Mathématiques

IUT INFO 2 / 2012-2013

Mathématiques discrètes pour l'informatique (VI)

方程

Guillaume CONNAN

IUT NANTES
UNIVERSITÉ DE NANTES

TABLE DES MATIÈRES

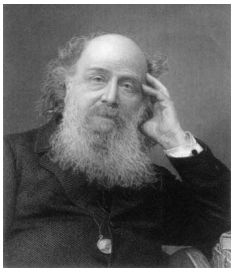
CHAPITRE 11

Calcul matriciel

Le calcul matriciel est très utilisé en réseaux, théorie des graphes, en infographie, en structures de données, etc. Bref, c'est une nouvelle notion indispensable à bien maîtriser par tous les informaticien(ne)s...

1 La matrice

1 1 Un peu d'histoire



J.SYLVESTER (1814-1897)

Vers 160 avant le petit Jésus, parut en Chine le Fang Tcheng ou, si vous préférez, « la disposition rectangulaire » qui désigne en mandarin actuel « équation mathématique ». C'est le huitième des *Neuf chapitres sur l'art mathématique*. Dès cette époque, les chinois regroupaient les coefficients des systèmes d'équations linéaires dans un tableau rectangulaire et transformait celui-ci pas à pas pour le résoudre. Cette méthode sera reprise vingt siècles plus tard par les Allemands Carl Friedrich GAUSS (1777 - 1855) et Wilhelm JORDAN (1842 - 1899)(à ne pas confondre avec le mathématicien français Camille JORDAN) et c'est elle que nous programmerons en CAML.
Le terme « matrice » pour désigner ces tableaux a été introduit en 1850 par James SYLVESTER en 1850. Son ami Arthur CAYLEY (1821 - 1855) introduisit le produit de matrices et la notion d'inverse cinq années plus tard. Le terme « matrice » est dérivé du latin *mater* qui signifie « mère »...

1 2 Qu'est-ce que c'est ?

1 2 1 Vecteur

Nous appellerons *vecteur* une liste d'éléments d'un ensemble $\mathbb{A}$. Cet ensemble $\mathbb{A}$ est muni de deux opérations, $\boxplus$ et $\boxdot$, qui lui confèrent une structure d'**anneau** :

- $(\mathbb{A}, \boxplus)$ a une structure de groupe commutatif ;
- $(\mathbb{A}, \boxdot)$ a une structure de monoïde ;
- $\boxdot$ est distributive sur $\boxplus$.

On désigne souvent par $0_{\mathbb{A}}$ l'élément neutre de $\boxplus$ et par $1_{\mathbb{A}}$ l'élément neutre de $\boxdot$.
$(\mathbb{A}, \boxdot)$ n'étant pas forcément un groupe, tout le monde n'admet pas forcément un inverse par $\boxdot$.

Recherche

Quels sont les éléments inversibles de $(\mathbb{Z}, \cdot)$?

Sur CAML, on créera un type anneau :

```
type 'a anneau =
  {zero      : 'a;
   un        : 'a;
   som       : 'a -> 'a -> 'a;
   prod      : 'a -> 'a -> 'a;
   sous      : 'a -> 'a -> 'a;
   div       : 'a -> 'a -> 'a;
   to_string : 'a -> string;
   egal      : 'a -> 'a -> bool;
   ordre     : 'a -> 'a -> bool;
  };;
```

Par exemple, l'anneau $(\mathbb{R}, +, \cdot)$ sera représenté par :

```
let reel =
  {zero      = 0.;
```

```
  un        = 1.;
  som       = ( +. );
  prod      = ( *. );
  sous      = ( -. );
  to_string = string_of_float;
  div       = ( /. );
  egal      = (=);
  ordre     = (>=);
 };;
```

Recherche

Définissez de la même manière l' anneau $(\mathbb{Z}, +, \cdot)$ sur CAML.

On notera $\mathbb{A}^p$ l'ensemble des vecteurs à coefficients dans $\mathbb{A}$ de *taille* p.

Notons $u = [a_1, a_2, ..., a_p]$ et $v = [b_1, b_2, ..., b_p]$ de vecteurs de même taille. On peut alors en faire la *somme* :

$$u \boxplus v = [a_1 \boxplus b_1, a_2 \boxplus b_2, ..., a_n \boxplus b_n]$$

On peut multiplier un vecteur par un *scalaire*, c'est-à-dire un élément de $\mathbb{A}$:

$$ku = [k \boxdot a_1, k \boxdot a_2, ..., k \boxdot a_p]$$

On notera en particulier $-u = (-1_{\mathbb{A}})u$.

On peut multiplier deux vecteurs composante par composante :

$$u \boxdot v = [a_1 \boxdot b_1, a_2 \boxdot b_2, ..., a_p \boxdot b_p]$$

Sur CAML, on utilisera à bon escient les fonctions `map` et `mapi`. Voici ce que dit la documentation :

```
val map : ('a -> 'b) -> 'a list -> 'b list
     List.map f [a1; ...; an] applies function f to a1, ..., an, and builds the list [f
          a1; ...; f an] with the results returned by f. Not tail-recursive.

val mapi : (int -> 'a -> 'b) -> 'a list -> 'b list
     Same as List.map, but the function is applied to the index of the element as first
          argument (counting from 0), and the element itself as second argument. Not tail
          -recursive.
```

Recherche

Que font alors les fonctions suivantes :

```
let map_vec = fun op vec ->
  map (fun x -> op x) vec;;

let vecs_op = fun op v1 v2 ->
    mapi ( fun i x -> op (nth v1 i) x ) v2;;
```

sachant que `nth` est défini par :

```
val nth : 'a list -> int -> 'a

    Return the n-th element of the given list. The first element (head of the list)
         is at position 0. Raise Failure "nth" if the list is too short. Raise
         Invalid_argument "List.nth" if n is negative.
```

1 2 2 Matrice

Une matrice à n lignes et p colonnes est un tableau d'éléments appartenant à un ensemble $\mathbb{A}$. On note alors $\mathbb{A}^{n\times p}$ l'ensemble des matrices de n lignes et p colonnes à coefficients dans $\mathbb{A}$.

$$\begin{array}{c} \\ i \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & & j & & \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,j} & \cdots & a_{1,p} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i,1} & a_{i,2} & \cdots & a_{i,j} & \cdots & a_{i,p} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,j} & \cdots & a_{n,p} \end{pmatrix} \end{array}$$

On note souvent a_{ij} le coefficients $a_{i,j}$ et M_{ij} le coefficient de M situé sur la ligne i et la colonne j.

En fait, une matrice est un vecteur dont les coefficients sont eux-mêmes des vecteurs ce qui facilitera notre tâche en programmation. Par exemple...

Recherche

...que vous inspire :

```
let mats_op = fun op m1 m2 ->
  vecs_op (vecs_op op) m1 m2;;

let rec make_vec long x =
  match long with
  |0 -> []
  |_ -> x::(make_vec (long-1) x);;

let rec make_mat row col x =
  match row with
  |0 -> []
  |_ -> (make_vec col x)::(make_mat (row -1) col x);;
```

1 3 Opérations sur les matrices

1 3 1 Égalité

Deux matrices sont égales si, et seulement si, elles ont les mêmes dimensions et des coefficients égaux.

1 3 2 Opération terme à terme

Notons $(S_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ la somme terme à terme des deux matrices $(A_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ et $(B_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ alors chaque coefficient de S est obtenu par :

$$S_{ij} = A_{ij} \boxplus N_{ij}$$

On peut définir de la même manière un produit terme à terme qu'il ne faudra pas confondre avec le produit de matrices ou le produit par un scalaire...

1 3 3 Multiplication par un scalaire

Soit $k \in \mathbb{A}$ et $M \in \mathbb{A}^{n\times p}$ alors la matrice $P = kM$ est définie par :

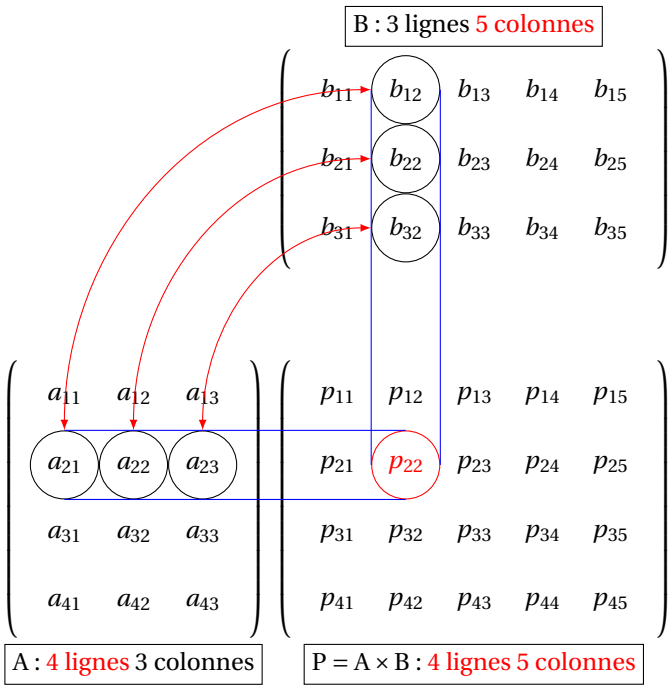
$$P_{ij} = k \boxdot M_{ij}$$

1 3 4 Produit de matrices

Un petit rappel...

Notons $(p_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant p}}$ le produit des deux matrices $(a_{ij})_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant m}}$ et $(b_{ij})_{\substack{1\leqslant i\leqslant m\\1\leqslant j\leqslant p}}$ alors

$$p_{ij} = \boxplus_{k=1}^{m} a_{ik} \boxdot b_{kj}$$



Nous mettrons au point des fonctions nous permettant de calculer le produit de deux matrices :

```
# let a = [[1;2;3];[4;5;6]];;
val a : int list list = [[1; 2; 3]; [4; 5; 6]]
# printm entier a;;
 |     1      2      3  |
 |     4      5      6  |

# let b = [[0;10;20;30];[-10;0;10;20];[-20;-10;0;10]];;
val b : int list list =
  [[0; 10; 20; 30]; [-10; 0; 10; 20]; [-20; -10; 0; 10]]
# printm entier b;;
 |     0     10     20     30  |
 |   -10      0     10     20  |
 |   -20    -10      0     10  |

# printm entier (prod_mat entier a b);;
 |   -80    -20     40    100  |
 |  -170    -20    130    280  |
```

Recherche

Montrez que $(\mathbb{A}^{n\times n}, +, \times)$ est un anneau. Est-il commutatif ?
On notera $\mathbb{I}_n$ l'élément neutre de $(\mathbb{A}^{n\times n}, \times)$: quelle est sa tête ?

1 3 5 Transposée d'une matrice

Soit $M \in \mathbb{A}^{n\times p}$. Sa transposée est la matrice T de $\mathbb{A}^{p\times n}$ définie par

$$T_{ij} = A_{ji}$$

On note le plus souvent $T = {}^tA$.

```
# printm entier a;;
 |     1      2      3  |
 |     4      5      6  |
- : unit = ()
# printm entier (transpose a);;
 |     1      4  |
 |     2      5  |
 |     3      6  |
- : unit = ()
```

En pratique, on échange lignes et colonnes.

Recherche

Démontrez que ${}^t(A \times B) = {}^tA \times {}^tB$.

1 4 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 ?

1 5 Opérations sur les lignes

1 5 1 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, $E_3^{12} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ 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 :

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

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 ?

```
# printm entier (elem entier 2 3 4);;
 |    0      0      0      0  |
 |    0      0      0      0  |
 |    0      0      0      1  |
 |    0      0      0      0  |
```

1 5 2 Transvections de lignes

On s'intéresse à la fonction :

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

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 $(\mathbb{I}_n + \lambda E_n^{ij})^{-1} = \mathbb{I}_n - \lambda E_n^{ij}$.

```
# printm entier (transvec entier 0 3 (-2) (unite entier 4));;
 |     1      0      0     -2  |
 |     0      1      0      0  |
 |     0      0      1      0  |
 |     0      0      0      1  |
```

1 5 3 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.

```
# printm entier (dilat entier 1 7 (attila entier (3,4)));;
 |     1      1      1      1  |
 |     7      7      7      7  |
 |     1      1      1      1  |
```

1 5 4 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 $L_i \leftrightarrow L_j$.

```
# printm entier (swap entier 0 1 [[11;12;13];[21;22;23];[31;32;33]]);;
 |    21     22     23  |
 |    11     12     13  |
 |    31     32     33  |
```

1 5 5 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 $\varphi^{-1}(\mathbb{I}_n)$.

1 5 6 Lien avec les matrices régulières

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

Théorème 11 - 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 :

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

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 11 - 2

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

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

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

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

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 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 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

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

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

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

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 2 L réduite échelonnée

La matrice $M = \left(m_{ij}\right) \in \mathbb{A}^{n\times p}(\mathbb{K})$ 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).

1

Zone du prochain pivot

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 11 - 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 $N = \text{lré}(M)$.

La démonstration de ce théorème se fait par récurrence sur n et elle est un peu difficile.
Le **rang** de M, noté 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 $\text{rang}(M') \leq \text{rang}(M)$.

3 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$\cdots$). 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.

```
# printm entier ( lre entier [[2;5;-2;3];[3;6;3;6];[1;2;-1;2]]);;
 |     1      0      0      4  |
 |     0      1      0     -1  |
 |     0      0      1      0  |
```

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 11 - 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.

4 Déterminant d'une matrice carrée

4 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 $\mathrm{A}=\begin{pmatrix} a & a' \\ b & b' \end{pmatrix}$ le nombre $ab'-a'b$ et on note :

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

4 2 Propriétés des déterminants

Dans la suite, $\mathrm{A}=(a_{ij})$, $\mathrm{B}=(b_{ij})$ 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 11 - 1

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

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

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

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

C'est-à-dire :

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

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

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

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 11 - 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 11 - 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 11 - 4

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

Démontrez cette propriété.

Propriété 11 - 5

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

Démontrez cette propriété.

Propriété 11 - 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 11 - 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. ${}^{t}A = -A$) d'ordre 37 est nul.

Propriété 11 - 8

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

Démontrez cette propriété en utilisant la propriété 11 - 3 page précédente.

Propriété 11 - 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 11 - 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 11 - 11

Soit A et B deux matrices carrées de même taille alors

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

Nous admettrons cette propriété. Démontrez alors les propriétés suivantes :

Propriétés 11 - 12

- Si A est inversible alors $\det A \neq 0$;
- Si A est singulière (non-inversible) alors $\det A = 0$;
- A est inversible si, et seulement si, $\det A \neq 0$.

5 Résolution de systèmes

5 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 :

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

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}

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

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

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

5 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

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

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

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

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

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

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 :

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

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 rang(T) = 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 $\left(p-r\right)$ 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.

EXERCICES

Exercice 11 - 1

On considère les matrices suivantes à coefficients réels :

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 2 & -1 & 2 \\ 0 & 1 & 3 \end{pmatrix}, B = \begin{pmatrix} 2 & 0 \\ 1 & 5 \\ -1 & -3 \end{pmatrix}, C = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, E = \begin{pmatrix} 1 & 2 & 4 & -3 \\ 1 & -2 & 0 & 5 \\ 0 & 3 & 1 & 6 \end{pmatrix}, F = \begin{pmatrix} 7 & 0 \\ 5 & -6 \\ -3 & 0 \\ -1 & 1 \end{pmatrix}, G = \begin{pmatrix} 1 & 2 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 1 \end{pmatrix}$$

1. Calculer : $A - 2G$, $3A + 2G$, ${}^tA$, ${}^tG$.
2. Calculer $B \times A, A \times B, E \times F, D \times C, C \times D, A^2, A \times G, G \times A$.

Exercice 11 - 2

Soit $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ et $B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ deux matrices booléennes. Calculez $A \times B$, $B \times A$, A^2.

Exercice 11 - 3

A et B sont deux matrices carrées d'ordre n, développer

1. $(A + B)^2$
2. $(A + B)^3$

Exercice 11 - 4

$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$, calculer A^n.

Exercice 11 - 5

$A \in \mathbb{A}^{3\times 2}$, peut-on trouver une matrice B de sorte que $A \times B = B \times A$?

Exercice 11 - 6

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$. 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. $a_{ij} = |i - j - 1|$

Exercice 11 - 7

$A = (a_{i,j}) \in \mathbb{A}^{n\times n}$ et $B = {}^tA \times A$.

1. Démontrer que B est symétrique.

2. $B = \left(b_{i,j}\right)$, calculer $b_{i,j}$ en fonction des $a_{k,l}$.

Exercice 11 - 8

$A = (a_{i,j})$ 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 :

$$A \times D = D \times A \rightarrow A \text{ est diagonale.}$$

Exercice 11 - 9

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$, on appelle trace de A la somme de ses éléments diagonaux que l'on note tr(A) ce nombre : $\mathbf{tr}(A) = \sum_{i=1}^{n} a_{ii}$. Démontrer :

$$A = (a_{ij}) \in \mathbb{R}^{n\times n} \text{ et } B = (a_{ij}) \in \mathbb{R}^{n\times n} \rightarrow \mathbf{tr}(A \times B) = \mathbf{tr}(B \times A)$$

Exercice 11 - 10

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$, calculer la trace de ${}^tA \times A$.

Exercice 11 - 11

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

1. Calculer A^2.

2. Calculer $A^2 - 3A + 2I_3$.

3. En déduire que A est inversible et déterminer A^{-1}.

Exercice 11 - 12

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. $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 5 & 10 \end{pmatrix}$

Exercice 11 - 13

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. $D = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

Exercice 11 - 14

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, $\prod_{i=1}^{n} d_{i,i} \neq 0$.

Exercice 11 - 15

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 :

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 3 \end{pmatrix} \quad C = \begin{pmatrix} i & 2 & -3 \\ 0 & i & 2 \\ 0 & 0 & i \end{pmatrix} \quad D = \begin{pmatrix} 2 & 2 & 3 \\ 1 & -1 & 0 \\ -1 & 2 & 1 \end{pmatrix}$$

Exercice 11 - 16

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.** $\begin{cases} 5x + 2y + 3z = 2 \\ 2x - 5y - 6z = -2 \\ 4y - 3z = 8 \\ 4x + 3y + 2z = 0 \end{cases}$

Exercice 11 - 17

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.

Exercice 11 - 18

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.** $\begin{cases} x+y+z=1 \\ x-y-2z=3 \\ 2x-z=4 \\ 3x+y=5 \end{cases}$

Exercice 11 - 19

Déterminer le rang des systèmes linéaires d'inconnues réelles suivants, en fonction du paramètre $m \in \mathbb{R}$. On donnera l'ensemble des solutions.

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

Exercice 11 - 20

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 $n \in \mathbb{N}$.

Exercice 11 - 21

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 ?

Exercice 11 - 22

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 ?

Exercice 11 - 23

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

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

a. Calculer les images des éléments suivants :

U	V
1111	
1100	
0011	
0110	
0101	
0010	
1010	
0100	

b. Qelle remarque faites-vous sur le choix de B ?

c. Donner une matrice B' ayant la même taille que B et qui ne possède pas à coup sûr le défaut précédent.

d. Donner une matrice B'' ayant la même taille que B et ayant le défaut (même pire si vous le voulez) de B.

Exercice 11 - 24

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.

Exercice 11 - 25

Les déterminants peuvent servir à calculer certaines surfaces. On peut montrer par exemple que l'aire du triangle dont les sommets sont les points $A(x_1, y_1)$, $B(x_2, y_2)$ et $C(x_3, y_3)$ est égal à la valeur absolue de

$$\frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Quelle est l'aire du triangle dont les sommets ont pour coordonnées (2,3), (1,1) et (5,-1) ? (3,4), (2,1) et (4,7) ? Qu'en concluez-vous dans ce dernier cas ?

Exercice 11 - 26

Soit $A = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$ et $B = \begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix}$
Calculez det(A), det(B) et det(A + B). Conclusion ?

Exercice 11 - 27

Résolvez les équations suivantes dans $\mathbb{R}$:

1. $\begin{vmatrix} 3 & x \\ x+2 & 1-x \end{vmatrix} = -3$

2. $\begin{vmatrix} x-2 & 3 \\ x & x+1 \end{vmatrix} = 3$

Exercice 11 - 28

Pour quelles valeurs de x les matrices suivantes sont-elles singulières ?

1. $M_1 = \begin{bmatrix} 1 & x-1 \\ x & 3x \end{bmatrix}$

2. $M_2 = \begin{bmatrix} 2x & x+1 \\ x-2 & 3x \end{bmatrix}$

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

4. $M_4 = \begin{bmatrix} 1 & x & x^2 \\ 1 & 1 & 1 \\ 1 & 3 & 9 \end{bmatrix}$

Exercice 11 - 29

$A_{ij} = \begin{cases} 2 \text{ si } i \leqslant j \\ 0 \text{ si } i > j \end{cases}$. Calculer det(A). Faites de même si $A_{ij} = \begin{cases} i \text{ si } i = j \\ 0 \text{ si } i \neq j \end{cases}$.

Exercice 11 - 30

Soit $M = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$ et $\det(M) = -4$. Calculez le déterminant des matrices suivantes :

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

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

$C = \begin{bmatrix} 3a & d & -g \\ 3b & e & -h \\ 3c & f & -i \end{bmatrix}$

$D = \begin{bmatrix} a & d & g \\ 4b+3a & 4e+3d & 4h+3g \\ c & f & i \end{bmatrix}$

Exercice 11 - 31

Soient A, B et P trois matrices de même ordre telles que $B = P^{-1}AP$. Calculez le déterminant de B en fonction de celui de A.

Exercice 11 - 32

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 $A = \begin{bmatrix} \sin(t) & \cos(t) \\ -\cos(t) & \sin(t) \end{bmatrix}$ est-elle orthogonale pour tout $t \in \mathbb{R}$?

3. Montrez que le déterminant d'une matrice orthogonale est 1 ou -1.

4. Déterminez la troisième ligne de la matrice orthogonale : $\begin{bmatrix} 1/3 & 2/3 & 2/3 \\ 2/3 & -2/3 & 1/3 \\ & & \end{bmatrix}$

Chiffrement de Hill

On travaille avec l'alphabet $\{@, A, B, C, D, ..., Z\}$, le caractère @ symbolisant l'espace.
On fait correspondre à chaque lettre un élément de $\mathbb{Z}/27\mathbb{Z}$ soit respectivement $\{0, 1, 2, 3, 4, ..., 26\}$.
Le chiffrement proposé par Lester HILL en 1929 s'effectue en plusieurs étapes :

1. on choisit une matrice K de $\left(\mathbb{Z}/_{27\mathbb{Z}}\right)^{n\times n}$ inversible : c'est notre clé de chiffrement.
2. on regroupe chaque bloc de n lettres consécutives du message clair M en blocs d

Nous allons utiliser nos outils de calcul matriciel en cryptographie. En 1929, Lester HILL (1891 - 1961) proposa une méthode de chiffrement par blocs de lettres en travaillant avec des matrices à coefficients dans $\mathbb{Z}/26\mathbb{Z}$ (combien y a-t-il de lettres dans notre alphabet ?). Par la suite nous noterons $\mathbb{Z}_n$ l'ensemble $\mathbb{Z}/n\mathbb{Z}$.
Nous allons explorer cette méthode en rajoutant un vingt-septième caractère qui représentera un espace entre les lettres.

Codage

On considère les vingt-six lettres de l'alphabet plus une espace représenté par le caractère `@`. La première étape est de passer d'un caractère à un nombre appartenant à $\mathbb{Z}_{27}$.
La plupart des langages de programmation, et bien sûr CAML, ont une commande qui associe le code ASCII (éventuellement étendu) à un caractère. On pourra donc l'utiliser pour résoudre notre problème. Il s'agit de la commande `code(caratère)` de la bibliothèque **`Char`** qui renvoie le code ASCII du caractère `caractère`.
Par exemple :

```
# open Char;;
# code('@');;
- : int = 64
```

Mais attention ! Cela ne fonctionne par pour les chaînes de caractères :

```
# code('AB');;
Characters 5-6:
  code('AB');;
       ^
Error: Syntax error
```

Il faudra donc considérer les caractères d'une chaîne un par un.
Définissez une fonction **`code(chaîne)`** qui transformera une chaîne en la liste des « numéros » des caractères qui la compose, A portant le numéro 1, Z le numéro 26 et @ le numéro 0.
Par exemple :

```
# encode "COUCOU@MAMAN";;
- : int list = [3; 15; 21; 3; 15; 21; 0; 13; 1; 13; 1; 14]
```

Pour cela, on commencera par créer une fonction **`explose`** qui transforme une chaine en la liste des caractères qui la composent. On utilisera :

```
let tete_ch s = String.get s 0;;

let queue_ch s = String.sub s 1 (String.length s -1);;
```

Par exemple :

```
# explose "COUCOU@MAMAN";;
- : char list = ['C'; 'O'; 'U'; 'C'; 'O'; 'U'; '@'; 'M'; 'A'; 'M'; 'A'; 'N']
```

Tant que vous y êtes, créez une fonction **`decode`** qui effectuera l'opération inverse.
Par exemple :

```
# decode [3; 15; 21; 3; 15; 21; 0; 13; 1; 13; 1; 14];;
- : string = "COUCOU@MAMAN"
```

On pourra utiliser **`List.rev_map`**, **`Char.escaped`**, **`Char.chr`** et **`String.concat`**. Voici des extraits de la documentation des différentes bibliothèques impliquées :

```
val chr : int -> char
Return the character with the given ASCII code.
Raise Invalid_argument "Char.chr" if the argument is outside the range 0--255.

val escaped : char -> string
Return a string representing the given character,
with special characters escaped following the lexical conventions of OCaml.

val concat : string -> string list -> string
String.concat sep sl concatenates the list of strings sl,
inserting the separator string sep between each.

val rev_map : ('a -> 'b) -> 'a list -> 'b list
List.rev_map f l gives the same result as List.rev (List.map f l), but is tail-recursive and
```

Choisissons à présent un message à coder, assez long :

Je ne suis pas un numéro, je suis un homme libre !

Mettons tout en majuscules, enlevons la ponctuation et les accents et codons :

```
# let lc = encode("JE@NE@SUIS@PAS@UN@NUMERO@JE@SUIS@UN@HOMME@LIBRE");;
```

Le principe consiste alors à regrouper les nombres par paquets de longueur constante p et donc à construire à partir de la liste **L** une matrice de p lignes complétée éventuellement de zéros où les codes des caractères seront rentrés colonne par colonne.
Nous allons pour cela créer une fonction **mat_of_list liste p** qui transforme une liste en une matrice de p colonnes puis nous utiliserons la fonction **transpose** créée précédemment.
Attention ! Nous devons travailler modulo 27.
On commencera par créer un type **classe**

```
type classe  =
  {representant : int ; modulo : int};;
```

Vous reverrez les résultats obtenus en arithmétique l'an passé.
Créez ensuite un anneau général **quotient n** puis on posera :

```
# let z27 = quotient 27;;
val z27 : classe anneau =
  {zero = {representant = 0; modulo = 27};
   un = {representant = 1; modulo = 27}; som = <fun>; prod = <fun>;
   sous = <fun>; div = <fun>; to_string = <fun>; egal = <fun>; ordre = <fun>}
```

Par exemple, on pourra obtenir la lré d'une matrice modulo 27 :

```
# let m27 = [[1%27 ; 22%27 ; 25%27] ; [0%27 ; 25%27 ; 1%27] ; [25%27 ; 3%27 ; 1%27]];;
val m27 : classe list list =
  [[{representant = 1; modulo = 27}; {representant = 22; modulo = 27};
    {representant = 25; modulo = 27}];
   [{representant = 0; modulo = 27}; {representant = 25; modulo = 27};
    {representant = 1; modulo = 27}];
   [{representant = 25; modulo = 27}; {representant = 3; modulo = 27};
    {representant = 1; modulo = 27}]]
# printm (z27) m27;;
 |  1%27  22%27  25%27  |
 |  0%27  25%27   1%27  |
 | 25%27   3%27   1%27  |

- : unit = ()
```

```
# printm z27 (lre z27 m27);;
 |  1%27   0%27   0%27  |
 |  0%27   1%27   0%27  |
 |  0%27   0%27   1%27  |

- : unit = ()
```

Bref, le message en clair devient une matrice. Choisissons par exemple un 3-chiffrement de Hill.
Il faut choisir à présent une clé de chiffrement : ce sera une matrice carrée de taille 3, à coefficients dans $\mathbb{Z}_{27}$ et inversible dans $\mathbb{Z}_{27}$. Nous allons vérifier que la matrice $\begin{pmatrix} 1 & 22 & 25 \\ 0 & 25 & 1 \\ 25 & 3 & 1 \end{pmatrix}$ convient.
Utilisez à bon escient les fonctions crées autour de l'algorithme Fang-Tcheng.
On obtient donc enfin le message crypté à transmettre :

"LQVPQNDUGNPENPAEZ@RYUTFEULHYJXVPLEZ@OJOOQPQKETQF".

Si on le compare au message initial :
JE@NE@SUIS@PAS@UN@NUMERO@JE@SUIS@UN@HOMME@LIBRE
on remarque que le premier E est codé par Q alors que le dernier est codé par F : ce n'est donc pas un codage lettre par lettre et il est donc apparemment plus compliqué que le chiffrement de César.
Nous pouvons résumer tout le codage en une seule fonction.

```
# codage ("JE@NE@SUIS@PAS@UN@NUMERO@JE@SUIS@UN@HOMME@LIBRE") cle
- : string = "LQVPQNDUGNPENPAEZ@RYUTFEULHYJXVPLEZ@OJOOQPQKETQF"
```

Décodage

Celui qui reçoit ce message possède la clé de décodage qui est **inv_cle**.
Les étapes sont à peu près les mêmes que pour le codage. Proposez une fonction **decodage message_crypte inv_cle** qui retournera le message décodé à partir du message crypté reçu.

Algèbre linéaire

Exercice 11 - 33

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 $\text{Vect}\left(\mathcal{F}'\right) = \mathbb{R}^2$.

Exercice 11 - 34

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

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

Démontrer que F est un sev de E puis déterminer deux matrices M_1 et M_2 vérifiant $\mathrm{F} = \text{Vect}(\mathcal{F})$ avec $\mathcal{F} = (\mathrm{M}_1, \mathrm{M}_2)$. Cette famille est-elle libre ? A-t-elle d'autres propriétés ?

Exercice 11 - 35

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

Exercice 11 - 36

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

Exercice 11 - 37

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

Exercice 11 - 38

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

Exercice 11 - 39

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 $\mathbb{R}^4$?

Exercice 11 - 40

1. On note $\mathrm{H} = \{(x,y,z) \in \mathbb{R}^3 \mid x+2y+3z=0\}$, démontrer que H est un sev de $\mathbb{R}^3$ en déterminant u et v, deux vecteurs de $\mathbb{R}^3$, de sorte que $\mathrm{H} = \mathrm{Vect}(u,v)$.
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 $w \notin \mathrm{H}$.
7. On note $\mathrm{K} = \mathrm{Vect}(w)$. Donner une base de K.
8. Quelle est la dimension de K ?
9. Démontrer que (u,v,w) est une base de $\mathbb{R}^3$.

Exercice 11 - 41

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

Exercice 11 - 42

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

Exercice 11 - 43

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 $\mathrm{Vect}(u,v) = \{(x,y,z) \in \mathbb{R}^3 \mid ax+by+cz=0\}$.

Exercice 11 - 44

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 ?

Exercice 11 - 45

$u_1 = (1,2)$ 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 $\mathcal{B}' = (-3u_2, 7u_1)$.

Exercice 11 - 46

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. 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 $\mathbb{R}^3$.

Exercice 11 - 47

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

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 ?

Exercice 11 - 48

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.

- **a.** Que représentent les colonnes de A ?
- **b.** Déterminer $\mathcal{B}$ la matrice de la famille $\left(f(u_1), f(u_2), f(u_3)\right)$ dans la base $\mathcal{B}$.
- **c.** On note $\ker\left(f\right)$ le noyau de f, définir $\ker\left(f\right)$.
- **d.** 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 $\mathcal{B}$.
- **e.** Déterminer une base de $\ker\left(f\right)$, on notera $\mathcal{F}_{\ker}$ cette famille.
- **f.** Donner une famille génératrice de $\mathbf{Im}\left(f\right)$.
- **g.** Déterminer une base de $\mathbf{Im}\left(f\right)$. On notera $\mathcal{F}_{\mathbf{Im}}$ cette famille.

4. On note $\mathcal{B}' = \left(e'_1, e'_2, e'_3\right)$ la famille obtenue en juxtaposant le ou les vecteurs de $\mathcal{F}_{\mathbf{Im}}$ 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.

- **a.** Déterminer H la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$.
- **b.** On note U la matrice de f relativement ou dans la base $\mathcal{B}'$, U = $\mathbf{Mat}_{\mathcal{B}'}\left(f\right)$, 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 :

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

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

- **a.** Déterminer V la matrice de g dans la base $\mathcal{B}'$.
- **b.** Déterminer une base de $\ker\left(g\right)$.
- **c.** Déterminer une base de $\mathbf{Im}\left(g\right)$.
- **d.** Exprimer en fonction de V et de H la matrice de g dans la base $\mathcal{B}$.
- **e.** Exprimer à l'aide "de lettres" la matrice de $g \circ f$ dans la base $\mathcal{B}'$.

Exercice 11 - 49

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

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

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

Exercice 11 - 50

$$\mathrm{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$ 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 $\mathbf{Im}\varphi$. Quelle est la dimension de $\mathbf{Im}\varphi$? φ est-elle surjective ? Donner une matrice génératrice de $\mathbf{Im}\varphi$ dans la base canonique de $\mathbb{F}_2^5$.

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

7 Matrices et CAML

7 1 Comment modéliser ?

Au moment de choisir une modélisation pour nos matrices, l'informaticien(ne) se pose des questions : vais-je choisir un type **`array`** mutable ou un type **`list`** immuable ?
Ça se discute. Le premier peut paraître plus maniable a priori : on a accès à tout le tableau construit statiquement ce qui ne nous gène pas car on connaît la taille de la matrice a priori. D'un autre côté, un type mutable est dangereux : il laisse la porte ouverte aux effets de bord, source de nombreux bugs car souvent mal maîtrisés. De plus, la machine ne gère pas très efficacement les tableaux statiques.
Tournons-nous vers les listes : un langage muni d'un bon ramasse-miettes les traitera très efficacement, voire plus efficacement qu'un tableau statique. Elles permettent de créer dynamiquement les matrices ce qui engendre plus de souplesse et cela sans effets de bord ce qui assure plus de sécurité dans la programmation.
Nous choisirons donc de programmer sans effet de bord : du fonctionnel aussi pur que possible. Cela entraîne un changement des habitudes : au lieu de raisonner en termes de grosses boucles imbriquées et pas toujours contrôlées, nous utiliserons des petits modules fonctionnels. Cela pourra avoir quelques désavantages dans certains cas en termes d'efficacité mais nous améliorerons la performance dans un deuxième temps.
Nous essaierons dans une certaine mesure de créer des fonctions polymorphes : cela pourra passer pour une contrainte forte au départ mais cela vous facilitera la vie pour votre mini-projet sur le chiffrement de HILL : vous n'aurez qu'à changer d'anneau mais toutes les fonctions sur les matrices resteront valables sans aucun changement !

7 2 Le type anneau

Nous travaillerons sur des matrices de $\mathbb{A}^{n\times m}$ c'est-à-dire, des matrices de n lignes et m colonnes dont les coefficients appartiennent à un ensemble $\mathbb{A}$ muni de deux opérations $\boxplus$ et $\boxdot$ vérifiant un certain nombre de propriétés (voir cours...). Nous aurons besoin des propriétés des opérations sur $\mathbb{A}$ pour créer les opérations sur $\mathbb{A}^{n\times m}$.
Nous utiliserons un enregistrement :

```
type 'a anneau =
  {zero      : 'a;
   un        : 'a;
   som       : 'a -> 'a -> 'a;
   prod      : 'a -> 'a -> 'a;
   sous      : 'a -> 'a -> 'a;
   div       : 'a -> 'a -> 'a;
   to_string : 'a -> string;
   egal      : 'a -> 'a -> bool;
   ordre     : 'a -> 'a -> bool;
  };;
```

Recherche

Créez les anneaux **`entier`** correspondant à $(\mathbb{Z},+,\times)$ et **`reel`** correspondant à $(\mathbb{R},+,\times)$.
Créez également l'anneau booléen $\mathcal{B}_2$ dont les deux seuls éléments sont **`true`** et **`false`**.

L'appel de champs des enregistrements se fait à l'aide du point comme en POO :

```
# entier.zero;;
- : int = 0
# reel.zero;;
- : float = 0.
# entier.som 2 3;;
- : int = 5
# entier.to_string 2;;
- : string = "2"
```

7 3 Joli affichage

Une matrice sera modélisée par une liste de listes (la liste de ses lignes). Par exemple $\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}$ sera modélisée par **`[[1;2;3];[4;5;6]]`** ce qui n'est pas joli. Nous allons donc créer une fonction permettant un joli affichage : c'est de l'E/S non fonctionnel mais nous sommes tolérants.
Comme dans de nombreuses situations à venir, nous nous occupons d'abord d'une ligne :

```
let rec printv a v =
match v with
  |[] -> print_string " | ";
  |t::q ->
    Printf.printf "%7s" (" "^(a.to_string t)^" ");
    (printv a q);;
```

Nous précisons l'anneau a utilisé pour pouvoir appeler la bonne fonction d'affichage (pas de surcharge javaesque...) :

```
# printv entier [1;2;3];;
     1      2      3  | - : unit = ()
```

Le type **`unit`**, c'est tout ce qui n'est pas fonctionnel. On retrouve le **`printf`** « à la C ».
Ensuite, on parcours une liste de liste :

```
let rec printm a m =
match m with
  |[] -> print_newline();
  |t::q ->
    print_string " |";
    printv a t;
    print_newline();
    printm a q;
;;
```

On fait du mutable donc on utilise l'horrible `;` de Java...

```
# printm entier [[1;2;3];[4;5;6]];;
 |     1      2      3  |
 |     4      5      6  |

- : unit = ()
```

7 4 Remplissage de matrices

Nous voudrions construire des matrices Attila ou des matrices remplies de zéros voire d'éléments identiques quelconques.

On commencera comme toujours par fabriquer des lignes puis des listes de lignes de **x**.
On pense à cela :

```
let make_vec long x =
   match long with
   |0 -> acc
   |_ -> x::(make_vec (long-1) x);;
```

Le `::` ou l'ajout en tête est l'opérateur le plus important en programmation fonctionnelle. Cela consiste en fait à rajouter une branche à notre peigne.
Le problème est que cette fonction n'est pas récursive terminale (« *tail recursive* » : nous en avons parlé l'an passé).
Il est très simple d'y remédier :

```
let make_vec long x =
  let rec local long x acc =
      match long with
      |0 -> acc
      |_ -> local (long-1) x (x::acc)
  in local long x [];;
```

Recherche

À vous de jouer pour fabriquer :

```
(* une matrice de taille row*col remplie de x *)
let make_mat row col x = ...

(* matrice remplie de 1  *)
let attila a (r,c) = ...

(* matrice remplie de neutres *)
let zeros a (r,c) = ...
```

Vous obtiendrez par exemple :

```
# printm booleen (attila booleen (2,3));;
 |  vrai   vrai   vrai  |
 |  vrai   vrai   vrai  |
```

7.5 Utilitaires de collecte d'informations

Pour calculer la longueur d'une liste, on dispose de la fonction **`length`**.

Recherche

Déduisez-en les deux fonctions :

```
(* nb de lignes *)
let n_rows = function ...

(* nb de colonnes *)
let n_cols = function
```

Pour s'amuser, on peut la fabriquer de différentes façons. Une rigolote consiste à utiliser **`fold_left`**, le pliage d'une liste par la gauche qui est très utile en programmation fonctionnelle.

Voici un extrait de la documentation :

```
val fold_left : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a
List.fold_left f a [b1; ...; bn] is f (... (f (f a b1) b2) ...) bn.
```

On peut l'utiliser ainsi :

```
let compte a somme_partielle y = a.som a.un somme_partielle;;

let long a liste = fold_left (compte a) a.zero liste;;

# long entier [1;2;3;4;5;6;6];;
- : int = 7
```

Pour extraire la n^e composante d'une liste on dispose de la fonction **nth**.

Recherche

Déterminez les fonctions :

```
(* extraction de la ligne no i de la matrice mat *)
let extract_line = function
   mat -> ...

(*extraction de la colonne no i de la matrice mat *)
let rec extract_col = function
   match mat with
   | ...
(* extraction de la 1ère colonne *)
let tete_col = function
   mat -> ...

(* extraction de la matrice privée de sa 1ère colonne *)
let rec queue_col = function
   ...
```

Vous pouvez essayer de rendre vos récursions terminales.
Pour rendre la dernière fonction terminale, on pourrait utiliser la fonction **append** qui rajoute un nœud au peigne par la droite mais cette fonction n'est pas récursive terminale.
Nous allons donc plutôt utiliser **rev** qui retourne la liste retournée...

Définissez enfin une fonction **transpose** qui renvoie la transposée d'une matrice en utilisant les fonctions précédentes.

7 6 Opérations sur les matrices

Nous allons faire un peu de polymorphisme, comme en Java, mais sans surcharge, contrairement à Java.
Nous voudrions dans un premier temps effectuer une opération terme à terme sur des vecteurs : par exemple, la somme terme à terme de [1;2;3] et [4;5;6] est [5;7;9], le produit terme à terme est [4;10;18], etc.
Regardons la description du module **List** :

```
val map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
List.map2 f [a1; ...; an] [b1; ...; bn] is [f a1 b1; ...; f an bn]. Raise
     Invalid_argument if the two lists have different lengths. Not tail-recursive.
```

```
val rev_map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
List.rev_map2 f l1 l2 gives the  same result as List.rev (List.map2 f l1
l2), but is tail-recursive and more efficient.

val map : ('a -> 'b) -> 'a list -> 'b list
List.map f [a1; ...; an] applies function f to a1, ..., an, and builds the list [f a1; ..
    .; f an] with the results returned by f. Not tail-recursive.

val rev_map : ('a -> 'b) -> 'a list -> 'b list
List.rev_map f l gives the same result as List.rev (List.map f l), but is tail-recursive
    and more efficient.
```

Recherche

Créez les fonctions :

```
(* opération terme à terme sur deux lignes *)
let vecs_op = fun op v1 v2 -> ...

(* opération terme à terme sur deux matrices *)
let mats_op = fun op m1 m2 -> ...
```

puis en pensant à **`fold_left`** :

```
(* op de tous les termes d'une ligne (v1 op v2 op...) *)

let rec som_vec a vec = ...

let rec prod_vec a vec = ...
```

Nous en arrivons au produit de deux matrices...

Recherche

```
(* multiplication d'une ligne par une matrice *)
let ligne_prod_mat a ligne1 m2 =
  let rec local m2 acc = ...

(* multiplication de deux  matrices *)
let prod_mat a m1 m2 =
  let rec local m1 acc = ...
```

7 7 Matrice définie par une fonction de ses numéros de ligne et colonne

Voici une proposition :

```
let mat_fonc  = fun f r c ->
  let rec aux_ligne = fun f i j acc ->
    match j with
    |j when (j=c)  -> rev acc
    |_ -> aux_ligne f i (j+1)  ((f i j)::acc)
  in
  let rec aux_col = fun f i j acc ->
    match i with
    |i when (i=r) -> rev acc
```

```
    |_ -> aux_col f (i+1) j ((aux_ligne f i 0 []::acc)
  in
  aux_col f 0 0 [];;
```

Recherche

Commentez...
Déduisez-en une fonction **unite a n** qui renvoie la matrice unité de taille **n** dans l'anneau **a**.

7.8 Trace

Tout est dans le titre...N'oubliez pas la récursion terminale.

7.9 Matrices élémentaires - Opérations sur les lignes

Il faut bien connaître son cours...Vous construirez les diverses fonctions comme produit de matrices et composées de fonctions.

Recherche

Commencez par créer **elem a i j n** i.e. $E_n^{i,j}$ puis **kelem a i j k n** i.e. $kE_n^{i,j}$.
Ensuite

```
(* matrice de transvection : (Id +k E(ij))M
   Li <- Li+k.Lj *)
let transvec a i j k mat = ...

(* dilatation : * Diag(1 1 1 k 1 1 1...)   Li <- k Li *)
let dilat a i k mat = ...

(* échange de lignes *)
let swap a i j mat =...
```

On aura enfin besoin d'une fonction qui élimine une ligne d'une matrice. Par exemple :

```
# let m4 = [[11;12;13;14];[21;22;23;24];[31;32;33;34];[41;42;43;44]];;
val m4 : int list list =
  [[11; 12; 13; 14]; [21; 22; 23; 24]; [31; 32; 33; 34]; [41; 42; 43; 44]]
# elim entier m4 1;;
- : int list list = [[11; 12; 13; 14]; [31; 32; 33; 34]; [41; 42; 43; 44]]
```

Recherche

Créez la fonction **elim**.

Enfin, il sera utile de disposer d'une fonction qui combine des lignes sous la forme $L_i \leftarrow k_i L_i - k_j L_j$.

Recherche

Créez **(* Li <- kiLi - kj Lj *) let comb a i ki j kj mat =...**

7.10 Concaténation de tableaux

Nous aurons besoin de « concaténer » deux matrices, que ce soit pour déterminer l'inverse ou résoudre un système linéaire par la méthode de la LRÉ.

Recherche

Créez **concat_mat = fun m1 m2** en pensant à utiliser **rev_append** et **rev_map2**.
Créez ensuite une fonction **mat_to_tab a mat** qui « colle » à **mat** la matrice unité de l'anneau **a**.
Créez enfin une fonction **tab_to_mat** qui extrait le demi-tableau de droite final.

7 11 Permutations circulaires

Au moment de chercher des pivots, nous aurons besoin de permuter des lignes. Une possibilité est de mettre systématiquement la ligne contenant le zéro en dernier et de décaler circulairement celles du dessous pour éviter de faire des « *swap* » sur les deux mêmes lignes à chaque fois.
Par exemple, en lançant **permut_circi m 1** on passera de :

$$\begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{pmatrix} \text{ à } \begin{pmatrix} 11 & 12 & 13 & 14 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \\ 21 & 22 & 23 & 24 \end{pmatrix}.$$

Recherche

Déterminez une fonction **permut_circi a mat i** en utilisant si besoin **rev_append**, **rev**, **elim**, **nth**.

7 12 Triangulation de Gauß

On commence par explorer une colonne et trouver un candidat-pivot, c'est-à-dire qu'un cherche le premier élément non nul dans une colonne à partir d'une ligne donnée (repensez à ce qui a été fait sur papier les semaines précédentes).

Recherche

Déterminer **trouve_pivot a mat pos** qui trouve un pivot dans **mat** à partir de la ligne **pos**, c'est-à-dire qu'elle teste $\texttt{mat}_{\texttt{pos,pos}}$: s'il est non nul, on renvoie la matrice inchangée, sinon, on la renvoie après un coup de **permut_circi**.

Une fois le candidat-pivot trouvé, il faut faire le vide en-dessous, c'est-à-dire combiner les lignes pour y faire apparaître des zéros.

Recherche

Déterminez **vide_sous_pivot a mat pos**.

Par exemple :

```
# printm entier m4;;
 |    11     12     13     14  |
 |    21     22     23     24  |
 |    31     32     33     34  |
 |    41     42     43     44  |

# printm entier (vide_sous_pivot entier m4 1);;
 |    11     12     13     14  |
 |    21     22     23     24  |
 |    10      0    -10    -20  |
 |    20      0    -20    -40  |
```

Il ne reste plus qu'à composer ces fonctions pour obtenir la matrice trigonalisée par la méthode de GAUSS.

Recherche

Créez **`tri_gauss a mat`**.

```
# let mex = [[1;3;2;-1];[-1;-2;2;6];[2;4;-3;2];[-2;-7;4;1]];;
val mex : int list list =
  [[1; 3; 2; -1]; [-1; -2; 2; 6]; [2; 4; -3; 2]; [-2; -7; 4; 1]]
# printm entier (tri_gauss entier mex);;
 |     1      3      2     -1  |
 |     0      1      4      5  |
 |     0      0      1     14  |
 |     0      0      0   -164  |
```

Recherche

Peut-on en déduire le déterminant de la matrice ?

7 13 Lé et Lré parla méthode de Gauß-Jordan

Cette fois il faut faire le vide non seulement sous le pivot mais aussi au-dessus :

```
# printm entier (vide_autour_pivot entier m4 1);;
 |   -10      0     10     20  |
 |    21     22     23     24  |
 |    10      0    -10    -20  |
 |    20      0    -20    -40  |
```

Recherche

Créez **`vide_autour_pivot`**.

Il ne reste plus qu'à faire le vide sur toute la matrice en gardant des pivots « généralisés » c'est-à-dire pas forcément égaux à 1 car selon les anneaux dans lesquels on travaille, tous les nombres ne sont pas inversibles.

Recherche

Créez la fonction **`le a mat`** qui renvoie la ℓ-échelonnée pas forcément réduite.

Par exemple :

```
# printm entier (le entier [[11;12;13;14];[21;22;23;24];[31;32;33;34];[41;42;43;44]]);;
 |  -110      0    110    220  |
 |     0    -10    -20    -30  |
 |     0      0      0      0  |
 |     0      0      0      0  |
```

Certaines lignes vont pouvoir être simplifiées. Il faut donc créer une fonction qui « simplifie » une ligen (mais quelle opération se cache derrière cette notion de simplification ?).

```
# simp_vec entier [0;0;3;6;9];;
- : int list = [0; 0; 1; 2; 3]
# simp_vec entier [0;0;3;0;0];;
- : int list = [0; 0; 1; 0; 0]
# simp_vec entier [0;0;3;6;7];;
Exception: Inverse_indefini.
```

Recherche

Créer **simp_vec a vec** puis utilisez **le** pour créer une fonction **lre** qui renvoie, si elle existe, la lré d'une matrice dans un anneau donné.

```
# printm entier ( le entier [[2;5;-2;3];[3;6;3;6];[1;2;-1;2]]);;
 |   -72      0      0   -288  |
 |     0    -36      0     36  |
 |     0      0     12      0  |

# printm entier ( lre entier [[2;5;-2;3];[3;6;3;6];[1;2;-1;2]]);;
 |     1      0      0      4  |
 |     0      1      0     -1  |
 |     0      0      1      0  |
```

7.14 Inverse d'une matrice

Nous avons tout ce qu'il nous faut pour obtenir l'inverse d'une matrice, si elle existe, par la méthode de GAUSS-JORDAN : on colle l'identité à droite, on calcule la lré, on extrait la partie gauche du tableau : ya plus qu'à...

Recherche

Déterminez la fonction **inverse a mat** :

Par exemple :

```
# printm reel (inverse reel [[1.;0.;1.];[0.;1.;1.];[1.;1.;0.]]);;
 |   0.5   -0.5    0.5  |
 |  -0.5    0.5    0.5  |
 |   0.5    0.5   -0.5  |
```

7.15 Corps des rationnels

Si on travaille dans $\mathbb{Z}$, on est vite limité pour inverser des matrices : il n'y a que 1 et -1 qui soit inversibles.
Nous allons donc créer un type **rationnel** pour modéliser les nombres appartenant à $\mathbb{Q}$ sous forme d'un enregistrement puis nous créerons un anneau **rat** pour travailler avecc des matrices de rationnels.
Un rationnel est défini par un numérateur et un dénominateur :

```
type rationnel = {
num : int;
den : int;
};;
```

Par exemple, $\frac{6}{9}$ sera modélisé par :

```
let r = {num = 6; den = 9};;
```

Mais ça se simplifie...Comment ?

Recherche

Créez une fonction **frac a b** qui renvoie le rationnel **{num = a; den= b}** sous forme simplifiée :

```
# frac 6 9;;
- : rationnel = {num = 2; den = 3}
```

Pour simplifier l'entrée de ces nombres, on peut introduire un opérateur infixe :

```
let (//) a b = frac a b ;;
```

Par exemple :

```
# (-6) // (-9);;
- : rationnel = {num = 2; den = 3}
```

Pour créer notre anneau de rationneles, il faut créer les fonctions utiles : somme, produit, etc.

Recherche

Créez **`r_plus`**, **`r_mult`**, **`r_sup`**, **`r_egal`**, **`r_div`**, **`r_sous`**.

Pour la conversion en chaîne, voici comment faire :

```
let r_to_string r =
  let s = (r.num // r.den) in
  if s.num = 0 then "0"
  else if s.den = 1 then (string_of_int s.num)
  else (string_of_int s.num)^"/"^(string_of_int s.den);;
```

Par exemple :

```
# r_to_string (6//9);;
- : string = "2/3"
# r_to_string (8//4);;
- : string = "2"
```

Nous sommes donc prêts :

```
let rat = {
   zero = 0//1;
   un   = 1//1;
   som  = r_plus;
   prod = r_mult;
   sous  = (fun x y -> r_plus  x (r_opp y));
   div = (fun x y -> r_mult x (r_inv y));
   to_string = r_to_string;
   egal = r_egal;
   ordre = r_sup;
  };;
```

Recherche

Construisez une fonction qui convertit un entier en rationnel puis une matrice entière en matrice rationnelle.

Nous pouvons alors travailler plus confortablement :

```
# printm rat (inverse rat (mat_rat_of_int ([[1;0;1];[0;1;1];[1;1;0]])));;
 |   1/2   -1/2    1/2  |
 |  -1/2    1/2    1/2  |
 |   1/2    1/2   -1/2  |
```