$.
On recherche donc x et y ainsi qu'un scalaire λ vérifiant :

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

C'est un simple petit système linéaire :

$$(S_\lambda) : \begin{cases} 2x + y = \lambda x \\ x + 2y = \lambda y \end{cases}$$

qui est équivalent au système :

$$(S_\lambda) \Leftrightarrow \begin{cases} (2-\lambda)x + y = 0 \\ x + (2-\lambda)y = 0 \end{cases}$$

Ce système admet au moins une solution (laquelle ?...) mais cela ne nous arrange pas. Pour que v soit un vecteur propre, il faut et il suffit que le déterminant de ce système soit nul (pourquoi ?).
Or :

$$\begin{vmatrix} (2-\lambda) & 1 \\ 1 & (2-\lambda) \end{vmatrix} = 0 \Leftrightarrow \lambda^2 - 4\lambda + 3 = 0$$

Cette équation admet exactement deux solutions, 3 et 1 qui sont les valeurs propres de A.
Il reste à déterminer des vecteurs propres associés (sont-ils uniques ?).
Remplaçons pour cela λ par 3 dans l'équation.
Le système devient :

$$(S_3) \Leftrightarrow \begin{cases} -x + y = 0 \\ x - y = 0 \end{cases} \Leftrightarrow x = y$$

Un vecteur propre de A associé à 3 est donc par exemple $(1,1)$ mais également tous ses multiples.
Le *sous-espace propre* E_3 associé à 3 (c'est-à-dire l'ensemble des vecteurs propres associé à λ) est donc $\text{Vect}\{(1,1)\}$.
On trouve de même que $E_1 = \text{Vect}\{(1,-1)\}$.
Si la somme des dimensions des sous-espaces propres est égale à la dimension de l'espace E, alors on dit que A est DIAGONALISABLE et peut s'écrire sous la forme :

$$A = P \times \Delta \times P^{-1}$$

avec P la matrice de passage de $\mathscr{B}$ vers la base de vecteurs propres et Δ une matrice diagonales dont les éléments diagonaux sont les valeurs propres de A.
Il est à noter que toute matrice n'est pas diagonalisable. C'est un problème important qui rend de nombreux services dans des domaines très variés quand il est résolu.

6 2 Décomposition en valeurs singulières

6 2 1 Valeurs et vecteurs singuliers

Nous venons d'étudier les éléments propres d'une matrice *carrée*. Les éléments singuliers en sont une généralisation aux matrices de taille quelconque.
Soit A la matrice d'une application linéaire d'un espace E de dimension m dans un espace F de dimension n. Alors A est rectangulaire.
Un réel *positif* σ est une valeur singulière associée à A si, et seulement si, il existe un vecteur orthonormé u dans E et un vecteur orthonormé v dans F tels que (avec les notations usuelles) :

$$A \times V = \sigma \cdot U \qquad \text{et} \qquad {}^{t}A \times U = \sigma \cdot V$$

On dit que u (v) est un vecteur singulier à gauche (à droite) pour σ.

6 2 2 La SVD

Un beau théorème affirme alors que toute matrice rectangulaire A se décompose sous la forme :

$$A = U \times S \times {}^{t}V$$

avec U et V des matrices orthogonales et S une matrice nulle partout sauf sur sa diagonale principale qui contient les valeurs singulières de A rangées dans l'ordre décroissant.
Ce qui est remarquable, c'est que n'importe quelle matrice admet une telle décomposition, alors que la décomposition en valeurs propres (la diagonalisation d'une matrice) n'est pas toujours possible.
Cependant, la décomposition en éléments singuliers utilise la décomposition en éléments propres comme nous le verrons en exercice
Notons r le rang de A et U_i et V_i les vecteurs colonnes de U et V. La décomposition s'écrit :

$$A = \begin{pmatrix} U_1 & U_2 & \cdots & U_r & \cdots & U_m \end{pmatrix} \times \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & \ddots & \\ & & & & 0 \end{pmatrix} \times \begin{pmatrix} {}^{t}V_1 \\ \vdots \\ {}^{t}V_r \\ \vdots \\ {}^{t}V_n \end{pmatrix}$$

ou formulé autrement :

$$\begin{aligned} A &= \sigma_1 \cdot U_1 \times {}^{t}V_1 + \sigma_2 \cdot U_2 \times {}^{t}V_2 + \cdots + \sigma_r \cdot U_r \times {}^{t}V_r + 0 \cdot U_{r+1} \times {}^{t}V_{r+1} + \cdots \\ &= \sigma_1 \cdot U_1 \times {}^{t}V_1 + \sigma_2 \cdot U_2 \times {}^{t}V_2 + \cdots + \sigma_r \cdot U_r \times {}^{t}V_r \end{aligned}$$

Il faut ensuite se souvenir que les σ_i sont classés dans l'ordre décroissant ce qui va avoir une application importante en informatique comme nous le verrons en exercice...

RECHERCHES

Types

Recherche 1 - 1

Quelle est la signature et le rôle de cette fonction :

```haskell
mystere f s []     = s
mystere f s (x:xs) = f x (mystere f s xs)
```

Que donne :

```haskell
mystere (:) " Papa" "Maman"
```

Que pensez-vous de :

```haskell
foldl (:) " Papa" "Maman"
```

Comment obtenir `<polyline points="1,2 3,4 5,6"/>` sous forme de chaîne de caractères à partir de `[(1,2), (3,4), (5,6)]` en utilisant `mystere` ? En utilisant `foldl` ?

Recherche 1 - 2

Encore une fonction mystère :

```haskell
mystere' f p [] = []
mystere' f p (x:xs)
  | p x       = f x : mystere' f p xs
  | otherwise = []
```

Signature ? Résultat de :

```haskell
mystere' (2+) (>7) [8,12,7,13,16]
```

Comparez avec :

```haskell
mystere'' f p = foldr (\x acc -> if p x then (f x) : acc else acc) []
```

Recherche 1 - 3

Donnez deux *one liners* qui concatènent deux listes. La première utilisera `foldr` et la seconde `foldl`
Comparez leur complexité. Pouvez-vous améliorer la moins efficace ?

Recherche 1 - 4

1. Créez un *one liner* qui additionne les éléments d'une paire dans une liste :

```haskell
> ajoutePaires [(1,2), (3,4)]
[3,7]
```

2. Définissez un *one liner* `extremList` qui, selon le premier argument, renvoie le maximum ou le minimum d'une liste. Votre définition doit s'écrire avec une commande de 6 caractères. Jeu du pendu :

```haskell
extremList = ......
```

3. Définissez un *one liner* qui additionne les éléments d'une liste de paires par composante :

```haskell
> ajouteParComposante [(1,2), (3,4), (5,6)]
(9,12)
```

4. Un caractère a une représentation unicode 32 bits en Haskell. C'est un peu beaucoup pour Joe le crapaud qui n'a pas un gros cerveau. Il veut communiquer secrètement avec Josette la reinette sans se faire repérer par la NSA. Il augmente donc chaque code des caractères de son message d'une unité en se limitant aux caractères de 0 à 255. Fournissez à Joe une fonction `chiffre` et à Josette une fonction `dechiffre`.
   Vous pourrez utiliser les fonctions `fromEnum` et `toEnum` qui permettent de convertir un entier en caractère et vice-versa.
5. Vous voulez créer une fonction qui prend une somme d'argent et une liste de montants de pièces et renvoie la liste des nombres de pièces nécessaires pour rendre la monnaie. Vous utiliserez un algorithme glouton.

Recherche 1 - 5

On définit un type `Naturel` pour représenter les nombres naturels de la sorte :

```haskell
type Naturel a = (a -> a) -> (a -> a)
```

Le naturel n est donc représenté par « applique l'argument n fois ». On définit par exemple les fonctions suivantes :

```haskell
zero f  = id
un f    = f
deux f  = f.f
trois f = f.f.f
```

1. Quel est le type de `zero, un, deux, trois` ?
2. Quel est le rôle de `f` dans les définitions précédentes ?
   On peut visualiser un entier avec la fonction :

```haskell
ent n = n (+1) 0
```

3. Quelle est la signature de `ent` ?
4. Donnez la trace de `ent trois`.
5. Définissez les fonctions suivantes :

```haskell
suc :: Naturel a -> Naturel a
-- renvoie la representation du successeur de n

plus :: Naturel a -> Naturel a -> Naturel a
-- renvoie la representation de la somme des deux premiers arguments

fois :: Naturel a -> Naturel a -> Naturel a
-- renvoie la representation du produit des deux premiers arguments
```

Recherche 1 - 6

On considère les types et classes suivants :

```haskell
data Vecteur = Vec Float Float

data Point = Point Float Float

data Figure = Ligne Point Point | Cercle Point Float

class Bougeable a where
  bouge  :: Vecteur -> a -> a -- translation
  symX   :: a -> a -- symetrie par rapport a axe abscisses
  symY   :: a -> a -- symetrie par rapport a axe ordonnees
  rot180 :: a -> a -- rotation de 180 deg par rapport a origine
```

Faites de Point et Figure une instance de Bougeable.
Si a est Bougeable, faites de [a] une instance de Bougeable.

Recherche 1 - 7 bind

Rappel :

```
m >>= f =
   do
     res <- m
     f res
```

et

```
action1 >> action2 =
   do
     action1
     action2
```

Quelle est le type de `<-`
Le type de `>>=` est :

```
λ> :t (>>=)
(>>=) :: Monad m => m a -> (a -> m b) -> m b
```

Considérons la monade IO et les *actions* :

```
return    :: a -> IO a
putStr    :: String -> IO () -- renvoie une chaine dans la sortie standard
getLine   :: IO String  -- lit une chaine depuis l'entree standard
getChar   :: IO Char
writeFile :: FilePath -> String -> IO ()
```

sachant que `FilePath` est un synonyme de `String`.
Par exemple :

```
echo :: IO ()
echo =
  do
   ligne <- getLine
   writeFile "monFichier.txt" ligne
   putStrLn ("Ligne lue : " ++ ligne)
```

1. Écrivez echo avec `>>=` donc sans `do`.
2. Quelle est la signature de `putStrLn . show` ? Son rôle ?
3. Rappel sur la classe `Read` :

```
read :: read a => String -> a
```

Par exemple :

```
λ> read "4" :: Int
4
λ> read "4" :: Float
4.0
```

Créez alors une fonction `getInt :: IO Integer`

4. Créez une fonction qui lit des entiers jusqu'à lire un zéro et ensuite affiche la somme.
5. Créez une fonction wc qui copie l'entrée tant que cette entrée n'est pas vide. Ce programme doit également afficher le nombre de lignes, de mots et de caractères affichés.

6. Créez une fonction :

```
fmapio :: (a -> b) -> IO a -> IO b
```

En fait, il existe une classe `Functor` :

```
class Functor ft where
   fmap :: (a -> b) -> ft a -> ft b
```

7. Quelle lien peut-on créer entre `IO` et `Functor` ?
En fait, la classe `Monad` est définie ainsi :

```
class Monad m where
   (>>=)  :: m a -> (a -> m b) -> m b
   return :: a -> m a
```

8. Quel lien peut-on créer entre `IO` et `Monad` ?

9. On crée le type :

```
data Erreur a = OK a | Erreur String
```

Peut-il avoir une structure de monade ?

10. Une action est une valeur comme une autre. On peut en particulier en faire une liste :

```
listeAction = [putStr "Bonjour Monde\n", writeFile "fichierTest.txt" "Bonjour système de fichier"]
```

Ces actions ne font rien en elles-mêmes : ce sont juste des valeurs. On pourrait utiliser un `do` mais il est plus pratique parfois d'utiliser la fonction `sequence_` qui est une sorte de « mappage » de do.

```
λ> :t sequence_
sequence_ :: Monad m => [m a] -> m ()
```

Redéfinissez alors `putStr` à l'aide de `sequence_`, `map` et `putChar`.

11. L'opérateur `>>` prononcé *sequence* peut être défini par :

```
m >> k = m >>= \ _ -> k
```

Quelle est la signature de `(>>)` ?

Recherche 1 - 8 Dessin

Nous allons utiliser le SVG pour tracer quelques lignes simples.
Complétez le module suivant :

```
module ImagesSVG where

import System.IO

-- Coordonnées sont des paires (x,y) d'entiers
--
--  o------> axe x
--  |
--  |
--  V
-- axe y

type Pixel = Float

data Point = Pt Pixel Pixel
```

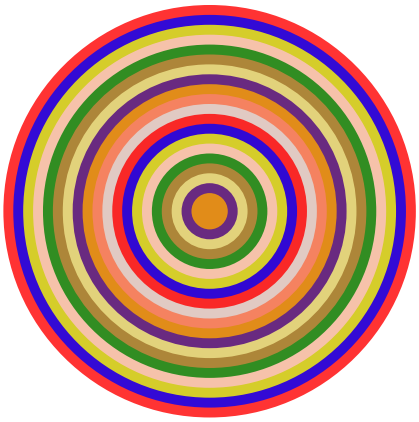
```haskell

data Vecteur = Vec Pixel Pixel

data Figure = Ligne [Point] | Ellipse Point Pixel Pixel

class Bougeable a where
  bouge  :: Vecteur -> a -> a -- translation
  symV   :: Pixel -> a -> a -- symetrie par rapport a axe vertical
  symH   :: Pixel -> a -> a -- symetrie par rapport a axe horizontal

instance Bougeable Point where
  ???

iso :: (Point -> Point) -> Figure -> Figure
-- applique une isometrie f sur une figure
????

instance Bougeable Figure where
   ???

-- si a est une instance de Bougeable, [a] instance de Bougeable ?

cercle :: Point -> Pixel -> Figure
-- definit un cercle par son centre et son rayon
???


type Graphique = String


toSVG :: Figure -> Graphique
toSVG (Ligne ps) =  "polyline points=\"" ++
                (foldr (\ (Pt x y) acc -> (show x) ++ "," ++ (show y) ++ " " ++ acc ) "\" " ps)
???
-- consulter la doc svg pour l'ellipse

data Style = Trait | Remplit | EpTrait | Transp

svgStyle :: Style -> String
svgStyle    Trait   = "stroke"
svgStyle    Remplit = "fill"
svgStyle    EpTrait = "stroke-width"
svgStyle    Transp  = "fill-opacity"

optionGr :: Style -> String -> String
optionGr st op = (svgStyle st) ++ "=\"" ++ op ++ "\" "

addOpt :: Figure -> [(Style,String)] -> Graphique
-- rajoute une liste d'options à une figure et renvoie un graphique
???

crochet :: [Graphique] -> String
-- entoure une liste de graphiques de leurs crochets xml
???

type Dim = (Int,Int)


exportSVG :: [Graphique] -> Dim -> String -> IO ()
exportSVG (grs) (w,h)  nom =
    let fichier = "<svg width=\"" ++ show w ++  "\" height=\"" ++ show h ++ "\" version=\"1.1\"\n"
                  ++ "xmlns=\"http://www.w3.org/2000/svg\" xmlns:xlink=\"http://www.w3.org/1999/xlink\">\n"
                        ++ crochet grs ++ "\n</svg>"
    in
      do
        sortie <- openFile (nom ++ ".svg") WriteMode
```

```
82          hPutStrLn sortie fichier
83          hClose sortie
84
85  cercles :: Point -> Pixel -> Pixel -> Pixel -> [Figure]
86  -- liste de n cercles concentriques et de rayons variant de rmin a rmax
87  ???
88
89  cols :: [String]
90  -- liste de couleurs svg
91  cols = cycle ["red","blue","yellow","pink", "green", "peru", "khaki", "indigo", "orange", "salmon",
            ↪  "gainsboro"]
92
93  cible :: Point -> Pixel -> Pixel -> Pixel -> [Graphique]
94  -- la cible
```

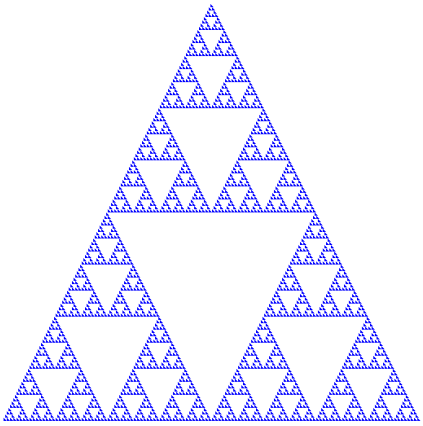
Essayez :

```
1  λ> let e = Ellipse (Pt 100 200) 50 75
2  λ> let ee = bouge (Vec 50 20) e
3  λ> let eee = symV 200  e
4  λ> exportSVG [addOpt e [(Remplit,"yellow"), (Trait,"red"), (Transp, "0.6")], addOpt ee [(Remplit,"pink"),
            ↪  (Trait,"blue"), (Transp, "0.7")], addOpt eee [(Trait, "black"), (Remplit,"gray")]] (500,500)
            ↪  "ellipse"
```

Dessinez aussi une cible de tir à l'arc :



ou un triangle se Sierpinski :



Transformations

Recherche 1 - 9

On travaille dans l'ev $\mathbb{R}^2$:

1. $u = (6,-9)$ et $v = (-10,15)$. Donner des CL de la famille (u), de la famille (u,v).
2. Que représente $\text{Vect}(\mathcal{F})$ avec $\mathcal{F} = (u,v)$?
3. Démontrer que $w = (2,-3) \in \text{Vect}(\mathcal{F})$.
4. u et v sont-ils colinéaires ?
5. $\mathcal{F}$ est-elle libre ou liée ?
6. Démontrer que $\text{Vect}(\mathcal{F}) = \text{Vect}(w)$.
7. Démontrer que $(1,2) \notin \text{Vect}\mathcal{F}$.
8. $t = (1,2)$, et $\mathcal{F}' = (w,t)$; démontrer que $\text{Vect}(\mathcal{F}') = \mathbb{R}^2$.

Recherche 1 - 10

On sait que $\mathbb{R}^{2\times 2}$, l'ensemble des matrices carrées d'ordre 2 à coefficients réels, est un espace vectoriel sur $\mathbb{R}$ avec les deux opérations : addition de deux matrices et la multiplication d'une matrice par un réel (scalaire). On considère

$$F = \left\{ \begin{pmatrix} a & b \\ b & a \end{pmatrix} \mid (a,b) \in \mathbb{R}^2 \right\}$$

Démontrer que F est un sev de E puis déterminer deux matrices M_1 et M_2 vérifiant $F = \text{Vect}(\mathcal{F})$ avec $\mathcal{F} = (M_1, M_2)$. Cette famille est-elle libre ? A-t-elle d'autres propriétés ?

Recherche 1 - 11

1. Démontrer que toute « sous famille » d'une famille libre est libre.
2. Démontrer que toute « sur famille » d'une famille liée est liée.

Recherche 1 - 12

Dire si les familles $\mathcal{F} = (u_i)$ suivantes de $\mathbb{R}^3$ sont libres ou liées :

1. $u_1 = (1,1,0), u_2 = (1,0,2)$
2. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-2)$
3. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-1)$
4. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-1), u_4 = (12,155,703)$

Recherche 1 - 13

Dire si les familles $\mathcal{F} = (u_i)$ suivantes de $\mathbb{R}^3$ sont libres ou liées :

1. $u_1 = (1,1,0), u_2 = (1,0,2)$
2. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-2)$
3. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-1)$
4. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-1), u_4 = (12,155,703)$

Recherche 1 - 14

Dire si les familles $\mathcal{F} = (u_i)$ suivantes de $\mathbb{R}^3$ sont libres ou liées :

1. $u_1 = (1,1,0), u_2 = (1,0,2)$
2. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-2)$
3. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-1)$
4. $u_1 = (1,1,0), u_2 = (1,0,2), u_3 = (0,1,-1), u_4 = (12,155,703)$

Recherche 1 - 15

1. $\mathcal{F}_1 = (u_1, u_1+u_2, u_1+u_2+u_3, u_1+u_2+u_3+u_4)$ est-elle une base de $\mathbb{R}^4$?
2. $\mathcal{F}_2 = (u_1+u_2, u_2+u_3, u_3+u_4, u_4+u_1)$ est-elle une base de $\mathbb{R}^4$?

Recherche 1 - 16

1. On note $H = \left\{(x,y,z) \in \mathbb{R}^3 \mid x+2y+3z = 0\right\}$, démontrer que H est un sev de $\mathbb{R}^3$ en déterminant u et v, deux vecteurs de $\mathbb{R}^3$, de sorte que $H = \text{Vect}(u,v)$.

2. Donner une famille génératrice de H.

3. La famille (u, v) est-elle libre ?

4. Donner une base de H.

5. Quelle est la dimension de H ?

6. On note $w = (1, 2, 3)$, démontrer que $w \notin H$.

7. On note $K = \text{Vect}(w)$. Donner une base de K.

8. Quelle est la dimension de K ?

9. Démontrer que (u, v, w) est une base de $\mathbb{R}^3$.

Recherche 1 - 17

1. Démontrer que $H = \{(x, y, z) \in \mathbb{R}^3 \mid x - z = 0\}$ est un sev de $\mathbb{R}^3$ et en donner une base.

2. Démontrer que $H = \{(x, y, z) \in \mathbb{R}^3 \mid x = 0\}$ est un sev de $\mathbb{R}^3$ et en donner une base.

3. $H = \{(x, y, z) \in \mathbb{R}^3 \mid ax + by + cz = 0\}$ où a, b et c sont des réels fixés. Dans quel cas a-t-on $H = \mathbb{R}^3$? Démontrer que H est un sev de $\mathbb{R}^3$.

Recherche 1 - 18

On travaille dans $\mathbb{R}^3$:

1. Résoudre le système $\begin{cases} x + 2y + 3z = 0 \\ 2x + y - z = 0 \end{cases}$ par la méthode de la ℓ-réduite échelonnée.

2. On note $H = \{(x, y, z) \in \mathbb{R}^3 \mid x + 2y + 3z = 0\}$ et $K = \{(x, y, z) \in \mathbb{R}^3 \mid 2x + y - z = 0\}$. Déterminer une base de $H \cap K$.

Recherche 1 - 19

On travaille dans $\mathbb{R}^3$ et on note $u = (1, 2, 2)$ et $v = (-2, 1, 2)$. Déterminer des réels a, b et c pour que $\text{Vect}(u, v) = \{(x, y, z) \in \mathbb{R}^3 \mid ax + by + cz = 0\}$.

Recherche 1 - 20

1. $\mathcal{B}$ et $\mathcal{B}'$ sont deux bases de $\mathbb{K}^n$. Qu'appelle-t-on la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$? On note P cette matrice, à quoi sert-elle ?

2. $\mathcal{B}$ et $\mathcal{B}'$ sont des bases de $\mathbb{R}^3$ et A est la matrice de $\mathcal{B}'$ dans la base $\mathcal{B}$. Démontrer que le rang de A est 3. La matrice A est-elle inversible ?

Recherche 1 - 21

$u_1 = (1, 2)$ et $u_2 = (2, 3)$ sont deux vecteurs de $\mathbb{R}^2$, $\mathcal{B} = (e_1, e_2)$ désigne la base canonique de $\mathbb{R}^2$.

1. Expliciter la base canonique.

2. Quelles sont les coordonnées de u_2 dans $\mathcal{B}$?

3. Quelles sont les coordonnées de e_1 dans $\mathcal{B}$?

4. Démontrer que la famille $\mathcal{F} = (u_1, u_2)$ est une base de $\mathbb{R}^2$.

5. Déterminer la matrice P de passage de $\mathcal{B}$ à $\mathcal{F}$.

6. Déterminer la matrice de passage de $\mathcal{F}$ à B. Que représentent les colonnes de cette matrice ?

7. Déterminer les coordonnées de u_1 dans la base $\mathcal{F}$.

8. Déterminer les coordonnées de e_1 dans la base $\mathcal{F}$.

9. $x = (-1, 5)$, déterminer les coordonnées de x dans la base $\mathcal{B}$ et dans la base $\mathcal{F}$.

10. Déterminer les coordonnées de x dans la base $\mathcal{B}' = (-3u_2, 7u_1)$.

Recherche 1 - 22

On note $\mathcal{B} = (e_1, e_2, e_3)$ la base canonique de $\mathbb{R}^3$.

1. $x = (2, 3, -5)$, déterminer les coordonnées de x dans $\mathcal{B}$.

2. $u = (1, 1, -1)$, $v = (1, -1, 2)$, $w = (2, 0, 2)$. Déterminer les coordonnées des vecteurs u, v et w dans la base B ainsi que la matrice A de la famille $F = (u, v, w)$ dans B.

3. $\text{Vect}(\mathcal{F})$ est-il un sev de $\mathbb{R}^3$?

4. Déterminer la LRÉ de A. Quel est le rang de A ?

5. Démontrer que $\mathcal{F}$ est libre.

6. Démontrer que $\mathcal{F}$ est une base de $\mathbb{R}^3$.

7. Est-il vrai que $\mathbb{R}^3 = \text{Vect}\,\mathcal{F}$?

8. Déterminer la matrice A' de la famille $\mathcal{B}$ dans la base $\mathcal{F}$.

9. Déterminer la matrice de $\mathcal{F}$ dans la base $\mathcal{F}$ et la matrice de $\mathcal{B}$ dans la base $\mathcal{B}$.

10. Déterminer les coordonnées de x dans la base $\mathcal{F}$.

11. Démontrer de plusieurs façons que $\mathcal{G} = (u, v)$ n'est pas une base de $\mathbb{R}^3$.

Recherche 1 - 23

$f \in \mathcal{L}\left(\mathbb{R}^3\right)$, $\mathcal{B} = (e_1, e_2, e_3)$ est une base de $\mathbb{R}^3$, $A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \boldsymbol{Mat}_{\mathcal{B}}(f)$.

1. f est-il un endomorphisme de $\mathbb{R}^3$?

2. Quelles sont les coordonnées de e_1 dans la base $\mathcal{B}$?

3. Quelles sont les coordonnées de $f(e_2)$ dans la base $\mathcal{B}$?

4. Quelles sont les coordonnées de $f(x)$ dans la base $\mathcal{B}$ si $x = e_1 + e_2 - 2e_3$?

5. Qu'est $\ker f$?

Recherche 1 - 24

On travaille dans l'espace vectoriel $\mathbb{R}^3$ rapporté à la base canonique $\mathcal{B} = (e_1, e_2, e_3)$. On donne :

$$\text{la lré de } \begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 1 \\ -1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

On considère la famille $\mathcal{F} = (u_1, u_2, u_3)$ avec

$$u_1 \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}_{\mathcal{B}}, u_2 \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}_{\mathcal{B}}, u_3 \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}_{\mathcal{B}}$$

1. Déterminer la matrice A de la famille $\mathcal{F}$ dans la base B.

2. Donner la lré de A.

3. On note f l'endomorphisme de $\mathbb{R}^3$ dont la matrice dans la base $\mathcal{B}$ est A.

- **i.** Que représentent les colonnes de A ?
- **ii.** Déterminer $\mathcal{B}$ la matrice de la famille $(f(u_1), f(u_2), f(u_3))$ dans la base $\mathcal{B}$.
- **iii.** On note $\ker(f)$ le noyau de f, définir $\ker(f)$.
- **iv.** On note $x = (x_1, x_2, x_3)$ un vecteur quelconque de $\mathbb{R}^3$. Déterminer les coordonnées de $f(x)$ dans la base $\mathcal{B}$.
- **v.** Déterminer une base de $\ker(f)$, on notera $\mathcal{F}_{\ker}$ cette famille.
- **vi.** Donner une famille génératrice de $\boldsymbol{Im}(f)$.
- **vii.** Déterminer une base de $\boldsymbol{Im}(f)$. On notera $\mathcal{F}_{\boldsymbol{Im}}$ cette famille.

4. On note $\mathcal{B}' = (e'_1, e'_2, e'_3)$ la famille obtenue en juxtaposant le ou les vecteurs de $\mathcal{F}_{\boldsymbol{Im}}$ et le ou les vecteurs de $\mathcal{F}_{\ker}$. On admet que $\mathcal{B}'$ est une base (il suffit de vérifier qu'elle est libre) ; x'_1, x'_2 et x'_3 désignent les coordonnées d'un vecteur x de $\mathbb{R}^3$ dans cette base.

- **i.** Déterminer H la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$.
- **ii.** On note U la matrice de f relativement ou dans la base $\mathcal{B}'$, $U = \boldsymbol{Mat}_{\mathcal{B}'}(f)$, exprimer U en fonction des matrices H et A.

5. On note g l'application de $\mathbb{R}^3$ dans $\mathbb{R}^3$ définie par :

$$x\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}_{\mathcal{B}'} \to g(x)\begin{pmatrix} x'_1 \\ x'_2 \\ 0 \end{pmatrix}_{\mathcal{B}'}$$

et g est manifestement un endomorphisme de $\mathbb{R}^3$.

- **i.** Déterminer V la matrice de g dans la base $\mathcal{B}'$.
- **ii.** Déterminer une base de $\ker(g)$.
- **iii.** Déterminer une base de $\boldsymbol{Im}(g)$.
- **iv.** Exprimer en fonction de V et de H la matrice de g dans la base $\mathcal{B}$.
- **v.** Exprimer à l'aide "de lettres" la matrice de $g \circ f$ dans la base $\mathcal{B}'$.

Recherche 1 - 25

$\mathcal{B} = (u, v, w)$ est une base de l'e.v. $\mathbb{R}^3$, f est un automorphisme de $\mathbb{R}^3$ et $A = \boldsymbol{Mat}_{\mathcal{B}}(f)$.

1. Démontrer que $\mathcal{F} = (f(u), f(v), f(w))$ est une famille libre et donc que c'est une base de $\mathbb{R}^3$.

2. Déterminer la matrice de f dans la base $\mathcal{F}$.

Recherche 1 - 26

$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$ est la matrice de l'application linéaire φ de $\mathbb{F}_2^3$ dans $\mathbb{F}_2^5$ relativement aux bases canoniques.

1. Démontrer que φ est injective. Déterminer $\ker\varphi$.

2. Déterminer une base de $\boldsymbol{Im}\varphi$. Quelle est la dimension de $\boldsymbol{Im}\varphi$? φ est-elle surjective ? Donner une matrice génératrice de $\boldsymbol{Im}\varphi$ dans la base canonique de $\mathbb{F}_2^5$.

3. Quelle est la dimension de $\boldsymbol{Im}\varphi$?

Recherche 1 - 27 Reconnaître une fonction linéaire

On rappelle qu'une fonction linéaire est une fonction totale f sur un $\mathbb{K}$-espace vectoriel E qui vérifie

$$\forall x, y \in E^2. \forall \lambda \in \mathbb{K}. f(\lambda x + y) = \lambda f(x) + f(y)$$

Est-ce que l'application : $\begin{array}{ccc} \mathbb{R}^3 & \longrightarrow & \mathbb{R}^3 \\ \vec{x} & \longmapsto & \vec{x} + \vec{a} \end{array}$ avec $\vec{a}$ un vecteur fixe est linéaire ?

Recherche 1 - 28

$f \in \mathcal{L}(\mathbb{R}^3)$, $\mathcal{B} = (e_1, e_2, e_3)$ est une base de $\mathbb{R}^3$, $A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \boldsymbol{Mat}_{\mathcal{B}}(f)$.

1. f est-il un endomorphisme de $\mathbb{R}^3$?

2. Quelles sont les coordonnées de e_1 dans la base $\mathcal{B}$?

3. Quelles sont les coordonnées de $f(e_2)$ dans la base $\mathcal{B}$?

4. Quelles sont les coordonnées de $f(x)$ dans la base $\mathcal{B}$ si $x = e_1 + e_2 - 2e_3$?

5. Qu'est $\ker f$?

Recherche 1 - 29

$f \in \mathcal{L}(\mathbb{R}^p, \mathbb{R}^n)$, démontrer :

$$\ker f = \{0_{\mathbb{R}^p}\} \Leftrightarrow f \text{ est injective}$$

Recherche 1 - 30 détermination d'une rotation à partir d'une matrice

On donne la matrice $A = \frac{1}{3}\begin{pmatrix} 1 & 2 & 2 \\ -2 & 2 & -1 \\ -2 & -1 & 2 \end{pmatrix}$ d'une certaine transformation f de $\mathbb{R}^3$ dans lui-même. Décrivez f le plus précisément possible.

Faites de même avec la matrice $B = \frac{1}{3}\begin{pmatrix} 2 & 1 & 2 \\ -2 & 2 & 1 \\ 1 & 2 & -2 \end{pmatrix}$

Recherche 1 - 31 matrice d'une rotation

On considère la rotation u d'angle $\frac{\pi}{4}$ autour de l'axe dirigé par le vecteur $\overrightarrow{v_1}$ de coordonnées $\left(\frac{1}{3}, \frac{2}{3}, \frac{2}{3}\right)$ dans la base canonique $\mathscr{B}_0$.

1. Montrez que $\overrightarrow{v_2}$ de coordonnées $\left(\frac{2}{3}, -\frac{2}{3}, \frac{1}{3}\right)$ dans $\mathscr{B}_0$ est orthogonal à $\overrightarrow{v_1}$.
2. Déterminez un vecteur $\overrightarrow{v_3}$ tel que $(\overrightarrow{v_1}, \overrightarrow{v_2}, \overrightarrow{v_3})$ soit une base orthonormée directe.
3. Déterminez $\mathrm{mat}_{\mathcal{B}_0} u$.

Recherche 1 - 32 composée de rotations

Dans $\mathbb{R}^3$, soit R_x la rotation autour de l'axe dirigé par $\overrightarrow{e_1}$ et d'angle $\frac{\pi}{2}$, soit R_y la rotation autour de l'axe dirigé par $\overrightarrow{e_2}$ et d'angle $\frac{\pi}{2}$ et soit R_z la rotation autour de l'axe dirigé par $\overrightarrow{e_3}$ et d'angle $\frac{\pi}{2}$.
Décrivez $R_x \circ R_z$.

Recherche 1 - 33

On travaille dans $\mathbb{R}^3$ et $\mathcal{B} = (i, j, k)$ est une $BOND$. Dans ce qui suit, pour les représentations, on choisira une mesure d'angle peu différente de $\frac{\pi}{6}$ ou $\frac{\pi}{3}$.

1. On considère la rotation $\mathrm{Rot}_1(i, \alpha)$ qui transforme $\mathcal{B}$ en $\mathcal{B}_1 = (i_1, j_1, k_1)$, représenter $\mathcal{B}$ et $\mathcal{B}_1$ et déterminer la matrice R_1 de $\mathrm{Rot}_1(i, \alpha)$ dans la base $\mathcal{B}$.
2. On considère la rotation $\mathrm{Rot}_2(j, \beta)$ qui transforme $\mathcal{B}$ en $\mathcal{B}_2 = (i_2, j_2, k_2)$, représenter $\mathcal{B}$ et $\mathcal{B}_2$ et déterminer la matrice R_2 de $\mathrm{Rot}_2(j, \beta)$ dans la base $\mathcal{B}$.
3. On considère la rotation $\mathrm{Rot}_3(k, \gamma)$ qui transforme $\mathcal{B}$ en $\mathcal{B}_3 = (i_3, j_3, k_3)$, représenter $\mathcal{B}$ et $\mathcal{B}_3$ et déterminer la matrice R_3 de $\mathrm{Rot}_3(k, \gamma)$ dans la base $\mathcal{B}$.
4. Déterminer la matrice de $\mathrm{Rot}_3(k, \gamma)$ dans la base $\mathcal{B}_3$.
5. Déterminer la matrice de $\mathrm{Rot}_3(k, \gamma)$ dans la base $\mathcal{B}_2$.
6. Déterminer la matrice de $\mathrm{Rot}_2(j, \beta)$ dans la base $\mathcal{B}_2$.
7. Déterminer la matrice de $\mathrm{Rot}_2(j, \beta)$ dans la base $\mathcal{B}_3$.
8. On considère la rotation $\mathrm{Rot}_4(j_3, \varphi)$ qui transforme la base $\mathcal{B}_3 = (i_3, j_3, k_3)$ en la base $\mathcal{B}_4 = (i_4, j_4, k_4)$.
   i. Déterminer la matrice de $\mathrm{Rot}_4(j_3, \varphi)$ dans la base $\mathcal{B}_3$.
   ii. Déterminer la matrice de $\mathrm{Rot}_4(j_3, \varphi)$ dans la base $\mathcal{B}$.
9. Déterminer la matrice $\mathrm{Rot}_1 \circ \mathrm{Rot}_2 \circ \mathrm{Rot}_3$ dans la base $\mathcal{B}_2$.
10. Détertminer la matrice de $\mathrm{Rot}_2 \circ \mathrm{Rot}_4 \circ \mathrm{Rot}_2 \circ \mathrm{Rot}_3 \circ \mathrm{Rot}_2$ dans la base $\mathcal{B}_4$.
11. Déterminer la matrice de passage de $\mathcal{B}_2$ à $\mathcal{B}_4$.
12. Déterminer la matrice de passage de $\mathcal{B}_4$ à $\mathcal{B}_3$.

Recherche 1 - 34

$\mathcal{B} = (i, j, k)$ est une **BOND** de $\mathbb{R}^3$, r_1 est la rotation d'axe j et d'angle de mesure $\alpha \in \left[0, \frac{\pi}{2}\right]$ qui transforme la BOND $\mathcal{B}$ en la BOND $\mathcal{B}_1 = (i_1, j_1, k_1)$.

1. Faire un dessin clair représentant $\mathcal{B}$ et $\mathcal{B}_1$.
2. Donner R la matrice de r_1 dans la base $\mathcal{B}$.

3. r_2 est la rotation d'axe i_1 qui transforme la BOND $\mathcal{B}_1$ en la BOND $\mathcal{B}_2$ et r_3 est la rotation d'axe k qui transforme la BOND $\mathcal{B}$ en la BOND $\mathcal{B}_3$. On note S la matrice de r_2 dans la base $\mathcal{B}_1$ et T la matrice de r_3 dans la base $\mathcal{B}$. Donner la matrice de $r_2 \circ r_1 \circ r_3 \circ r_1 \circ r_2$ dans la base B_3.

Recherche 1 - 35

$\mathcal{B} = (i, j, k)$ est une **BOND** de $\mathbb{R}^3$, on considère les vecteurs u et v :

$$u \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}_{\mathcal{B}} \quad v \begin{pmatrix} -2 \\ 0 \\ 1 \end{pmatrix}_{\mathcal{B}}$$

et il est manifeste que u et v sont orthogonaux. Déterminer des vecteurs a, b et c vérifiant : a colinéaire à u, c colinéaire à v et (a, b, c) est une BOND.

Recherche 1 - 36 Théorème de Pythagore

Démontrez que u et v sont orthogonaux si, et seulement si,

$$||u + v||^2 = ||u||^2 + ||v||^2$$

Recherche 1 - 37

1. Démontrez la propriété 1 - 19 page 18.
2. Démontrez la propriété 1 - 20 page 19. Pour cela, vous commencerez par développer $||u + xv||^2$ avec x un réel quelconque. Vous obtenez alors un polynôme du second degré en x : quel est son signe ? Qu'en déduit-on concernant son discriminant ? Concluez.
3. Démontrez la propriété 1 - 21 page 19

Recherche 1 - 38 Niveaux de gris

Un pixel, en général, est représenté par un vecteur (R,V,B), chacune des composante pouvant prendre 256 valeurs entre 0 et 255. Cela donne un échantillon de 2^{24} couleurs...
L'espace des couleurs est alors l'intérieur d'un cube de côté de longueur 256.
Lors de la conversion en niveau de gris, on veut obtenir un nouveau vecteur gris (i.e. avec trois composantes égales) le plus « proche » possible du pixel en couleur.

1. Donnez une base de « l'espace des gris ».
2. Expliquez intuitivement quelle est la plus petite distance entre le point de coordonnées (R,V,B) et la droite des gris dans un repère bien choisi.
3. Déduisez-en une formule permettant de convertir une image en niveau de gris connaissant sa matrice « en couleur ».

Recherche 1 - 39 Calcul manuel de la SVD

On suppose que la SVD d'une matrice A de taille $m \times n$ s'écrit $U \times S \times {}^tV$.

1. Quelles sont les dimensions de U et V ?
2. Montrez que $A \times {}^tA$ est diagonalisable et que les colonnes de U en forment une base orthonormée de vecteurs propres.
   Montrez de même que les colonnes de V forment une base orthonormée de vecteurs propres de ${}^tA \times A$.
3. Calculez (à la main...) la SVD de $A = \begin{pmatrix} 2 & -2 \\ 1 & 1 \end{pmatrix}$.
   On aura besoin d'orthogonaliser la base de vecteurs propres obtenue...
4. Shoot again avec $\begin{pmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{pmatrix}$

Recherche 1 - 40 Transformations affines

Sans trop rentrer dans les détails, on désignera par espace affine un ensemble de points « associés » à un espace vectoriel. C'est ce qui est concrètement utilisé en infographie : on peut effectuer des opérations sur les points et les vecteurs.
Par exemple, dans un **repère** $(O;\vec{\imath},\vec{\jmath})$:

- $A+\overrightarrow{AB}=B$
- $B-A=\overrightarrow{AB}$
- $A=O+\overrightarrow{OA}$
- $2A+3B=O+2\overrightarrow{OA}+3\overrightarrow{OB}$
- etc.

On peut composer des fonctions linéaires (vectorielles) dans une base en multipliant les matrices de ces applications dans cette base mais quid des transformations affines qui ne sont pas vectorielles ?
Il existe une « ruse » que nous ne détaillerons pas mais qui est largement utilisé en infographie (SVG, Blender, Illustrator, etc.) qui consiste à « plonger » les objets 2D dans la 3D mais en gardant la troisième coordonnées fixe.
Ainsi, un point de coordonnées (x,y) dans $(O;\vec{\imath},\vec{\jmath})$ sera représenté par la matrice $\begin{pmatrix}x\\y\\1\end{pmatrix}$

1. Comment sera représentée une symétrie d'axe (Ox) ? d'axe (Oy) ?
2. Comment sera représentée une rotation de centre O et d'angle α ?
3. Comment sera représentée une translation de vecteur $\vec{u}$ de coordonnées (v_x,v_y) ?
4. Comment sera représentée la fonction qui tranforme le point $P(x,y)$ en le point $P'(\lambda x,y)$? En le point $P''(x,\lambda y)$?
5. Étudiez la doc SVG http://www.w3.org/TR/SVG/coords.html.
6. Une couleur peut être représentée par un vecteur RVBA de quatre dimensions. Une transformation de couleurs est alors représentée par une matrice.
   Quel est l'action de cette matrice :

$$\begin{pmatrix}1/3 & 1/3 & 1/3 & 0\\ 1/3 & 1/3 & 1/3 & 0\\ 1/3 & 1/3 & 1/3 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

7. Étudiez http://www.w3.org/TR/SVG/filters.html#feColorMatrixElement
8. Faites de jolis dessins...

Expliquez, développez, améliorez le module suivant :

```haskell
module PicturesSVG where

import System.IO

data Picture
 = Img Image
 | Sur Picture Picture
 | Acote Picture Picture
 | Sous Picture Picture
 | SymH Picture
 | SymV Picture
 | Negative Picture
   deriving (Show)

type Point = (Int,Int)

data Image = Image Name Point
             deriving (Show)

data Name  = Name String
             deriving (Show)
```

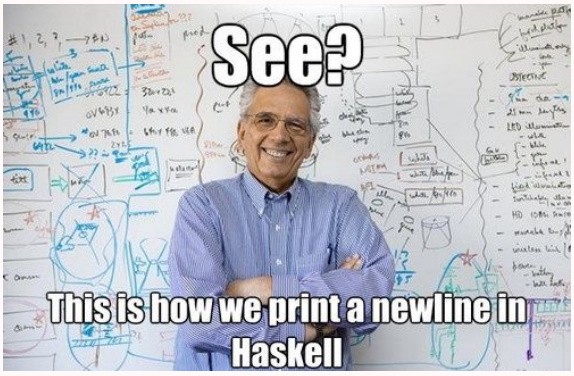
```haskell

sur, aCote, sous :: Picture -> Picture -> Picture

sur  = Sur
aCote = Acote
sous   = Sous


symH, symV, negative :: Picture -> Picture

symH (Sur pic1 pic2)  = (symH pic2) `Sur` (symH pic1)
symH (Acote pic1 pic2) = (symH pic1) `Acote` (symH pic2)
symH (Sous pic1 pic2)   = (symH pic1) `Sous` (symH pic2)
symH pic                = SymH pic

symV (Sur pic1 pic2)  = (symV pic1) `Sur` (symV pic2)
symV (Acote pic1 pic2) = (symV pic2) `Acote` (symV pic1)
symV (Sous pic1 pic2)   = (symV pic1) `Sous` (symV pic2)
symV pic                = SymV pic

negative = Negative

invertColour = Negative

img :: Image -> Picture

img = Img


width,height :: Picture -> Int

width (Img (Image _ (x,_))) = x
width (Sur pic1 pic2)     = max (width pic1) (width pic2)
width (Acote pic1 pic2)    = (width pic1) + (width pic2)
width (Sous pic1 pic2)      = max (width pic1) (width pic2)
width (SymH pic)           = width pic
width (SymV pic)           = width pic
width (Negative pic)        = width pic

height (Img (Image _ (x,y))) = y
height (Sur pic1 pic2)     = (height pic1) + (height pic2)
height (Acote pic1 pic2)    = max (height pic1) (height pic2)
height (Sous pic1 pic2)      = max (height pic1) (height pic2)
height (SymH pic)           = height pic
height (SymV pic)           = height pic
height (Negative pic)        = height pic


data Filter = Filter {fH, fV, neg :: Bool}
              deriving (Show)

newFilter = Filter False False False

data Basic = Basic Image Point Filter
             deriving (Show)


flatten :: Point -> Picture -> [Basic]

flatten (x,y) (Img image)        = [Basic image (x,y) newFilter]
flatten (x,y) (Sur pic1 pic2)  = flatten (x,y) pic1 ++ flatten (x, y + height pic1) pic2
flatten (x,y) (Acote pic1 pic2) = flatten (x,y) pic1 ++ flatten (x + width pic1 , y) pic2
flatten (x,y) (Sous pic1 pic2)   = flatten (x,y) pic1 ++ flatten (x,y) pic2
flatten (x,y) (SymH pic)        = map symFH $ flatten (x,y) pic
flatten (x,y) (SymV pic)        = map symFV $ flatten (x,y) pic
```

```haskell
flatten (x,y) (Negative pic)     = map symNeg $ flatten (x,y) pic


symFH (Basic img (x,y) f@(Filter {fH=boo}))   = Basic img (x,y) f{fH = not boo}
symFV (Basic img (x,y) f@(Filter {fV=boo}))   = Basic img (x,y) f{fV = not boo}
symNeg (Basic img (x,y) f@(Filter {neg=boo})) = Basic img (x,y) f{neg = not boo}


convert :: Basic -> String

convert (Basic (Image (Name name) (width, height)) (x,y) (Filter fH fV neg))
  = "\n  <image x=\"" ++ show x ++ "\" y=\"" ++ show y ++ "\" width=\"" ++ show width ++ "\" height=\"" ++
    show height ++ "\" xlink:href=\"" ++ name ++ "\"" ++ symPart ++ negPart ++ "/>\n"
        where
          symPart
              = if      fH && not fV
                then " transform=\"translate(0," ++ show (2*y + height) ++ ") scale(1,-1)\" "
                else if fV && not fH
                then " transform=\"translate(" ++ show (2*x + width) ++ ",0) scale(-1,1)\" "
                else if fV && fH
                then " transform=\"translate(" ++ show (2*x + width) ++ "," ++ show (2*y + height) ++ ")
                       ↪  scale(-1,-1)\" "
                else ""
          negPart
              = if neg
                then " filter=\"url(#negative)\""
                else ""



render :: Picture -> IO ()

render pic
 =
   let
       picList = flatten (0,0) pic
       svgString = concat (map convert picList)
       newFile = preamble ++ svgString ++ postamble
   in
     do
       outh <- openFile "svgOut.xml" WriteMode
       hPutStrLn outh newFile
       hClose outh



preamble
 = "<svg width=\"100%\" height=\"100%\" version=\"1.1\"\n" ++
   "xmlns=\"http://www.w3.org/2000/svg\" xmlns:xlink=\"http://www.w3.org/1999/xlink\">\n" ++
   "<defs>" ++
   "<filter id=\"negative\">\n" ++
   "<feColorMatrix  in=\"SourceGraphic\" type=\"matrix\"\n"++
   "values=\"-1 0  0  0  0  0 -1  0  0  0  0  0 -1  0  0  1  1  1  0  0\" />\n" ++
   "</filter>\n"++
   "<filter id=\"greyscale\">\n" ++
   "<feColorMatrix in=\"SourceGraphic\" type=\"matrix\"\n" ++
   "values=\".33 .33 .33 0 0 .33 .33 .33 0 0 .33 .33 .33 0 0 0 0 0 1 0\"/>\n" ++
   "</filter>\n" ++
   "</defs>\n"

postamble
 = "\n</svg>\n"


conan = Img $ Image (Name "Conan.jpg") (357,266)

```

```haskell
test = (conan `aCote` (symV $ negative conan))
                      `sur`
       ((symH $ negative conan) `aCote` (symH $ symV conan))

test2 = test `aCote` symV test
```

THÈME

Pourquoi apprendre Haskell fait de moi un meilleur programmeur en Python ?



Comment notre travail sur les types et la programmation fonctionnelle peuvent nous aider à manipuler des images en Python et tout cela au grand dam' de GVR ?

1 Matrices

1 1 Préliminaire : Python est un langage fonctionnel ;-)

Vous avez une liste de nombres et une deuxième liste de nombres. Vous voulez retourner la liste des indices des éléments de la seconde dans la première.
Par exemple :

```
In [7]: recherche_liste([1,4,1,4,6,5,5,5,4,2,3], [1,3,5])
Out[7]: [[0, 2], [10], [5, 6, 7]]
```

Rien de plus simple :

```
recherche_liste = lambda l,e : [[ind for ind,el in enumerate(l) if el == elf] for elf in
    ↪  e]
```

Du Haskell ? Ben non, du Python :D
C'est quoi `enumerate` ?

```
In [16]: enumerate(['jan', 'fev', 'mar'])
Out[16]: <enumerate at 0x7f3e9089a8b8>

In [17]: list(enumerate(['jan', 'fev', 'mar']))
Out[17]: [(0, 'jan'), (1, 'fev'), (2, 'mar')]
```

On retrouve en Python `map`, `filter` et `reduce` qui est un synonyme de `foldl`.
Bref, pour bien programmer :

```
from fonctionnel import *
```

1 2 Fabriquons nos outils

1 2 1 Création d'une classe « Matrice »

Dans cette section, nous créerons nos matrices en donnant leur dimension et la fonction définissant leurs coefficients en fonction de leurs indices...comme nous l'avons fait en Haskell...sauf que c'est moins joli et mois contrôlable...

```
class Mat:
    """ une matrice sous la forme Mat([nb lignes, nb cols], fonction(i,j))"""

    def __init__(self, dim, f):
        self.F = f   # fonction (i,j) -> coeff en position i,j
        self.D = dim # liste [nb de lignes, nb de cols]
```

Par exemple, $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ sera creée avec :

```
>>> M = Mat([3,2],lambda i,j : 2*i + (j + 1))
```

On a quand même l'habitude de rentrer une matrice comme une liste de lignes. On va donc créer une fonction qui permet de convertir une liste de lignes en matrice :

```
def list2mat(mat):
    r,c = len(mat),len(mat[0])
    return Mat([r,c], lambda i,j : mat[i][j])
```

La matrice précédente aurait donc pu être définie par :

```
>>> M = list2mat([[1,2],[3,4],[5,6]])
```

```
>>> M.D # dimension de la matrice
[3, 2]
>>> M.F(0,1) # coefficient à la position 0,1
2
```

Recherche

Écrivez une fonction qui renvoie une matrice nulle de taille $n \times m$ puis une autre qui renvoie la matrice identité de taille n.

Nous rajoutons quelques méthodes habituellement définies pour les objets structurés de ce type :

```
    def __getitem__(self,cle):
        """ permet d'obtenir Mij avec M[i,j] """
        return self.F(*cle)

    def __iter__(self):
        """ pour itérer sur la liste des coefficients donnés par colonnes """
        [r,c] = self.D
        for j in range(c):
            for i in range(r):
                yield self.F(i,j)
```

L'emploi du mot-clé `yield` au lieu de `return` permet de ne *retourner* `self.F(i,j)` que lorsque cela est demandé. Ainsi la liste de tous les coefficients n'est pas créée en entier systématiquement ce qui économise la mémoire.
Nous pouvons ainsi créer une méthode qui va permettre un joli affichage :

```
    def __str__(self):
        """ joli affichage d'une matrice"""
        [r,c],f = self.D, self.F
        lmax = len(str(max(iter(self)))) + 1
        s = '\n'.join( (' '.join('{0:>.{l}G}'.format(f(i,j),l=lmax) for j in range(c)))
                ↪  for i in range(r))
        return s

    def __repr__(self):
        """ représentation dans le REPL"""
        return str(self)
```

La méthode `join` est à employer systématiquement pour la concaténation des chaînes de caractères : elle est beaucoup plus efficace que son pendant sous forme de boucle..

```
>>> M
1 2
3 4
5 6
```

Recherche

Créez deux fonctions qui créent un itérateur respectivement sur les lignes et sur les colonnes. Par exemple laliste des coefficients de la première colonne de M sera donnée par :

```
>>> [i for i in M.col(0)]
[1, 3, 5]
```

1 2 2 Transposée d'une matrice

La transposée d'une matrice $(M_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant m}}$ est égale à la matrice $(M_{ji})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant m}}$. Attention aux indices et aux nombre de lignes et de colonnes !

Recherche

Créez une fonction qui renvoie la transposée d'une matrice donnée en argument.

```
>>> M.transpose()
1 3 5
2 4 6
```

1 2 3 Somme de matrices

On voudrait créer une méthode qui prend comme arguments deux matrices et renvoie leur somme. Il faudra vérifier que les matrices à additionner sont de bonnes tailles : on utilisera la fonction `assert` qui est suivie d'une condition à vérifier et d'un message à affichher en cas de problème.
Pour pouvoir utiliser le symbole + par la suite, on va nommer notre méthode __add__ :

```
    def __add__(self,other):
        """ somme de deux matrices : utilisation du symbole +"""
        assert self.D == other.D, "tailles incompatibles"
        return Mat(self.D, lambda i,j : self.F(i,j) + other.F(i,j))
```

Par exemple, avec la matrice M précédemment introduite :

```
>>> M + M
  2   4
  6   8
 10  12
```

Recherche

On définit de même __neg__ pour l'opposé et __sub__ pour la soustraction : faites-le !

1 2 4 Produit par un scalaire

On voudrait obtenir la matrice $k \cdot M$ à partir d'une matrice M et d'un scalaire k.

```
>>> A = Mat([2,3], lambda i,j : 3*i + j)
>>> A
 0  1  2
 3  4  5
>>> A.prod_par_scal(5)
  0   5  10
 15  20  25
```

Recherche

Écrivez une implémentation de la méthode `prod_par_scal`

1 2 5 Produit de matrices

Soit $A = (a_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant i\leqslant m}}$ et $B = (b_{ij})_{1\leqslant i\leqslant p}^{1\leqslant i\leqslant m}$. Alors $A \times B = C$ avec $C = (c_{ij})_{1\leqslant i\leqslant p}^{1\leqslant i\leqslant n}$ et

$$\forall (i,j) \in \mathbb{N}_n^* \times \mathbb{N}_p^*, \quad c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

C'est l'algorithme habituel trois boucles imbriquées).

Nous allons l'abstraire d'un petit niveau : chaque coefficient c_ij est en fait égal au produit scalaire de la ligne i de A et de la colonne j de B.
Commençons donc par construire une fonction qui calcule le produit scalaire de deux itérables. Pour cela, nous allons utiliser une fonction extrêmement importante qui vient du monde de la programmation fonctionelle : `map`.
`map(fonc, iterable(s))` renvoie un itérateur de type `map` mais où chaque élément sera remplacé par son image par la fonction (si c'est une fonction de une variable) ou par l'image combinée des itérables si c'est une fonction de plusieurs variables.
Par exemple :

```
>>> map(lambda x: x*x, [1,2,3,4])
<map at 0x7fe8307f2ef0>
>>> [i for i in m]
[1 4 9 16]
```

arghhh : l'objet crée est de type `map` et on a son adresse mais on ne sait pas ce qu'il y a dedans...

```
>>> m = map(lambda x: x*x, [1,2,3,4])
>>> [i for i in m]
[1 4 9 16]
```

Mais atention ! m a maintenant été « consommé ». Si on en redemande :

```
>>> [i for i in m]
[]
```

Si on veut en garder une trace on peut utiliser par exemple `list` :

```
>>> m = map(lambda x: x*x, [1,2,3,4])
>>> l = list(m)
>>> l
[1 4 9 16]
```

Avec une fonction de deux variables et deux itérables :

```
>>> m = map(lambda x, y: x + y, [1,2,3,4], [-1,-2,-3,-4])
>>> list(m)
[0, 0, 0, 0]
```

On peut aller plus vite en utilisant les opérateurs arithmétiques écrits en notation préfixée dans la bibliothèque `operator` :

```
from operator import mul
>>> m = map(mul, [1,2,3,4], [-1,-2,-3,-4])
>>> list(m)
[-1, -4, -9, -16]
```

Pour notre produit scalaire, nous utilisons également la classique fonction `sum`.

Recherche

Créez une fonction `prod_scal`.
Utilisez `prod_scal` pour écrire en une ligne la matrice produit de deux matrices données en arguments.
Vous créerez ainsi une méthode `prod_mat(self,other)` en ajoutant à la ligne précédente une ligne vérifiant que les formats sont corrects.

Pour utiliser le symbole $*$ pour le produit d'une matrice par une autre matrice ou un scalaire, on crée dans notre classe une méthode `__mul__`. On utilisera la fonction `type` qui teste le type d'un objet.

```
  def __mul__(self,other):
        """ produit d'une matrice par un scalaire ou une matrice : utilisation du
            ↪ symbole *"""
        if Mat == type(other):
            return self.prod_mat(other)
        else:
            return self.prod_par_scal(other)
```

1 2 6 Comparaison de plusieurs produits matriciels

Notre produit n'est pas si mal : il s'écrit très simplement et est assez efficace. Comparons-le avec d'autres implémentations.

Avec les outils standards de Python

D'abord une implémentation classique en introduisant une liste Python et des matrices du type `array` de `numpy`. On travaillera par exemple avec des tableaux d'entiers (l'argument `dtype = int`) et on utilisera les commandes `shape` et `zeros` de `numpy`.

```
import numpy as np

def pymatmatprod(A, B):
    '''
    Multiplication matricielle avec les outils Python usuels
    '''
    ra, ca = A.shape
    rb, cb = B.shape
    assert ca == rb, "Tailles incompatibles"
    C = np.zeros((ra, cb),dtype = int)
    for i in range(ra):
        Ai,Ci = A[i],C[i]
        for j in range(cb):
            for k in range(ca):
                Ci[j] += Ai[k] * B[k, j]
    return C
```

Avec Cython

Cython (http://cython.org/) est un langage dont la syntaxe est très proche de celle de Python mais c'est un langage qui permet de générer des exécutables compilés et d'utiliser un style de programmation proche du C. Cela permet d'optimiser grandement Python.
Le logiciel de calcul formel Sage http://www.sagemath.org/ est principalement écrit en Cython pour gagner en efficacité.
Voyons un exemple en œuvre. L'usage conjoint de Python et Cython est grandement facilité par le travail dans l'environnement iPython.

```
In [1]: %load_ext cythonmagic

In [2]:
%%cython
cimport cython
import numpy as np
cimport numpy as np

cdef long[:, :] matprodC( long[:, :]  A, long[:, :]  B):
    '''
    multiplication matricielle via cython et les array de numpy
    '''
    cdef:
        int i, j, k
        int ra = A.shape[0]
        int ca = A.shape[1]
```

```
        int rb = B.shape[0]
        int cb = B.shape[1]
        long [:, :] C
        long [:] Ai, Ci

    assert ca == rb, 'Tailles non compatibles'

    C = np.zeros((ra, cb),dtype=int)
    for i in range(ra):
        Ai,Ci = A[i],C[i]
        for j in range(cb):
            for k in range(ca):
                Ci[j] = Ci[j] + Ai[k] * B[k, j]
    return C

def matprod(long [:, :]  A, B):
    return matprodC(A, B)
```

On remarque qu'écrire en Cython revient un peu à écrire en Python mais en déclarant des variables comme en C.
iPython permet également de comparer facilement les temps de calcul avec la commande magique `%timeit`.
On commence par créer une matrice de taille 200×200 puis on va demander son coefficient [100,100] :

```
In [3]: A = np.ones((200,200),dtype = int)
```

On compare alors le produit via Python, Cython et la multiplication de `numpy` :

```
In [4]: %timeit matprod(A,A)[100,100]
100 loops, best of 3: 16.3 ms per loop

In [5]: %timeit pymatmatprod(A,A)[100][100]
1 loops, best of 3: 8.35 s per loop

In [6]: %timeit np.dot(A,A)[100,100]
100 loops, best of 3: 8.94 ms per loop
```

Notre fonction en Cython est 500 fois plus rapide!!! Elle est très légèrement plus lente que le produit de `numpy`.
Et notre implémentation personnelle avec la classe `Mat` ?

```
In [7]: A = Mat([200,200],lambda i,j:1)

In [8]: %timeit (A*A)[100,100]
10000 loops, best of 3: 79.6 µs per loop
```

Trop fort ! Nous battons tout le monde à plate couture ! Nous sommes 100 fois plus rapides que `numpy` !!!
...et nous avons bien le bon résultat :

```
In [9]: (A*A)[100,100]
Out[9]: 200
```

Bon, ça va pour un seul produit. Pour calculer une puissance 200, ce sera moins magique...

127 Puissances d'une matrice

Par exemple, avec $J = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right)$:

```
>>> J = Mat([2,2], lambda i,j : 1)
>>> J ** 5
 16  16
 16  16
```

En effet, $J^n = 2^{n-1}J$ est un résultat classique. On utilise un algorithme d'exponentiation rapide. Par exemple, en voici une version récursive rapide à écrire qui se nomme __pow__ afin d'utiliser le symbole ** :

```
    def __pow__(self,n):
        r = self.D[0]
        if n == 0:
            return unite(r)
        def pui(m,k,acc):
            if k == 0:
                return acc
            return pui((m*m),k//2,acc if k % 2 == 0 else (m*acc))
        return pui(self,n,unite(r))
```

Recherche Déterminez une version impérative de la méthode précédente.

2 Rappels sur l'inversion des matrices

208 Matrice carrée inversible

Soit $M \in \mathbb{A}^{n\times n}$. M est inversible (régulière) si, et seulement si, il existe une matrice N dans $\mathbb{A}^{n\times n}$ telle que :

$$M \times N = N \times M = \mathbb{I}_n$$

On note alors $N = M^{-1}$.

On remarque (n'est-ce pas) que, d'après cette définition, $(M^{-1})^{-1} = M$.

Lorsque nous aurons étudier les déterminants, nous pourrons démontrer que si A et B sont deux matrices carrées de taille n vérifiant $A \times B = \mathbb{I}_n$, alors elles sont régulières et $A = B^{-1}$.

Recherche Si A et B sont régulières et de taille n, alors comment calculer $(A \times B)^{-1}$ à partir des inverses de A te B ?

209 Opérations sur les lignes

209a Matrices élémentaires

Nous désignerons par E_n^{ij} la matrice carrée de $\mathbb{A}^{n\times n}$ dont tous les coefficients sont nuls sauf le coefficient (i,j) qui vaut $1_{\mathbb{A}}$. Par exemple, $E_3^{12} = \left(\begin{smallmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$ dans $\mathbb{Z}^{3\times 3}$.

Leoplod KRONECKER (1823-1891)

Leopold KRONECKER est un mathématicien prussien né dans l'actuelle Pologne. On lui doit la célèbre citation : « *Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk* ».
En son honneur, on a donné son nom à la fonction suivante :

$$\delta : \begin{array}{ccc} \mathbb{N}\times\mathbb{N} & \to & \{0;1\} \\ (i,j) & \mapsto & 1 \text{ si } i=j,\ 0 \text{ sinon} \end{array}$$

On condense souvent la notation en δ_{ij} et on parle alors de **symbole de Kronecker**. Par exemple, la matrice identité peut être définie par :

$$\mathbb{I}_n = (\delta_{ij})_{\substack{1\leqslant i\leqslant n \\ 1\leqslant j\leqslant n}}$$

Recherche

Étudiez le produit $E_n^{ij} \times E_n^{\ell k}$ et exprimez-le à l'aide du symbole de KRONECKER.
Simplifiez ensuite le produit $(\mathbb{I}_n + \lambda E_n^{ij}) \times (\mathbb{I}_n - \lambda E_n^{ij})$: qu'en concluez-vous ?

2 0 9 b Transvections de lignes

On s'intéresse à la fonction :

$$T_\lambda^{ij} : \begin{array}{ccc} \mathbb{A}^{n\times p} & \to & \mathbb{A}^{n\times p} \\ M & \mapsto & (\mathbb{I}_n + \lambda E_n^{ij}) \times M \end{array}$$

Calculez par exemple l'image par T_λ^{23} de $\begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{pmatrix}$
Ainsi, $T_\lambda^{ij}(M)$ permet d'obtenir la matrice construite à partir de M en remplaçant la ligne i par elle-même plus λ fois la ligne j.
On note plus commodément cette transformation $L_i \leftarrow L_i \boxplus (\lambda \boxdot L_j)$.
Vous aurez bien noté que $(\mathbb{I}_n + \lambda E_n^{ij})^{-1} = \mathbb{I}_n - \lambda E_n^{ij}$.

2 0 9 c Dilatations de lignes

On veut effectuer l'opération $L_i \leftarrow \lambda \boxdot L_i$. On note

$$\Delta_n^{i,\lambda} = \mathbb{I}_n + (\lambda \boxplus (-1_{\mathbb{A}}))E_n^{ii}$$

Calculez par exemple le produit de $\Delta_3^{2,\lambda}$ par $\begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{pmatrix}$
Le produit par $\Delta_n^{i,\lambda}$ est une dilatation de ligne.

2 0 9 d Échange de lignes

On considère la matrice $S_n^{ij} = \Delta_n^{j,-1_{\mathbb{A}}} \times (\mathbb{I}_n + E_n^{ij}) \times (\mathbb{I}_n - E_n^{ji}) \times (\mathbb{I}_n + E_n^{ij})$.
Développez ce produit. Que vaut le produit de S_3^{23} par $\begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{pmatrix}$?
On note cette opération $L_i \leftrightarrow L_j$.

2 0 9 e Opérations sur les lignes

De manière plus générale, une fonction φ de $\mathbb{A}^{n\times p}$ dans lui-même est une **opération sur les lignes** si c'est la composée finie de transvections et de dilatations de lignes.
L'image d'une matrice par φ est donc le produit à gauche par des matrices de transvections ou de dilatations :

$$\varphi : M \mapsto (F_k \times F_{k-1} \cdots \times F_1) \times M$$

avec chaque F_i inversible. La fonction φ est donc elle-même totale bijective et sa réciproque est :

$$\varphi^{-1} : N \mapsto (F_1^{-1} \times F_2^{-1} \cdots \times F_k^{-1}) \times N$$

On en déduit en particulier que l'inverse de $\varphi(\mathbb{I}_n)$ existe et que c'est $\varphi^{-1}(\mathbb{I}_n)$.

2.0.9.f Lien avec les matrices régulières

Voici un théorème important qui nous sera très utile pour calculer l'inverse d'une matrice régulière.

Théorème 2 - 1

φ étant une opération élémentaire sur les lignes,

$$M \text{ inversible} \leftrightarrow \varphi(M) \text{ inversible}$$

En effet nous savons que $\varphi(M) = \varphi(\mathbb{I}_n) \times M$. Si M est inversible, $\varphi(M)$ s'exprime comme le produit de deux matrices inversibles et elle est donc inversible. Supposons maintenant $\varphi(M)$ inversible, comme $\varphi(\mathbb{I}_n)$ est inversible on obtient :

$$\varphi(\mathbb{I}_n)^{-1} \times \varphi(M) = \varphi(\mathbb{I}_n)^{-1} \times (\varphi(\mathbb{I}_n) \times M)$$

qui se transforme en

$$\varphi(\mathbb{I}_n)^{-1} \times \varphi(M) = \left(\varphi(\mathbb{I}_n)^{-1} \times \varphi(\mathbb{I}_n)\right) \times M = \mathbb{I}_n \times M$$

et en définitive

$$M = \varphi(\mathbb{I}_n)^{-1} \times \varphi(M) = \varphi^{-1}(\mathbb{I}_n) \times \varphi(M)$$

M s'exprimant comme le produit de deux matrices inversibles est inversible.

Théorème 2 - 2

Si $\varphi_1, \varphi_2, \ldots, \varphi_k$ est une suite d'opérations sur les lignes de M qui transforme M en $\mathbb{I}_n$ alors M est inversible et

$$M^{-1} = \varphi_k(\mathbb{I}_n) \times \varphi_{k-1}(\mathbb{I}_n) \times \ldots \times \varphi_1(\mathbb{I}_n) = \varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(\mathbb{I}_n)$$

En effet, si nous avons $\varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(M) = \mathbb{I}_n$ alors

$$\varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(M) = \mathbb{I}_n = (\varphi_k(\mathbb{I}_n) \times \varphi_{k-1}(\mathbb{I}_n) \times \ldots \times \varphi_1(\mathbb{I}_n)) \times M$$

Nous avons trouvé une matrice carrée qui multipliant M donne $\mathbb{I}_n$, c'est son inverse.
Le théorème précédent nous indique une méthode pour trouver l'inverse de M (s'il existe), nous allons étudier plus en détail cette technique et nous démontrerons plus loin que s'il est impossible de transformer M en $\mathbb{I}_n$ en utilisant les opérations élémentaires sur les lignes, alors M n'est pas inversible.

2.1 Rang d'une matrice

Dans tout ce qui suit nous allons utiliser les opérations élémentaires sur les lignes d'une matrice et on travaille dans $\mathbb{A}^{n\times p}$.

2.1.1 Matrices ligne-équivalentes

Nous dirons que deux matrices M et N sont **ligne-équivalentes** si, et seulement si, il existe une suite finie $(\varphi_i)_{i\in\mathbb{N}_k}$ d'opérations élémentaires sur les lignes de sorte que

$$N = \varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(M)$$

Nous écrirons alors $M \overset{\ell}{\equiv} N$. La relation $\overset{\ell}{\equiv}$ est manifestement une relation d'équivalence sur $\mathbb{A}^{n\times p}$, on démontre sans peine que

$$\begin{array}{l} M \overset{\ell}{\equiv} M \\ M \overset{\ell}{\equiv} N \rightarrow N \overset{\ell}{\equiv} M \\ \left(M \overset{\ell}{\equiv} N \text{ et } N \overset{\ell}{\equiv} C\right) \rightarrow M \overset{\ell}{\equiv} C \end{array}$$

Nous savons que $\varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(M) = \varphi_k \circ \varphi_{k-1} \circ \ldots \circ \varphi_1(\mathbb{I}_n) \times M$, par conséquent nous aurons $M \overset{\ell}{\equiv} N$ s'il existe une matrice R inversible vérifiant

$$N = R \times M$$

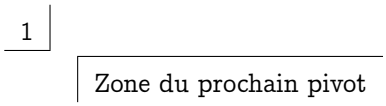
R étant le résultat du produit de matrices du type $\varphi(\mathbb{I}_n)$. Pour obtenir cette matrice R il suffit d'appliquer successivement en parallèle les opérations élémentaires qui transforment M en N sur $\mathbb{I}_n$. Pour cela on utilise un tableau où l'on juxtapose M et $\mathbb{I}_n$, M étant la partie gauche et $\mathbb{I}_n$ la partie droite de ce tableau. Toute opération élémentaire sur les lignes effectuée sur la partie gauche est simultanément effectuée sur la partie droite.

opérations φ	Partie gauche	Partie droite	Remarques
	M	$\mathbb{I}_n$	initialisation du tableau
φ_1	M_1	R_1	$M_1 = \varphi_1(M),\ R_1 = \varphi_1(\mathbb{I}_n)$
φ_2	M_2	R_2	$M_2 = \varphi_2(M_1),\ R_2 = \varphi_2(R_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
φ_i	M_i	R_i	$M_i = R_i \times M$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
φ_k	N	R	$N = R \times M$

2.1.2 L réduite échelonnée

La matrice $M = (m_{ij}) \in \mathbb{A}^{n\times p}(\mathbb{K})$ est dite **ℓ-réduite** (ℓ pour « ligne ») si, et seulement si, elle satisfait aux conditions suivantes :

1. Toutes les lignes nulles (une ligne est nulle si elle ne comporte que des zéros) sont au-dessous des lignes non nulles.
2. Dans chaque ligne non nulle le premier élément non nul est $1_{\mathbb{A}}$ (on lit une ligne de la gauche vers la droite), ce $1_{\mathbb{A}}$ est appelé **pivot** ou élément pivot. La colonne où se trouve ce $1_{\mathbb{A}}$ est appelée colonne pivot et c'est le seul élément non nul de cette colonne.
3. Si, **de plus**, les pivots apparaissent en ordre croissant par numéro de ligne **et** numéro de colonne, on dit que M est **ℓ-réduite échelonnée** (en abrégé ℓré ou LRé).



Donnons quelques exemples de matrices entières :

$$\begin{pmatrix} 0 & \boxed{1} & 2 & 0 \\ \boxed{1} & 0 & 3 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$ est ℓ-réduite non échelonnée

$$\begin{pmatrix} 0 & \boxed{1} & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \boxed{1} \end{pmatrix}$$ n'est pas ℓ-réduite

$$\begin{pmatrix} \boxed{1} & -2 & 0 \\ 0 & 0 & \boxed{1} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$ est ℓ-réduite échelonnée

Théorème 2 - 3

Pour toute matrice $M \in \mathbb{A}^{n\times p}$, il existe une unique matrice ℓ-réduite échelonnée $N \in \mathbb{A}^{n\times p}$ telle que $M \overset{\ell}{\equiv} N$, N est appelée la ℓ-réduite échelonnée de M que l'on peut noter par $N = \ell\text{ré}(M)$.

La démonstration de ce théorème se fait par récurrence sur n et elle est un peu difficile.

Le **rang** de M, noté $\text{rang}(M)$, est le nombre de lignes non nulles de sa ℓ-réduite échelonnée, c'est donc aussi le nombre de pivots da sa ℓré. M est dite de plein rang si son rang est égal à son nombre de lignes.
Nous énoncons les évidences (conséquences directes de la définition) :

1. Si $M \in \mathbb{A}^{n\times p}$ le rang de M est inférieur ou égal à n **et** à p.
2. Si $M \in \mathbb{A}^{n\times n}$ et si $\text{rang}(M) = n$ alors sa ℓ-réduite échelonnée est $\mathbb{I}_n$ et nous avons précédemment démontré que dans ce cas M est inversible.
3. Deux matrices ligne-équivalentes ont la même ℓ-réduite échelonnée et ont donc même rang.
4. Si M' est une matrice extraite de M, on a $\text{rang}(M') \leq \text{rang}(M)$.

2 2 Algorithme Fang-Tcheng

Wilhelm JORDAN
(1842-1899)

Notre eurocentrisme préfère nommer cet algorithme GAUSS-JORDAN...
Nous allons le présenter sur un exemple. Soit à chercher la ℓ-réduite échelonnée de la matrice

$$A = \begin{pmatrix} 2 & 5 & -2 & 3 \\ 3 & 6 & 3 & 6 \\ 1 & 2 & -1 & 2 \end{pmatrix}$$

Nous allons faire apparaître les pivots ordonnés par numéro de ligne et numéro de colonne, cela veut dire que si un pivot (qui sera toujours égal à 1) est obtenu à la ligne i et colonne j, le pivot suivant sera au moins en ligne $i+1$ et en colonne $j+1$ et on s'imposera de trouver la solution minimale en numéro de ligne et numéro de colonne.

— **Première étape.** On repère l'élément de la première colonne qui est le plus grand en valeur absolue (il peut y avoir plusieurs choix). Si le résultat est nul (la première colonne ne contient que des zéros), la première colonne ne peut être une colonne pivot et on passe à la colonne suivante. Ici c'est 3 qui se trouve sur la deuxième ligne première colonne. On permute alors la première ligne avec la deuxième ligne et on obtient

$$A_1 = \begin{pmatrix} 3 & 6 & 3 & 6 \\ 2 & 5 & -2 & 3 \\ 1 & 2 & -1 & 2 \end{pmatrix}$$

On divise tous les éléments de la première ligne par 3 :

$$A_2 = \begin{pmatrix} \boxed{1} & 2 & 1 & 2 \\ 2 & 5 & -2 & 3 \\ 1 & 2 & -1 & 2 \end{pmatrix}$$

On fait apparaître ensuite des zéros sous le premier 1 de la première colonne en utilisant les opérations $L_2 \leftarrow L_2 - 2L_1$ et $L_3 \leftarrow L_3 - 1L_1$:

$$A_3 = \begin{pmatrix} \boxed{1} & 2 & 1 & 2 \\ 0 & 1 & -4 & -1 \\ 0 & 0 & -2 & 0 \end{pmatrix}$$

La première colonne est une colonne pivot et elle ne devra pas être modifiée par la suite.

— **Deuxième étape.** On repère dans la deuxième colonne (en fait la colonne qui suit la dernière colonne pivot obtenue), à partir de la deuxième ligne (en fait à partir du numéro de ligne qui suit le numéro de la ligne qui a donné le dernier pivot), le plus grand élément en valeur absolue (si on obtient zéro on passe à la colonne suivante, etc···). Ici on obtient 1 qui se trouve sur la deuxième ligne et deuxième colonne et il n'y a pas de permutation de lignes à faire. Maintenant il faut faire apparaître des zéros dans la colonne pivot en ligne 1 et en

ligne 3. On utilise les opérations $L_1 \leftarrow L_1 - 2L_2$ et $L_3 \leftarrow L_3 - 0L_2$ (qui ne sert à rien) ; on obtient :

$$A_4 = \begin{pmatrix} \boxed{1} & 0 & 9 & 4 \\ 0 & \boxed{1} & -4 & -1 \\ 0 & 0 & -2 & 0 \end{pmatrix}$$

La deuxième colonne est une colonne pivot, elle ne devra pas être modifiée dans la suite.

— **Troisième étape.** On repère dans la colonne qui suit la dernière colonne pivot obtenue et à partir de la ligne qui suit la ligne qui a donné le dernier pivot le plus grand élément en valeur asolue. Ici c'est -2 qui se trouve sur la troisième ligne et troisième colonne, la troisième colonne est alors la troisième colonne pivot. Il n'y a pas de permutation de lignes à faire, nous n'avons qu'à faire apparaître un 1 à la place de -2 puis faire apparaître des zéros dans la colonne pivot en conservant évidemment le pivot 1. Appliquons à A_4 l'opération $L_3 \leftarrow -\frac{1}{2}L_3$

$$A_5 = \begin{pmatrix} \boxed{1} & 0 & 9 & 4 \\ 0 & \boxed{1} & -4 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

puis appliquons $L_1 \leftarrow L_1 - 9L_3$ et $L_2 \leftarrow L_2 + 4L_3$

$$A_6 = \begin{pmatrix} \boxed{1} & 0 & 0 & 4 \\ 0 & \boxed{1} & 0 & -1 \\ 0 & 0 & \boxed{1} & 0 \end{pmatrix}$$

Nous venons d'obtenir la ℓ-réduite échelonnée de A, A est de rang 3.

Nous avons utilisé, ici, la méthode dite du pivot partiel en cherchant dans chaque colonne le plus grand élément en valeur absolue. Si nous avions choisi de prendre le premier élément rencontré non nul, en vertu de l'unicité de la ℓ-réduite échelonnée, nous aurions obtenu le même résultat final. Il est conseillé, en programmation, d'utiliser la méthode dite pivot partiel pour éviter une trop grande propagation des erreurs lors des divisions par des petits nombres. Si on fait les calculs à la main il vaut mieux se contenter de choisir, tant que c'est possible, des nombres sympathiques.

Théorème 2 - 4

A est une matrice carrée d'ordre n inversible si, et seulement si, le rang de A est égal à n.

Nous avons en fait déjà démontré une partie de ce théorème mais, vu son importance, nous allons reprendre ce qui a été dit. Supposons donc que le rang de A est égal à n, dans ces conditions la ℓ-réduite échelonnée de A est $\mathbb{I}_n$, cela signifie que $A \overset{\ell}{\equiv} \mathbb{I}_n$ et donc qu'il existe A' vérifiant $A' \times A = \mathbb{I}_n$, A' est l'inverse de A. Supposons maintenant que la rang de A est strictement inférieur à n, il est alors sûr que la dernière ligne de la ℓ-réduite échelonnée de A est une ligne nulle. Notons B cette ℓ-réduite échelonnée, nous savons que A inversible équivaut à écrire que B est inversible puisque $A \overset{\ell}{\equiv} B$. Supposons B inversible, il existe alors B' vérifiant $B \times B' = \mathbb{I}_n$ or cette égalité est impossible car la dernière ligne de B étant nulle, la dernière ligne du produit $B \times B'$ sera aussi nulle et donc distincte de la dernière ligne de $\mathbb{I}_n$. Nous venons de démontrer

$$\text{rg}(A) < n \rightarrow A \text{ non inversible}$$

et donc A inversible $\rightarrow \text{rg}(A) = n$.

Pratique : pour rechercher la matrice inverse de A, si elle existe, on applique simultanément sur $\mathbb{I}_n$ les opérations faites pour déterminer la ℓ-réduite échelonnée de A. Pour cela on considère le tableau

$$[A|\mathbb{I}_n]$$

on applique l'algorithme Fang Tcheng tableau, et chaque opération faite sur la partie gauche est reproduite sur la partie droite. Si la ℓ-réduite de A est la matrice $\mathbb{I}_n$ (le résultat de la partie gauche est $\mathbb{I}_n$) alors A est inversible et sa matrice inverse est donnée par la partie droite. Si la partie gauche ne donne pas $\mathbb{I}_n$, A n'est pas inversible, son rang est strictement inférieur à n.

2 3 Résolution de systèmes

2 3 1 Généralités

Gabriel CRAMER
(1704-1752)

On appelle **système de n équations linéaires à p inconnues** dans $\mathbb{A}$ tout système de la forme :

$$(S):\begin{cases}\sum_{j=1}^{p} a_{i,j}x_j = b_i \\ i \in \{1,2,\cdots,n\} \\ a_{i,j} \text{ et } b_i \in \mathbb{K}\end{cases}$$

Soit aussi

$$(S):\begin{cases}a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,p}x_p = b_1 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,p}x_p = b_p\end{cases}$$

$x_1, x_2, \cdots, x_p$ sont les p inconnues, les $a_{i,j}$ sont appelés les coefficients du système (S) et les b_i sont appelés les seconds membres (ce sont aussi des coefficients du système (S)).
Notons $A = (a_{i,j}) \in \mathbb{A}^{n\times p}$ (A est appelée la matrice du système).

$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$ est la matrice colonne des inconnues et $B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$ est la matrice colonne des seconds membres, le système (S) s'écrit matriciellement :

$$\begin{pmatrix} a_{1,1} & a_{1,2} & . & . & . & a_{1,p} \\ a_{2,1} & a_{2,2} & . & . & . & a_{2,p} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ a_{n,1} & a_{n,2} & . & . & . & a_{n,p} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

soit aussi

$$A \times X = B \text{ ou } {}^tX \times {}^tA = {}^tB$$

Résoudre le système (S) c'est chercher tous les p-uplets $(x_1, x_2, \cdots, x_p)$ de $\mathbb{A}^p$ qui vérifient simultanément les n équations. C'est aussi chercher toutes les matrices $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} \in \mathbb{A}^{p\times 1}$ vérifiant l'égalité matricielle $A \times X = B$. C'est pourquoi, dans ce qui suit, nous serons souvent amenés à confondre le p-uplet $(x_1, x_2, \cdots, x_p)$ avec la matrice colonne $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$. Attention, en informatique et plus particulièrement en réseaux, le p-uplet $(x, x_2, \cdots, x_p)$ est aussi noté matriciellement par

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_p \end{pmatrix}$$

qui n'est autre que la transposée de X.
Le système est dit **homogène** si, et seulement si, tous les seconds membres (les coefficients b_i) sont nuls. Dans ce cas le système admet forcément au moins une solution : la solution nulle ou solution banale qui est le p-uplet

$$(0, 0, \cdots, 0) \in \mathbb{A}^p$$

ou bien la matrice nulle $0_{p,1}$ si on note matriciellement cette solution.

Le système $A \times X = B$ est dit **régulier** ou de **Cramer** si, et seulement si, il a autant d'équations que d'inconnues et si A est inversible (le système ayant autant d'équations que d'inconnues on a $n = p$ et A est une matrice carrée). A étant inversible, A^{-1} existe et

$$\begin{aligned} A \times X &= B \text{ se transforme en} \\ A^{-1} \times (A \times X) &= A^{-1} \times B, \text{ c'est-à-dire} \\ X &= A^{-1} \times B \end{aligned}$$

Le système admet donc au moins une solution qui est $A^{-1} \times B$. Nous allons prouver qu'il n'y en a pas d'autre. Pour cela supposons l'existence d'une autre solution X', nous avons alors simultanément

$$A \times X = B \text{ et } A \times X' = B$$

En retranchant membre à membre on obtient :

$$A \times (X - X') = 0_{n,1}$$

Multiplions maintenant les deux membres par A^{-1}

$$\begin{aligned} A^{-1} \times (A \times (X - X')) &= A^{-1} \times 0_{n,1} = 0_{n,1} \\ (A^{-1} \times A) \times (X - X') &= 0_{n,1} \\ \mathbb{I}_n \times (X - X') &= 0_{n,1} = X - X' \text{ et pour finir} \\ X &= X' \end{aligned}$$

Nous retiendrons que si le système $A \times X = B$ est tel que A est inversible (il est nécessaire que A soit carrée et donc que le système possède autant d'équations que d'inconnues) alors il admet une unique solution qui est

$$X = A^{-1} \times B$$

2.3.2 Systèmes équivalents et résolution

Deux systèmes linéaires (S) et (S') sont **équivalents** si, et seulement si, ils possèdent le même ensemble solution.
Nous allons définir trois opérations élémentaires :

1. Multiplication d'une équation par un scalaire (un élément de $\mathbb{A}$) non nul. Si E_i est la i^{e} équation, le résultat de la multiplication de E_i par $\alpha \in \mathbb{A}$ est noté αE_i et nous écrirons $E_i \leftarrow \alpha E_i$.

$$\begin{aligned} E_i &: \sum_{j=1}^{p} a_{i,j} x_j = b_i \\ \alpha E_i &: \sum_{j=1}^{p} \alpha a_{i,j} x_j = \alpha b_i \end{aligned}$$

   Pour retrouver l'équation initiale il suffit de multiplier l'équation obtenue par $\frac{1}{\alpha}$ et par conséquent le système obtenu par cette opération est équivalent au système initial.

2. Permutation de deux équations, nous noterons $E_i \longleftrightarrow E_j$ l'opération qui consiste à permuter l'équation numéro i avec l'équation numéro j. Pour retrouver le système initial il suffit d'appliquer de nouveau cette opération, cela nous permet d'affirmer que cette opération transforme un système en un système équivalent.

3. Ajout à une équation une autre équation multipliée par un scalaire : si on ajoute à l'équation E_i l'équation $E_{h \neq i}$ multipliée par β nous noterons $E_i \leftarrow E_i + \beta E_h$

$$\begin{cases} E_i : \sum_{j=1}^{p} a_{i,j} x_j = b_i \\ E_h : \sum_{j=1}^{p} a_{h,j} x_j = b_h \\ E_i \leftarrow E_i + \beta E_h : \sum_{j=1}^{p} (a_{i,j} + \beta a_{h,j}) x_j = b_i + \beta b_h \end{cases}$$

   Pour retrouver l'équation de départ il suffit d'appliquer au résultat l'opération $E_i \leftarrow E_i - \beta E_h$ et cette opération, comme les précédentes, transforme un système en un système équivalent.

Nous allons maintenant passer à la résolution : nous pouvons écrire le système

$$(S):\begin{cases}\sum_{j=1}^{p} a_{i,j}x_j = b_i \\ i \in \{1,2,\cdots,n\} \\ a_{i,j} \text{ et } b_i \in \mathbb{A}\end{cases} \quad \text{sous la forme } [A \mid B]$$

où A est la matrice du système et B la matrice colonne des seconds membres. Nous notons T cette matrice (ou ce tableau), le nombre de lignes de T est égal au nombre de lignes de A soit aussi le nombre d'équations de (S), le nombre de colonnes de T est égal au nombre de colonnes de $A+1$ soit aussi le nombre d'inconnues +1. Les opérations élémentaires sur les équations sont alors des opérations élémentaires sur les lignes de ce tableau, les opérations se faisant simultanément sur la partie gauche et sur la partie droite, c'est-à-dire sur le tableau T. Appliquons à ce tableau l'algorithme de Gauss-Jordan pour obtenir la ℓ-réduite échelonnée de A (on ne cherchera pas pour le moment à faire apparaître un pivot dans la dernière colonne du tableau) et notons R cette ℓ-réduite échelonnée. Nous notons le résultat

$$T' = [R \mid H], R = \ell\text{ré}(A)$$

Le système est alors devenu $R \times X = H$. Comme on a $A \overset{\ell}{\equiv} R$ et $B \overset{\ell}{\equiv} H$, il existe P (rappel, P est inversible) telle que

$$R = P \times A \qquad H = P \times B$$

Si nous notons U une solution du système (S), U vérifie

$$A \times U = B$$

et en multipliant les deux membres à gauhe par P on obtient

$$\begin{aligned} P \times A \times U &= P \times B \text{ soit} \\ R \times U &= H \end{aligned}$$

De même toute solution de $R \times X = H$ est solution de $A \times X = B$, en effet notons de même U une solution de $R \times X = H$, on a

$$\begin{aligned} R \times U &= H \text{ donc} \\ P \times A \times U &= P \times B \end{aligned}$$

et en multipliant les deux membres à gauche par P^{-1} on obtient

$$A \times U = B$$

On a bien prouvé que les deux systèmes $A \times X = B$ et $R \times X = H$ sont équivalents. Il nous reste à résoudre le système $R \times X = H$.
Continuons, si besoin, le calcul de la ℓ-réduite échelonnée de T et notons T'' le résultat :

$$T' = [R \mid H], T'' = [R \mid H']$$

Si on arrive à faire apparaître un pivot dans la dernière colonne, les opérations sur les lignes ne modifieront pas la partie gauche car la partie gauche de cette ligne pivot est constituée de zéros.

- Si le système est homogène (tous les seconds membres sont nuls) on a forcément $T' = T''$ et le système admet, rappelons-le, au moins la solution nulle.
- Si le rang de T est supérieur au rang de R (c'est donc que le rang de T est d'une unité supérieure au rang de A, on ne peut créer qu'un seul nouveau pivot au maximum) cela signifie qu'il y a plus de lignes nulles dans R que dans T'' et par conséquent que le système $R \times X = H'$ contient l'équation du type

  $$0x_1 + 0x_2 + \cdots + 0x_p = 1$$

  ou que le système $R \times X = H$ contient au moins une équation du type

  $$0x_1 + 0x_2 + \cdots + 0x_p = h \text{ avec } h \neq 0$$

  ce qui est impossible et le système n'admet pas de solution.

— Si $\text{rang}(T) = \text{rang}(R) = r$, c'est que $T' = T''$ et la résolution de $A \times X = B$ équivaut à la résolution des r équations non nulles de $R \times X = H$. Deux cas peuvent se présenter :

1. $r = p$, le système à résoudre est alors de la forme :

$$\begin{cases} x_1 & & & = h_1 \\ & x_2 & & = h_2 \\ & & \vdots & \vdots \\ & & x_p & = h_p \end{cases}$$

et il est résolu, il y a unicité de la solution.

2. $r < p$. Nous appelons inconnues pivots ou inconnues principales (IP) les r inconnues $x_{j_1}, x_{j_2}, \cdots, x_{j_r}$ correspondant aux r colonnes pivots $j_1, j_2, \cdots, j_r$ de R. Les $p-r$ autres inconnues sont appelées inconnues non principales (INP), certains les appellent aussi des inconnues ou des variables libres ou encore des paramètres. Pour résoudre le système la méthode consiste à faire passer dans les seconds membres les inconnues non principales, le système devient alors un sytème de Cramer résolu dont les solutions dépendent des $p-r$ inconnues non principales (ces inconnues non principales sont, à ce stade, considérées comme des paramètres), **il y a donc une infinité de solutions** ; si l'on désire une solution particulière, il suffit de donner des valeurs arbitraires aux inconnues non principales.

$$\begin{cases} x_{j_1} & & & = h_{j_1} + e_{j_1} \\ & x_{j_2} & & = h_{j_2} + e_{j_2} \\ & & \vdots & \vdots \\ & & x_{j_r} & = h_{j_r} + e_{j_r} \end{cases}$$

où les e_{j_i} sont des expressions linéaires des $(p-r)$ INP.

Il faut remarquer que si l'on modifie l'ordre d'écriture des inconnues nous n'obtiendrons pas forcément les mêmes inconnues principales et non principales mais l'ensemble solution sera évidemment le même.

3 Vade-mecum

— Vous devez savoir ce qu'est un groupe, E a une structure de groupe ssi on a défini dans E une loi de composition interne = une opération (le plus souvent cette opération est notée $+$) qui a « de bonnes propriétés » (voir le cours).

— H est un sous groupe de E ssi $H \subseteq E$ et $H \neq \varnothing$ et si H est stable pour l'opération de groupe $+$. Qu'est-ce que cela signifie ? Tout simplement que si on fait des additions ou des soustractions dans H, on « reste » ou on « ne sort pas » de H, il suffit de retenir ce résultat ! Donc, par exemple, si $x \notin H$ on a forcément $x - x = 0_E \in H$.

— E est un espace vectoriel sur $\mathbb{R}$ (par exemple mais ce peut être un autre corps, pour les codes détecteurs et correcteurs d'erreurs on utilse le corps $\mathbb{F}_2$) ssi on a défini 2 opérations dans E : une addition « interne » qui fait que E est un groupe commutatif et une multiplication par un scalaire = un nombre (ici un nombre réel) qui a de « bonnes » propriétés.

— Si E est un e.v. sur $\mathbb{R}$, que signifie « H est un sous espace vectoriel ? Tout simplement que $H \subseteq E$ et $H \neq \varnothing$ et que H est stable pour les opérations de l'espace vectoriel. Si on fait des calculs dans H, on reste dans H, il suffit de retenir ce résultat ! Par exemple, si H est un sev de E, et si $u \in H$ on doit avoir $u - u = 0_E \in H$ ou encore $0.u = 0_E \in H$.

— Si E est un e.v. sur $\mathbb{R}$, il est d'usage d'appeler les éléments de E des vecteurs, 0_E = « le zéro de E » i.e. le vecteur nul.

— Faire des calculs dans un e.v. c'est « faire des combinaisons linéaires » = CL.

— Une CL de vecteurs c'est un calcul où interviennent ces vecteurs.

— Une famille de vecteurs = une suite de vecteurs = une liste ordonnée de vecteurs

— Si $\mathcal{F} = (u, v, w)$ est une famille de vecteurs de E,

$$\begin{aligned} 2u - 3v + 0w &= 2u - 3v \\ 0u + 0v + 1w &= w \end{aligned}$$

sont des CL des vecteurs de $\mathcal{F}$.

— On note $\boldsymbol{Vect}(\mathcal{F})$ l'ensemble des CL des vecteurs de $\mathcal{F}$. Donc si $z \in \boldsymbol{Vect}(u, v, w)$, cela signifie que z peut s'écrire comme une CL des vecteurs de $\mathcal{F}$.

— $\boldsymbol{Vect}(\mathcal{F})$ est un sev (évident).

— Dire que $\mathcal{F}$ est génératice de E, c'est dire que $E = \boldsymbol{Vect}(\mathcal{F})$, c'est donc dire que tout vecteur de E peut s'écrire comme une CL des vecteurs de $\mathcal{F}$.

— Considérons la famille de k vecteurs $\mathcal{F} = (u_1, u_2, ..., u_k)$ et considérons l'équation

$$\alpha_1 u_1 + \alpha_2 u_2 + ... + \alpha_k u_k = 0 = 0_E$$

où les α_i sont les inconnues réelles. Il est clair que cette équation a pour solution

$$\alpha_1 = \alpha_2 = ... = \alpha_k = 0 = 0_{\mathbb{R}}$$

  — Si c'est la seule solution on dit que la famille $\mathcal{F}$ est libre ou que les vecteurs de $\mathcal{F}$ sont linéairement indépendants.
  — Si ce n'est pas la seule solution, on dit que la famille $\mathcal{F}$ est liée ou que les vecteurs de $\mathcal{F}$ sont linéairement dépendants.
  — Donc une famille qui n'est pas libre est liée et une famille qui n'est pas liée est libre, « libre » est le contraire logique de « liée ».

— $\mathcal{F} = (u)$ est libre ssi $u \neq 0_E$ (u est différent du vecteur nul)

— Si $\mathcal{F} = (u, v)$ est liée on dit que u et v sont colinéaires : l'un peut s'écrire comme une CL de l'autre.

— Une famille qui comporte le vecteur nul est forcément liée.

— $\mathcal{F}$ est liée ssi l'un des vecteurs de $\mathcal{F}$ peut s'écrire comme une CL des autres.

— Une base de E est une famille libre et génératrice, on démontre que toutes les bases d'un espace vectoriel ont le même nombre d'éléments, ce nombre s'appelle la dimension de l'e.v.

— Soit $\mathcal{B} = (e_1, e_2, ..., e_n)$ une base de E et u un vecteur quelconque de E ; $\mathcal{B}$ étant génératrice, u peut s'écrire comme une CL des vecteurs de $\mathcal{B}$ et comme $\mathcal{B}$ est libre on démontre que cette écriture est unique : il existe donc n réels $x_1, ..., x_n$ vérifiant

$$u = x_1 e_1 + ... + x_n e_n$$

ces réels sont uniques, on dit que l'on a décomposé u dans la base $\mathcal{B}$ et ces réels sont les coordonnées de u dans la base $\mathcal{B}$, notation :

$$u \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{\mathcal{B}}$$

pour exprimer que $u = \sum_{i=1}^{n} x_i e_i$ et on dit que $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ est la matrice des coordonnées de x dans la base $\mathcal{B}$.

— $\mathbb{R}^n$ est un e.v. sur $\mathbb{R}$ avec les opérations « addition de 2 n-uplets et multiplication d'un n-uplet par un réel ».

— Soit $x = (x_1, x_2, \ldots, x_n)$ un vecteur quelconque de $\mathbb{R}^n$, x peut s'écrire

$$x = x_1 (1, 0, \ldots, 0) + x_2 (0, 1, 0, \cdots, 0) + \cdots + x_n (0, 0, \cdots, 0, 1)$$

Nous notons e_i le n-uplet de $\mathbb{R}^n$ qui a tous ses éléments nuls sauf le $i^{ème}$ qui est égal à 1 :

$$\begin{aligned} e_1 &= (1, 0, 0, \cdots, 0, 0) \\ e_2 &= (0, 1, 0, \cdots, 0, 0) \\ &\vdots \\ e_n &= (0, 0, 0, \cdots, 0, 1) \end{aligned}$$

la famille $\mathcal{B} = (e_1, e_2, ..., e_n)$ est une base de $\mathbb{R}^n$, la dimension de $\mathbb{R}^n$ est donc égale à n.

— $\{0_E\}$ est un sev de E, sa dimension est égale à 0 (convention).

— Dans $\mathbb{R}^n$ toute famille libre de n vecteurs est génératrice et est donc une base.

— Dans $\mathbb{R}^n$ toute famille génératrice de n vecteurs est libre et est donc une base.

— On appelle **matrice d'une famille** dans une base, la matrice des coordonnées des vecteurs de cette famille dans la base considérée ; on juxtapose les coordonnées des vecteurs de la famille dans la base et le résultat est la matrice de la famille dans la base. Considérons la famille de p vecteurs de $\mathbb{K}^n$:

$$\mathcal{F} = (u_1, u_2, \ldots, u_p)$$

Notons les coordonnées de u_j dans la base $\mathcal{B}$ par

$$u_j \begin{pmatrix} a_{1,j} \\ a_{2,j} \\ \vdots \\ a_{n,j} \end{pmatrix}_{\mathcal{B}}$$

et la matice de $\mathcal{F}$ dans la base $\mathcal{B}$ est

$$A = \text{Mat}_{\mathcal{B}}(\mathcal{F}) = (a_{i,j}) \in \mathcal{M}_{n,p}(\mathbb{R})$$

A est aussi qualifiée de **matrice génératrice** de $\text{Vect}(\mathcal{F})$ dans la base $\mathcal{B}$ et les colonnes de A sont appelées des vecteurs colonnes.

— Le rang d'une famille $\mathcal{F}$ est le rang de sa matrice dans une base, on note

$$\text{rang}(\mathcal{F}) = \text{rang}(\text{Mat}_{\mathcal{B}}(\mathcal{F}))$$

— Le rang d'une famille $\mathcal{F}$ est la taille des familles libres maximales que l'on peut extraire de $\mathcal{F}$, ces familles libres maximales étant des bases de $\boldsymbol{Vect}(\mathcal{F})$.

— Si $\mathcal{F}$ est une famille de p vecteurs de $\mathbb{K}^n$ alors

$$\text{Vect}(\mathcal{F}) = \mathbb{K}^n \Leftrightarrow \text{rang}(\mathcal{F}) = n$$

— Voici le problème : $\mathcal{B} = (e_1, e_2, \cdots, e_n)$ et $\mathcal{B}' = (e'_1, e'_2, \cdots, e'_n)$ sont deux bases de $\mathbb{K}^n$, on connaît les coordonnées d'un vecteur x dans $\mathcal{B}$ ainsi que les coordonnées des vecteurs de $\mathcal{B}'$ dans la base $\mathcal{B}$ et on cherche les coordonnées de x dans la base $\mathcal{B}'$. Pour résoudre ce problème nous allons utiliser les notations suivantes :

— $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ est la matrice colonne des coordonnées de x dans la base $\mathcal{B}$.

— $X' = \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix}$ est la matrice colonne des coordonnées de x dans la base $\mathcal{B}'$.

— $P = (p_{i,j}) \in \mathcal{M}_n(\mathbb{K})$ est la matrice de la famille $\mathcal{B}'$ dans la base $\mathcal{B}$, $P = \text{Mat}_{\mathcal{B}}(\mathcal{B}')$. Cela signifie que la $j^{ième}$ colonne de P nous donne les coordonnées de e'_j dans $\mathcal{B}$, P est appelée la **matrice de passage** de $\mathcal{B}$ à $\mathcal{B}'$. Par hypothèse nous avons :

$$\begin{cases} x = \sum_{i=1}^{n} x_i e_i \\ x = \sum_{j=1}^{n} x'_j e'_j \\ e'_j = \sum_{i=1}^{n} p_{ij} e_i \end{cases}$$

on en déduit que

$$x = \sum_{j=1}^{n} x'_j e'_j = \sum_{j=1}^{n} x'_j \left(\sum_{i=1}^{n} p_{ij} e_i \right) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} p_{ij} x'_j \right) e_i$$

or nous savons que la décomposition d'un vecteur dans une base est unique et, par conséquent,

$$\begin{cases} \sum_{j=1}^{n} p_{ij} x'_j = x_i \\ i \in \mathbb{N}_n \end{cases}$$

les coordonnées de x dans $\mathcal{B}'$ sont donc les solutions du système linéaire précédent qui s'écrit matriciellement :

$$P \times X' = X$$

Ce système est forcément un système de Cramer, il a autant d'équations que d'inconnues et il possède un unique n-uplet solution puisque nous sommes sûrs de l'existence et de l'unicité des x'_j. P est donc de rang n et est inversible (on en était sûr, P est de rang n puisque $\mathcal{B}'$ est libre), la solution X' cherchée est alors donnée par

$$X' = P^{-1} \times X$$

Ce résultat est à retenir.

— Si P est la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$ alors P^{-1} est la matrice de passage de $\mathcal{B}'$ à $\mathcal{B}$.

— Soit $\mathcal{F}$ une famille de p vecteurs et $A = \text{Mat}_{\mathcal{B}}(\mathcal{F})$, de ce qui précède on déduit

$$A' = \text{Mat}_{\mathcal{B}'}(\mathcal{F}) = P^{-1} \times A$$

puisque si on note X une colonne de A, c'est-à-dire la matrice des coordonnées d'un vecteur de $\mathcal{F}$ dans $\mathcal{B}$, la colonne correspondante de A' est $P^{-1} \times X$. Comme le rang de A' est égal au rang de A, il est alors prouvé que le rang d'une famille ne dépend pas de la base choisie.

— E et F sont deux espaces vectoriels sur le même corps $\mathbb{R}$ (nous utiliserons les mêmes notations pour les lois de E et les lois de F) et f est une application de E dans F. On dit que f est une **application linéaire** de E dans F si, et seulement si,

$$\begin{cases} \forall (u_1, u_2) \in E \times E, f(u_1 + u_2) = f(u_1) + f(u_2) \\ \forall (\alpha, u) \in \mathbb{K} \times E, f(\alpha u) = \alpha f(u) \end{cases}$$

En français courant cela signifie que l'image de toute CL de vecteurs de E est égale à la CL des images de ces vecteurs.

— L'ensemble des applications linéaires de E dans F est noté $\mathcal{L}(E, F)$, c'est un espace vectoriel sur $\mathbb{K}$ avec les opérations suivantes :

$$(f, g) \in \mathcal{L}(E, F)^2, \begin{cases} f + g : & E \longrightarrow F \\ & x \longmapsto f(x) + g(x) \end{cases}$$

$$(\alpha, f) \in \mathbb{K} \times \mathcal{L}(E, F), \begin{cases} \alpha f : & E \longrightarrow F \\ & x \longmapsto \alpha f(x) \end{cases}$$

— $f(x + 0_E) = f(x) = f(x) + f(0_E)$ et par conséquent $f(0_E) = 0_F$. Une autre démonstration utilise : $f(0x) = f(0_E) = 0f(x) = 0_F$.

— $f(-x) = f((-1)x) = (-1)f(x) = -f(x)$.

— Si $f \in (E, F)$ et $g \in \mathcal{L}(F, G)$ alors $g \circ f \in \mathcal{L}(E, G)$, c'est très simple à démontrer, on prouve que $g \circ f$ vérifie bien les deux propriétés attendues.

— Si f est bijective on dit que f est un **isomorphisme** (en fait un isomorphisme d'espaces vectoriels) de E sur F.

— Si $E = F$ on dit que f est un **endomorphisme** de E. On note $\mathcal{L}(E)$ l'ensemble des endomorphismes de E au lieu de $\mathcal{L}(E, E)$.

— Un endomorphisme bijectif est appelé un **automorphisme**.

— Endomorphismes de $\mathbb{R}^n$:

  — Dans ce qui suit $\mathcal{B} = (e_1, e_2, \cdots, e_n)$ est une base de $E = \mathbb{R}^n$ et f est un endomorphisme de E.

  — L'image par f d'un sev est un sev.

  — $f(E) = \boldsymbol{Im}(f)$ est un sev de E.

  — $\ker(f) = \{x \in E \mid f(x) = 0_E\}$ est un sev de E.

— La matrice de f dans la base $\mathcal{B} = (e_1, e_2, \cdots, e_n)$ n'est autre que la matrice de la famille $(f(e_1), f(e_2), \cdots, f(e_n))$ dans la base $\mathcal{B}$. Si nous notons A cette matrice, A est une matrice carrée d'ordre n et si $A = (a_{i,j})$ nous avons

$$f(e_j)\begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{pmatrix}_{\mathcal{B}}$$

ce qui équivaut à écrire

$$f(e_j) = \sum_{i=1}^{n} a_{ij} e_i$$

et

$$\begin{cases} f : & E \longrightarrow E \\ & x \longmapsto y = f(x) \end{cases}$$
$$A = \text{Mat}(f, \mathcal{B}, \mathcal{B}) = \text{Mat}_{\mathcal{B}}(f)$$
$$Y = A \times X$$

— Le rang de f est le rang de la famille $(f(e_1), f(e_2), \cdots, f(e_n))$, c'est-à-dire la dimension de $\text{Vect}(f(e_1), f(e_2), \cdots, f(e_n)) = \boldsymbol{Im}(f)$, mais c'est aussi le rang de la matrice A.

— Théorème du rang : $\dim(\boldsymbol{Im}(f)) + \dim(\ker(f)) = \dim(E) = n$. En conséquence

  — Si f est injective alors $\ker(f) = \{0_E\}$ et $\dim(\boldsymbol{Im}(f)) = n$ ce qui prouve que f est surjective.

  — Si f est injective $(f(e_1), f(e_2), \cdots, f(e_n))$ est une base de $\boldsymbol{Im}(f)$ et $\boldsymbol{Im}(f) = E$.

  — Si f est surjective alors $\boldsymbol{Im}(f) = E$ et par conséquent $\dim(\ker(f)) = 0$ qui se traduit par $\ker(f) = \{0_E\}$ et f est injective.

  — Si $(f(e_1), f(e_2), \cdots, f(e_n))$ est une base de E, f est surjective.

  — Si A est de rang n, cela équivaut à écrire que A est inversible, et donc f est surjective.

  — Nous venons de démontrer que les affirmations suivantes sont équivalentes

    — f est injective.

    — $\ker(f) = \{0_E\}$.

    — f est surjective.

    — L'image d'une base de E par f est une base de E.

    — f est bijective.

    — A est inversible.

— Si f est bijective, la matrice de f^{-1} dans la base $\mathcal{B}$ est A^{-1}

$$Y = A \times X \Leftrightarrow X = A^{-1} \times Y$$

— Si nous cherchons A' la matrice de f dans la base $\mathcal{B}'$ en connaissant P la matrice de passage de la base $\mathcal{B}$ à la base $\mathcal{B}'$. Nous savons que si X désigne la matrice des coordonnées de x dans $\mathcal{B}$ et Y la matrice colonne des coordonnées de $y = f(x)$ dans $\mathcal{B}$ alors

$$Y = A \times X$$

En notant X' la matrice de x et Y' la matrice de y dans la base $\mathcal{B}'$ nous avons

$$Y' = A' \times X'$$

En utilisant $X = P \times X'$ et $Y = P \times Y'$ nous obtenons l'égalité

$$P \times Y' = A \times P \times X'$$

et comme P est inversible

$$Y' = P^{-1} \times A \times P \times X$$

ce qui nous donne

$$A' = P^{-1} \times A \times P$$

Un moyen mnémotechnique pour utiliser rapidement et sans effort ce résultat est le suivant

$$\begin{array}{ccc} E_{\mathcal{B}} & \xrightarrow[A]{f} & E_{\mathcal{B}} \\ P \downarrow\uparrow P^{-1} & & P^{-1} \uparrow\downarrow P \\ E_{\mathcal{B}'} & \xrightarrow[A']{f} & E_{\mathcal{B}'} \end{array}$$

Ce diagramme se lisant : $A' = P^{-1} \times A \times P$ ou encore $A = P \times A' \times P^{-1}$.

— Valeurs propres et diagonalisation : on se propose de chercher s'il existe des sev U de sorte que la restriction de f à U soit une homothétie (si $u \in U$, $f(u) = \lambda u$).

  — $\lambda \in \mathbb{R}$ est une valeur propre (vap) de f ssi il existe au moins un vecteur non nul vérifiant $f(u) = \lambda u$.

  — si λ est une vap de f alors $E_\lambda = \{u \in E \mid f(u) = \lambda u\}$ est un sev de E, c'est le sous espace propre (sep) associé à la valeur propre λ.

  — L'ensemble des valeurs propres de f est appelé le spectre de f.

  — La dimension d'un sep est forcément ≥ 1.

  — Si f n'est pas injective ce qui équivaut à dire que $\ker(f)$ n'est pas réduit à $\{0_E\}$ alors 0 est valeur propre et $E_{\lambda=0} = \ker(f)$.

  — La juxtaposition des bases des différents sep est une famille libre. Si cette famille est une base, on dit que cette base est une base de vecteurs propres et la matrice de f dans cette base est diagonale, on dit que l'on a diagonalisé f.

4 Back to code

4.1 Inverse d'une matrice par la méthode de Gauss-Jordan

4.1.1 Opérations élémentaires

Pour effectuer une combinaison linéaire des lignes, on va créer une fonction :
`comb_ligne(ki,kj,M,i,j)`
qui renverra la matrice construite à partir de M en remplaçant la ligne L_j par $k_j \times L_j + k_i \times L_i$.

```
1     def comb_lignes(self,ki,kj,i,j):
2         """Li <- ki*Li + kj * Lj"""
3         f = self.F
4         g = lambda r,c : ki*f(i,c) + kj*f(j,c) if r == i else f(r,c)
5         return Mat(self.D,g)
```

Recherche

On crée également une fonction `mult_ligne(k,M,j):` qui renverra la matrice construite à partir de M en remplaçant la ligne L_j par $k \times L_j$.

4.1.2 Calcul de l'inverse étape par étape

On commence par créer une fonction qui renvoie une matrice où se juxtaposent la matrice initiale et la matrice identité :
Créons une matrice :

```
1     >>> M = list2mat([[1,0,1],[0,1,1],[1,1,0]])
```

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

Complétons-la par la matrice identité de même taille :

```
>>> T = M.cat_carre_droite()
>>> T
 1  0  1  1  0  0
 0  1  1  0  1  0
 1  1  0  0  0  1
```

Recherche

Complétez...

```
    def cat_carre_droite(self):
        """
        Colle l'identité à droite pour la méthode de GJ

        """
        ???
```

On effectue $L_3 \leftarrow L_3 - L_1$:

```
>>> T = T.comb_lignes(1,-1,2,0)
>>> T
 1  0  1  1  0  0
 0  1  1  0  1  0
 0  1 -1 -1  0  1
```

puis $L_3 \leftarrow L_3 - L_2$:

```
>>> T = T.comb_lignes(1,-1,2,1)
>>> T
 1  0  1  1  0  0
 0  1  1  0  1  0
 0  0 -2 -1 -1  1
```

puis $L_3 \leftarrow \frac{1}{2}L_3$:

```
>>> T = T.mult_ligne(-0.5,2)
>>> T
 1  0  1  1   0   0
 0  1  1  0   1   0
 0  0  1  0.5 0.5 -0.5
```

On effectue $L_1 \leftarrow L_1 - L_3$:

```
>>> T = T = T.comb_lignes(1,-1,1,2)
>>> T
 1  0  1  1   0   0
 0  1  0 -0.5 0.5 0.5
 0  0  1  0.5 0.5 -0.5
```

et enfin $L_2 \leftarrow L_2 - L_3$:

```
>>> T = T.comb_lignes(1,-1,0,2)
>>> T
   1    0    0  0.5 -0.5  0.5
   0    1    0 -0.5  0.5  0.5
  -0   -0    1  0.5  0.5 -0.5
```

Il ne reste plus qu'à extraire la moitié droite du tableau qui sera l'inverse cherchée à l'aide d'une petite fonction :

Complétez...

Recherche

```
def extrait_carre_droite(self):
    """
    Extrait le carré de droite d'un tableau de GJ

    """
    ???
```

alors :

```
>>> iM = T.extrait_carre_droite()
>>> iM
 0.5 -0.5  0.5
-0.5  0.5  0.5
 0.5  0.5 -0.5
```

On vérifie que c'est bien la matrice inverse de M :

```
>>> iM * M
   1    0    0
   0    1    0
   0    0    1
```

4 1 3 Réduction sous-diagonale d'une matrice

On pourrait calculer l'inverse d'une matrice en généralisant la méthode précédente mais cela s'avèrerait beaucoup trop gourmand en temps.
Nous allons dans un premier temps trigonaliser la matrice en effectuant des opérations élémentaires.
Pour éviter de faire trop de transformations, nous allons petit à petit réduire la taille de la matrice à trigonaliser en ne considérant que le carré inférieur droit situé sous le pivot courant et mémoriser chaque ligne obtenue dans une matrice.
Cela permet également de généraliser l'emploi de la méthode de Gauss à des matrices non carrées pour la résolution de systèmes par exemple. Il faut donc faire attention maintenant à utiliser le minimum entre le nombre de lignes et le nombre de colonnes de la matrice.
On place des « mouchards » pour comprendre l'évolution du code.

```
    def triangle(self):
        """ renvoie la triangulation d'une matrice de haut en bas """
        [r,c] = self.D
        m     = min(r,c)
        S     = self
        T     = zeros(r,c)
        while m > 0:
            NoLigne = 0
            while S[NoLigne, 0] == 0 and (NoLigne < m - 1):
```

```
                    NoLigne += 1
            if S[NoLigne, 0] != 0:
                pivot = S[NoLigne,0]
                for k in range(1,m):
                    if S[k,0] != 0:
                        S = S.comb_lignes(pivot, -S[k,0],k,0)
                        print("pivot = "+str(pivot))
                        print("S dans for :")
                        print(S)
            T = T.remplace_ligned(r - m,S.F)
            print("Évolution de T :")
            print(T)
            S = S.dswap(NoLigne)
            m -= 1
        return T
```

Par exemple :

```
In [1]: M
Out[1]:
 1  2  3
 4  5  6
 7  8  8

In [2]: M.triangle()
pivot = 1
S dans for :
 1  2  3
 0 -3 -6
 7  8  8
pivot = 1
S dans for :
 1  2  3
 0 -3 -6
 0 -6 -13
Évolution de T :
 1  2  3
 0  0  0
 0  0  0
pivot = -3
S dans for :
-3 -6
 0  3
Évolution de T :
 1  2  3
 0 -3 -6
 0  0  0
Évolution de T :
 1  2  3
 0 -3 -6
 0  0  3
```

Nous avons besoin de deux fonctions intermédiaires, une remplissant le tableau T et l'autre réduisant S.

```
    def decoupe(self,i):
        """
        Fonction interne à triangle qui retire la 1ère ligne et la 1ère colonne

        """
        [lig, col], f = self.D, self.F
        return Mat([lig-1,col-1],lambda r,c : f(r+1,c+1))

```

```
    def remplace_ligned(self,i,g):
        """
        Fonction interne à triangle qui remplace dans la ligne i
        les coefficients à partir de la colonne i par ceux du tableau S
        """
        [lig, col], f = self.D, self.F
        h = lambda r,c: g(r-i,c-i) if r == i and c >= i else  f(r,c)
        return Mat([lig,col],h)
```

Recherche

Déterminez une fonction qui calcule rapidement le rang d'une matrice.

4.1.4 Calcul du déterminant

On triangularise la matrice et le déterminant est égal au produit des éléments diagonaux à un détail près : il ne faut pas oublier de tenir compte de la parité du nombre d'échanges de lignes ainsi que des multiplications des lignes modifiées par combinaisons.

Recherche

Créez une fonction qui calcule le déterminant :

```
    def det(self):
        """ renvoie le déterminant de self par Gauss-Jordan """
        ????
```

C'est assez efficace pour du Python basique : 555 ms pour un déterminant d'une matrice de taille 200.

```
In [1]: M = unite(200)
In [2]: %timeit M.det()
1 loops, best of 3: 555 ms per loop
```

4.1.5 Calcul de l'inverse d'une matrice

La trigonalisation est assez rapide du fait de la réduction progressive de S. L'idée est alors de trigonaliser de bas en haut puis de haut en bas pour arriver à une matrice diagonale ce qui sera plus efficace qu'un traitement de tout le tableau en continu.
On peut utiliser la fonction `det` qui est rapide et permet de ne pas prendre trop de précautions ensuite sachant que la matrice est inversible.

```
def inverse(self):
    return self.cat_carre_droite().triangle().diago_triangle().extrait_carre_droite()
```

Le tout est de construire cette méthode `diago_triangle` qui va trigonaliser le triangle en le parcourant de bas en haut.
On transforme légèrement les fonctions intermédiaires précedentes pour les adapter au nouveau parcours :

```
    def decoupe_bas(self):
        """
        Fonction interne à diago_triangle qui retire la dernière ligne et la
        colonne de même numéro de S

        """
        [lig, col], f = self.D, self.F
        #g = lambda r,c: f(lig-1,c) if r == lig else f(i,c) if r == lig-1 else f(r,c)
        g = lambda r,c: f(r,c) if c < lig - 1 else f(r,c+1)
        return Mat([lig-1,col-1],lambda r,c : g(r,c))
```

```
11
12      def remplace_ligneg(self,i,g):
13          """
14          Fonction interne à diago_triangle qui remplace dans la ligne i
15          les coefficients à partir de la colonne i par ceux du tableau S
16          """
17          [lig, col], f = self.D, self.F
18          h = lambda r,c: g(r,c - (lig - 1) + i) if r == i and c >= i else  f(r,c)
19          return Mat([lig,col],h)
```

Recherche

Construisez `diago_triangle`

```
1       def diago_triangle(self):
2           ????
```

Recherche

On considère la matrice

$$A = \begin{pmatrix} 10^{-20} & 0 & 1 \\ 1 & 10^{20} & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

Quel est son rang ? Quel est son déterminant ? Utilisez les fonctions précédentes puis du papier et un crayon...

5 Manipulation d'images

Dans cette section, des mathématiques concrètes vont nous permettre de réduire, agrandir, assombrir, éclaircir, compresser, bruiter, quantifier,... une photo. Pour cela, il existe des méthodes provenant de la théorie du signal et des mathématiques continues. Nous nous pencherons plutôt sur des méthodes plus légères basées sur l'algèbre linéaire et l'analyse matricielle. Une image sera pour nous une matrice carrée de taille 2^9 à coefficients dans $[\![0, 2^9 - 1]\!]$. Cela manque de charme ? C'est sans compter sur Lena qui depuis quarante ans rend les maths sexy (la population hantant les laboratoires mathématiques et surtout informatiques est plutôt masculine ...).
Nous étudierons pour cela la « Décomposition en Valeurs Singulières » qui apparaît de nos jours comme un couteau suisse des problèmes linéaires : en traitement de l'image et de tout signal en général, en reconnaissance des formes, en robotique, en statistique, en étude du langage naturel, en géologie, en météorologie, en dynamique des structures, etc.
Dans quel domaine travaille-t-on alors : algèbre linéaire, analyse, probabilités, topologie,... ? Un peu de tout cela et d'autres choses encore : cela s'appelle...la mathématique.

5.1 Lena

Nous allons travailler avec des images qui sont des matrices de niveaux de gris. Notre belle amie Léna sera représentée par une matrice carrée de taille 2^9 ce qui permet de reproduire Léna à l'aide de $2^{18} = 262\,144$ pixels. Léna prend alors beaucoup de place. Nous allons tenter de compresser la pauvre Léna sans pour cela qu'elle ne perde sa qualité graphique. Une des méthodes les plus abordables est d'utiliser la décomposition d'une matrice en valeurs singulières.
C'est un sujet extrêmement riche qui a de nombreuses applications. L'algorithme que nous utiliserons (mais que nous ne détaillerons pas) a été mis au point par deux très éminents chercheurs en 1965 (Gene Golub, états-unien et William Kahan, canadien, père de la norme IEEE-754). Il s'agit donc de mathématiques assez récentes, au moins en comparaison avec votre programme...
La petite histoire dit que des chercheurs américains de l'University of Southern California étaient pressés de trouver une image de taille 2^{18} pixels pour leur conférence. Passe alors un de leurs collègues avec, en bon informaticien, le dernier Playboy sous le bras. Ils décidèrent alors d'utiliser le poster central de la Playmate comme support...

La photo originale est ici : http://www.lenna.org/full/len_full.html mais nous n'utiliserons que la partie scannée par les chercheurs, de taille 5.12in × 5.12in...

5.2 Environnement de travail

Nous travaillerons dans Pylab avec l'option `pylab` qui charge les bibliothèques nécessaires à la manipulation d'images (en fait, pylab charge les bibliothèques pour se retrouver dans un environnement proche de celui de logiciels comme MatLab ou Scilab).

```
$ ipython --pylab
```

Les images sont alors sous forme d'`array` de la bibliothèque numpy.
Nous allons créer quelques outils pour continuer à travailler malgré tout avec notre classe `Mat`.

```
def array2mat(tab):
    """ convertit un array de numpy en objet de type Mat"""
    dim = list(tab.shape)
    return Mat(dim,  lambda i,j : tab[i,j])

def mat2array(mat):
    """ convertir un objet de type Mat en array de numpy """
    return np.fromfunction(mat.F,tuple(mat.D),dtype = int)

def montre(mat):
    """ permet de visualiser les images dans un terminal """
    return imshow(mat2array(mat), cmap = cm.gray)
```

Python contient un tableau carré de taille 512 contenant les niveaux de gris représentant Lena. Il suffit de cherger les bonnes bibliothèques :

Haskell

```
from scipy import misc

# lena sous forme d'un objet de type Mat
matlena = array2mat(misc.lena())
```

Lena est bien un carré de 512 × 512 pixels :

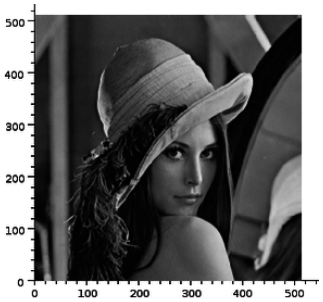
Haskell

```
In [9]: matlena.D
Out[9]: [512, 512]
```

On peut voir Lena :

```
In [3]: montre(matlena)
Out[3]: <matplotlib.image.AxesImage at 0x7f06c6d0c7b8>
```

et on obtient :



On peut préférer travailler sur une photo de format jpg (qui sera à l'envers) ou png (qui sera à l'endroit). Python la transforme en matrice avec la fonction imread. La matrice obtenue est en RVB : chaque coefficient est un vecteur (R,V,B). On la convertit en niveau de gris par exemple en calculant la moyenne des trois couleurs.
On récupère d'abord une image :

Haskell

```
In [4]: from urllib.request import urlretrieve

In [5]: urlretrieve('addresse de la photo en ligne','photo.png')
Out[5]: ('photo.png', <http.client.HTTPMessage at 0x7f3aeeb5fa20>)
```

et on la transforme en matrice :

Haskell

```
lena_face_couleur = imread('lena_face.jpeg')

def rvb_to_gray(m):
    r = len(m)
    c = len(m[0])
    return(np.array([[m[i,j][0] for j in range(c)] for i in range(r)]))

lena_face_array = rvb_to_gray(lena_face_couleur)

lena_face = array2mat(lena_face_array)
```

En exclusivité mondiale, voici Lena de face :

```
In [3]: montre(lena_face)
Out[3]: <matplotlib.image.AxesImage at 0x7f6d8c68cc88>
```

et on obtient :

5 3 Manipulations basiques

Comment obtenir les deux images de gauche sachant que l'image originale est à droite :

Sachant que l'échelle de gris varie entre 0 et 255, comment obtenir les images suivantes :

On peut modifier le contraste en « mappant » par une fonction croissant plus rapidement ou plus lentement que la fonction identité sur [0 ;255] vers [0 ;255].
Comment obtenir l'image en négatif de Lena ?

5.4 Résolution

Une matrice de taille 2^9 contient 2^{18} entiers codés entre 0 et 2^8-1 ce qui prend pas mal de place en mémoire. Peut-on être plus économique sans pour cela diminuer la qualité esthétique de la photo ?
La première idée est de prendre de plus gros pixels, c'est-à-dire une matrice plus petite : on extrait par exemple régulièrement un pixel sur k (en choisissant k parmi une puissance de 2 inférieure à 2^9).
Comment obtenir par exemple ces images ?

5.5 Quantification

On peut également réduire le nombre de niveaux de gris en regroupant par exemple tous les niveaux entre 0 et 63 en un seul niveau 0, puis 64 à 127 en 64, 128 à 191 en 128, 192 à 256 en 192.
Par exemple, avec 4, 8 puis 16 niveaux :

Pour cela, on divise chaque coefficient de la matrice par une puissance de 2.
Mais une division d'un nombre par une puissance de 2 revient à faire un décalage de la chaîne de bits représentant ce nombre vers la droite Ce décalage vers la droite est effectué en Python par

l'opérateur >>xcom]decalage !gauche@>>. L'image précedente peut donc être obtenue par ? ? ?

5 6 Enlever le bruit

La transmission d'informations a toujours posé des problèmes : voleurs de grands chemins, poteaux télégraphiques sciés, attaques d'indiens, etc.
Claude Elwood SHANNON (1916 - 2001) est un mathématicien-inventeur-jongleur américain qui, suite à son article « *A mathematical theory of communications* » paru en 1948, est considéré comme le fondateur de la *théorie de l'information* qui est bien sûr une des bases de... l'*informatique*.
L'idée est d'étudier et de quantifier l'« information » émise et reçue : quelle est la compression maximale de données digitales ? Quel débit choisir pour transmettre un message dans un canal « bruité » ? Quel est le niveau de sûreté d'un chiffrement ?...
La théorie de l'information de SHANNON est fondée sur des modèles probabilistes : leur étude est donc un préalable à l'étude de problèmes de réseaux, d'intelligence artificielle, de systèmes complexes.
Par exemple, dans le schéma de communication présenté par SHANNON, la source et le destinataire d'une information étant séparés, des perturbations peuvent créer une différence entre le message émis et le message reçu. Ces perturbations (bruit de fond thermique ou accoustique, erreurs d'écriture ou de lecture, etc.) sont de nature *aléatoire* : il n'est pas possible de prévoir leur effet. De plus, le message source est par nature *imprévisible* du point de vue du destinataire (sinon, à quoi bon le transmettre).
SHANNON a également emprunté à la physique la notion d'entropie pour mesurer le désordre de cette transmission d'information, cette même notion d'entropie qui a inspiré notre héros national, Cédric VILLANI...
Mais revenons à Lena. Nous allons simuler une Lena bruitée. Pour cela, nous allons ajouter ajouter des pixels aléatoirement selon une loi de Bernoulli de paramètre p.

```
1  from random import randint
2
3  def ber(p):
4      return 1 if random() < p else 0
5
6  def rand_image(dim,p):
7      return Mat(dim, lambda i,j : randint(0,255)*ber(p))
8
9  def bruit(mat,p):
10      return (mat + rand_image(mat.D,p)).map(lambda x : x % 256)
```

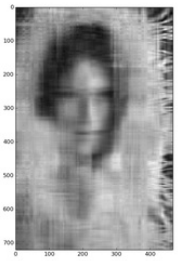
Par exemple, voici Lena bruitée à 0.1 :

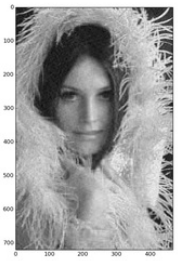
Il existe de nombreuses méthodes, certaines très élaborées, permettant de minimiser ce bruit. Une des plus simples est de remplacer chaque pixel par la moyenne de lui-même et de ses 8 autres voisins directs, voire aussi ses 24 voisins directs et indirect, voire plus, ou de la médiane de ces séries de « cercles » concentriques de pixels.
Voici les résultats avec respectivement la moyenne sur des 9 pixels puis 25 pixels et à doite la médiane de carrés de 9 pixels. La médiane est plus efficace que la moyenne !

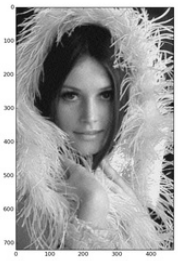
Do it !!

5 7 Compression par SVD

On en a déjà parlé.
Le problème est d'obtenir cette décomposition. Python heureusement s'en charge grâce à la fonction `linalg.svd(mat)` qui renvoie la décomposition en valeurs singulières sous la forme de trois « array » de numpy U, S et ${}^tV$.
Par souci d'efficacité, nous travaillerons ici uniquement avec les « array » de numpy.
Voici les trois niveaux de compression 10, 50 et 100 :





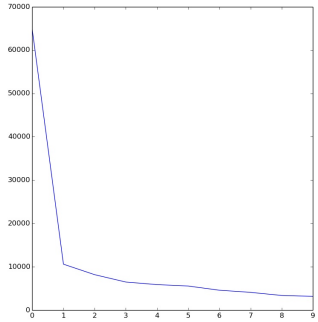


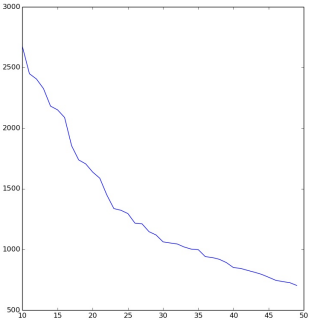
obtenus par :

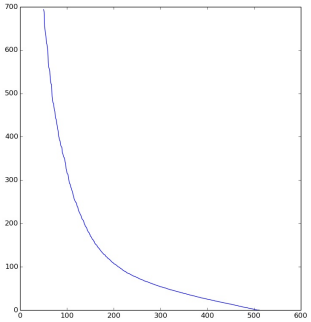
```
In [144]: imshow(c_svd(lena_face_array,100),cmap = cm.gray)
```

On peut observer la décroissance extrêmement rapide des valeurs singulières :

```
In [1]: U,s,tV = linalg.svd(misc.lena())

In [2]: x = np.arange(len(s))

In [3]: plot(x[:10],s[:10])

In [4]: plot(x[10:50],s[10:50])

In [5]: plot(x[50:],s[50:])
```







On peut également faire une petite animation des images compressées :

```
def anim(im,n):
    for k in range(n):
        plt.imsave("lena_face_svd" + str(1000 + k), c_svd(lena_face_array,k), cmap =
            ↪  cm.gray)
```

Puis ensuite dans un terminal :

```
$ apngasm anim_lena_face.png lena_face_svd*.png 1 5
```

Il faut installer au préalable le petit logiciel apngasm (disponible dans les dépôts Linux ou sur http://sourceforge.net/projects/apngasm/).
On obtient l'image animée suivante :
http://download.tuxfamily.org/tehessinmath/les_images/anim_lena_face.png

Recherche

Détection des bords

Pour sélectionner des objets sur une image, on peut essayer de détecter leurs bords, i.e. des zones de brusque changement de niveaux de gris.
On remplace alors un pixel par une mesure des écarts des voisins immédiats donnée par exemple par :

$$\sqrt{(m_{i,j+1} - m_{i,j-1})^2 + (m_{i+1,j} - m_{i-1,j})^2}$$

Écrivez une fonction qui trace le contour défini ci-dessus.

6 RECHERCHES

Recherche 2 - 1

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$. Expliciter la matrice A dans les cas suivants :

1. $a_{ij} = i + j$ si $i > j$ et $a_{ij} = 0$ sinon.
2. $a_{ij} = j$ si $i \leq j$ et $a_{ij} = i$ si $i > j$.
3. $a_{ij} = |i - j - 1|$

Recherche 2 - 2

$A = (a_{i,j}) \in \mathbb{A}^{n\times n}$ et $B = {}^t A \times A$.

1. Démontrer que B est symétrique.
2. $B = (b_{i,j})$, calculer $b_{i,j}$ en fonction des $a_{k,l}$.

Recherche 2 - 3

$A = (a_{i,j})$ est une matrice carrée d'ordre n à coefficients réels. $D = (d_{i,j})$ est une matrice carrée diagonale d'ordre n à coefficients réels ayant ses éléments diagonaux tous distincts. Démontrer :

$$A \times D = D \times A \rightarrow A \text{ est diagonale.}$$

Recherche 2 - 4

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$, on appelle trace de A la somme de ses éléments diagonaux que l'on note $\mathrm{tr}(A)$ ce nombre : $\boldsymbol{tr}(A) = \sum_{i=1}^{n} a_{ii}$. Démontrer :

$$A = (a_{ij}) \in \mathbb{R}^{n\times n} \text{ et } B = (a_{ij}) \in \mathbb{R}^{n\times n} \rightarrow \boldsymbol{tr}(A \times B) = \boldsymbol{tr}(B \times A)$$

Recherche 2 - 5

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$, calculer la trace de ${}^t A \times A$.

Recherche 2 - 6

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

1. Calculer A^2.
2. Calculer $A^2 - 3A + 2I_3$.
3. En déduire que A est inversible et déterminer A^{-1}.

Recherche 2 - 7

Déterminer les ℓ-réduites échelonnées (ℓré) des matrices suivantes et donner leur rang :

1. $A = \begin{pmatrix} 1 & 2 & 0 & 1 \\ -1 & 2 & 1 & 1 \\ 3 & 1 & 0 & -2 \end{pmatrix}$
2. $A = \begin{pmatrix} 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 2 & 1 & -1 & 0 & -1 & 2 \end{pmatrix}$
3. $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$
4. $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 5 & 10 \end{pmatrix}$

Recherche 2 - 8

Les matrices suivantes sont-elles inversibles ? On cherchera, pour chacune d'elles, l'inverse par la méthode de Gauss.

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

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

3. $C = \begin{pmatrix} 1 & 2 & -1 \\ -1 & -1 & 2 \\ 2 & 1 & 1 \end{pmatrix}$

4. $D = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

Recherche 2 - 9

Soit $D = \lceil d_{1,1}, d_{2,2}, \ldots, d_{n,n} \rfloor$ une matrice diagonale, démontrer que D est inversible si, et seulement si, $\prod_{i=1}^{n} d_{i,i} \neq 0$.

Recherche 2 - 10

a) Soit $U = \begin{pmatrix} 0 & 0 & 5 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix}$. Exprimer de façon simple la matrice U^3 en fonction de U^2, U et I_3. En déduire que U est inversible et donner U^{-1}.

b) Soit $U = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$. Déterminer $(a, b) \in \mathbb{R}^2$ tel que $(U - aI_3)(U - bI_3) = O_3$. En déduire que U est inversible et donner U^{-1}.

c) Inverser les matrices suivantes :

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 3 \end{pmatrix} \quad C = \begin{pmatrix} i & 2 & -3 \\ 0 & i & 2 \\ 0 & 0 & i \end{pmatrix} \quad D = \begin{pmatrix} 2 & 2 & 3 \\ 1 & -1 & 0 \\ -1 & 2 & 1 \end{pmatrix}$$

Recherche 2 - 11

Ecrire matriciellement de deux façons les systèmes suivants :

1. $\begin{cases} x - 2y + 3z = 2 \\ 2x - 5y - 4z = -1 \\ 4y - 3z = 8 \end{cases}$

2. $\begin{cases} 3x - z = 6 \\ 5y - 4z = -3 \\ 4x - 3y = 7 \end{cases}$

3. $\begin{cases} 5x + 2y + 3z = 2 \\ 2x - 5y - 6z = -2 \\ 4y - 3z = 8 \\ 4x + 3y + 2z = 0 \end{cases}$

Recherche 2 - 12

Résoudre le système :

$$\begin{cases} x - 2y + z = a \\ 2x - 3y - 2z = b \\ x - y + z = c \end{cases}$$

obligatoirement par la méthode Gauss-Jordan où x, y et z sont les inconnues. En déduire que la matrice $A = \begin{pmatrix} 1 & -2 & 1 \\ 2 & -3 & -2 \\ 1 & -1 & 1 \end{pmatrix}$ est inversible et donner son inverse.

Recherche 2 - 13

Résoudre les systèmes suivants par la méthode de Gauss-Jordan :

1. $\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 2 \\ 7x + 8y + 9z = 3 \end{cases}$

2. $\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 2 \\ 5x + 7y + 9z = 3 \end{cases}$

3. $\begin{cases} x + 2y + 3z + 4t = 1 \\ 4x + 5y + 6z + 2t = 2 \\ 7x + 8y + 9z - t = 3 \end{cases}$

4. $\begin{cases} x+y+z=1 \\ x-y-2z=3 \\ 2x+3y+5z=0 \\ 4x+3y+4z=4 \end{cases}$ **5.** $\begin{cases} x+y+z=1 \\ x-y-2z=3 \\ 2x+3y+5z=0 \\ 4x+3y+4z=1 \end{cases}$ **6.** $\begin{cases} x+y+z=1 \\ x-y-2z=3 \\ 2x-z=4 \\ 3x+y=5 \end{cases}$

Recherche 2 - 14

Déterminer le rang des systèmes linéaires d'inconnues réelles suivants, en fonction du paramètre $m \in \mathbb{R}$. On donnera l'ensemble des solutions.

$a) \begin{cases} mx+y+z=m \\ x+my+z=m \\ x+y+mz=m \end{cases}$ $b) \begin{cases} (m+1)x+my=2m \\ mx+(m+1)y=1 \end{cases}$ $c) \begin{cases} x-my+m^2z=2m \\ mx-m^2y+mz=2m \\ mx+y-m^2z=1-m \end{cases}$

Recherche 2 - 15

Soit la matrice $A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$.

a) Déterminer A^n pour tout $n \in \mathbb{N}$.

b) On considère les deux suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ définies par la donnée de u_0 et v_0 et les relations de récurrence

$$\begin{cases} u_{n+1} &= u_n - v_n \\ v_{n+1} &= -u_n + v_n \end{cases}$$

(i) On pose $W_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$. Etablir une relation entre W_{n+1}, A et W_n.

(ii) En déduire une expression de u_n et v_n en fonction de u_0, v_0 et n pour tout $n \in \mathbb{N}$.

Recherche 2 - 16

Soit $A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$, $B_1 = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$ et $B_2 = \begin{pmatrix} 32,1 \\ 22,9 \\ 33,1 \\ 30,9 \end{pmatrix}$

Résolvez les systèmes $A \times X = B_1$ et $A \times X = B_2$: commentaires ?

Recherche 2 - 17

Soit $A = \begin{pmatrix} 10^{-16} & 1 \\ 1 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$. Supposons que les nombres flottants soient représentés avec 10 chiffres significatifs. Résolvez le système $A \times X = B$ en choisissant 10^{-16} comme premier pivot. Recommencez en commençant par permuter les deux lignes puis en prenant 1 comme pivot. Généralisez en remplaçant 10^{-16} par un nombre strictement positif quelconque ε : des commentaires ?

Recherche 2 - 18

Dans toute la suite U désigne une suite de 4 bits qui peut être représentée par la matrice

$$U = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}$$

et la transposée de U est notée X :

$${}^tU = X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

et il est bien entendu que les éléments x_i sont considérés comme des éléments du corps $\mathbb{F}_2$ en ce sens que $1+1=0$. Pour améliorer la sécurité de la transmission des chaînes de 4 bits on décide de transformer le message U en

$$V = U \times B$$

où B est une matrice (judicieusement choisie) à coefficients dans $\mathbb{F}_2$ ou, si l'on préfère, on transforme le message X en $Y = A \times X$

1. Quelle relation existe-t-il entre Y et V ? Entre A et B ?
2. Si on veut que $V = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_2 \end{pmatrix}$, donner la matrice B.
3. Si on veut que $Y = \begin{pmatrix} x_1 \\ x_1+x_2 \\ x_3 \\ x_{1+}x_4 \\ x_2 \\ x_3 \end{pmatrix}$, donnner la matrice A.
4. On veut rajouter au message U un bit de parité paire (pour un bit de parité impaire, ce qui suit est impossible), décrire alors V et donner alors B ?
5. On veut que $V = \begin{pmatrix} x_4 & x_3 & x_2 & x_1 & x_1 & x_2 & x_3 & x_4 \end{pmatrix}$, déterminer B.
6. Quelqu'un décide de choisir

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

   i. Calculer les images des éléments suivants :

U	V
1111	
1100	
0011	
0110	
0101	
0010	
1010	
0100	

   ii. Qelle remarque faites-vous sur le choix de B ?

   iii. Donner une matrice B' ayant la même taille que B et qui ne possède pas à coup sûr le défaut précédent.

   iv. Donner une matrice B'' ayant la même taille que B et ayant le défaut (même pire si vous le voulez) de B.

Recherche 2 - 19

Soit $\mathcal{B} = (i, j)$ une base du plan ou $\mathcal{B} = (i, j, k)$ une base de l'espace, nous savons que tout vecteur se décompose de façon unique dans la base $\mathcal{B}$. Nous fixons un point O appelé origine, pour tout point M de notre plan ou espace, le vecteur $\overrightarrow{OM}$ se décompose de façon unique dans la base $\mathcal{B}$ et nous obtenons les coordonnées du vecteur $\overrightarrow{OM}$ qui correspondent au coordonnées du point M dans le repère $\mathcal{R} = (O, i, j)$ ou $\mathcal{R} = (O, i, j, k)$.
Ces coordonnées se notent, respectivement, par

$$M\begin{pmatrix} x \\ y \end{pmatrix}_{\mathcal{R}}, M\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathcal{R}}$$

Un changement de repère peut concerner un changement d'origine ou un changement de base ou les deux. Soit donc le repère $\mathcal{R} = (O, i, j, j)$ avec $\mathcal{B} = (i, j, k)$ la base associée et $\mathcal{R}' = (O', i', j', k')$ un autre repère avec $\mathcal{B}' = (i', j', k')$ la base associée. Nous notons

$$M\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathcal{R}}, M\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\mathcal{R}'}, O'\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}_{\mathcal{R}} \text{ et } M\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}_{\mathcal{R}_1=(O,i',j',k')}$$

et $P = (p_{i,j})$ la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$, c'est-à-dire la matrice des coordonnées des vecteurs de $\mathcal{B}'$ dans la base $\mathcal{B}$.

1. Quelle sont les coordonnées de O dans le repère $\mathcal{R}$?
2. Démontrer que P est inversible en cherchant les coordonnées des vecteurs de $\mathcal{B}$ dans la base $\mathcal{B}'$.
3. On note $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, X' = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}, X_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$. Démontrer que $X_1 = P \times X$.
4. Déterminer X' en fonction de X et de P.

Recherche 2 - 20

Une matrice est dite **orthogonale** si, et seulement si, sa transposée est égale à son inverse.

1. La matrice I_n est-elle orthogonale ?
2. La matrice $A = \begin{pmatrix} \sin(t) & \cos(t) \\ -\cos(t) & \sin(t) \end{pmatrix}$ est-elle orthogonale pour tout $t \in \mathbb{R}$?
3. Déterminez la troisième ligne de la matrice orthogonale : $\begin{pmatrix} 1/3 & 2/3 & 2/3 \\ 2/3 & -2/3 & 1/3 \\ & & \end{pmatrix}$