informatique

DUT INFO 1 / 2015-2016

Licence Creative Commons

Mis à jour le 8 janvier 2016 à 10:43

Programmation et Algèbre

Guillaume CONNAN

IUT NANTES Université de Nantes

UNIVERSITÉ DE NANTES

CHAPITRE

3 Algèbre générale

Monoïdes

Recherche 3 - 1 Pourcentages

On peut considérer que les pourcentages sont en fait les nombres réels entre 0 et 1 compris. Est-ce que ces nombres munis de l'addition forment un monoïde ? Avec la multiplication ?

Recherche 3 - 2

Un monoïde peut-il avoir plusieurs éléments neutres ?

Recherche 3 - 3 Have you got the power ?

On considère la LCI $\star$ définie sur $\mathbb{Z}$ par $a \star b = a - b$
L'écriture $a^3 = a \star a \star a$ a-t-elle un sens ? Quel est le sens de cette question ?

Recherche 3 - 4 Have you got the power now ?

On considère la LCI $\star$ définie sur $\mathbb{Z}$ par $a \star b = \underbrace{a \times a \times \cdots \times a}_{b \text{ facteurs}}$
L'écriture $a^3 = a \star a \star a$ a-t-elle un sens ?

Recherche 3 - 5 And now ?

On considère la LCI définie sur $\{A, B, C, D\}$ par :

$\star$	A	B	C	D
A	B	C	D	A
B	D	A	B	C
C	A	B	C	D
D	C	D	A	B

L'écriture $A^3 = A \star A \star A$ a-t-elle un sens ?

Recherche 3 - 6

Une définition :
Un élément a de E est dit **régulier à gauche** *(ou* **simplifiable à gauche***) si, et seulement si :*

$$(\forall x)(\forall y)((a \star x = a \star y) \to (x = y))$$

On a une définition similaire de la régularité à droite.
Un élément à la fois régulier à gauche et à droite est dit **régulier**.
Une loi telle que tout élément soit régulier est dite régulière.
Un élément inversible est régulier (vérifiez-le). Un élément régulier est-il inversible ?

Recherche 3 - 7

On considère l'opération définie sur $\mathbb{Z}$ par :

$$a \top b = a + 2b$$

Cette opération est-elle une LCI ? Est-elle commutative ? Associative ? Y a-t-il un élément neutre ? Y a-t-il des éléments simplifiables ? Symétrisables ?
Mêmes questions avec la loi définie sur $\mathbb{Q}$ par $a \dagger b = (a + b)/2$.

Recherche 3 - 8

On considère l'opération $\uparrow$ définie sur $\mathbb{N}_1 = [\![1, +\infty[\![$ par :

$$a \uparrow 1 = a, \quad a \uparrow b = a \times (a \uparrow (b - 1))$$

Cette opération vous rappelle-t-elle quelque chose ? Est-ce une LCI ? Est-elle commutative ? Associative ? etc.
Cherchez un élément régulier à droite mais pas à gauche.

Recherche 3 - 9
En termes de magma, monoïde, que pensez-vous de la soustraction des entiers ? Du produit cartésien des ensembles ? De la composition des relations ? De la multiplication d'un vecteur du plan par un réel ? etc.

Recherche 3 - 10
Donnez des exemples de couples $\langle E, \star \rangle$ avec $\star$ une application définie sur $E \otimes E$ telle que $\langle E, \star \rangle$ ne soit pas un magma.

Recherche 3 - 11
Soit $A = \{a_1, a_2, ..., a_p\}$ un ensemble de caractères. On note A^* l'ensemble des chaînes de caractères construites à partir des caractères de A.

1. A^* muni de la concaténation des chaînes a-t-il une structure de magma associatif ? de monoïde ?
2. Les ensembles suivants munis de la concaténation sont-ils des magmas associatifs ? Des monoïdes ?
   - i. L'ensemble des chaînes de longueur paire ;
   - ii. L'ensemble des chaînes de longueur impaire ;
   - iii. $\{(a_1 a_2)^n \quad | \quad n \in \mathbb{N}\}$;
   - iv. $\{a_1^n a_2^n \quad | \quad n \in \mathbb{N}\}$;
   - v. L'ensemble des chaînes ne contenant que a_1 et a_2, ceux-ci apparaissant un nombre égal de fois.

Recherche 3 - 12
Soit $\mathbb{B}_2 = \{0, 1\}$ l'ensemble des booléens. Est-ce que $\mathbb{B}_2$ muni de NAND est un monoïde ? Muni de NOR ?

Recherche 3 - 13
Soit l'opération $\top$ définie par $x \top y = \sqrt{x^2 + y^2}$. Est-ce que $\langle \mathbb{R}, \top \rangle$ est un magma associatif ? Un monoïde ? Un monoïde simplifiable ? Y a-t-il un élément absorbant ?

Recherche 3 - 14
Soit une fonction $\mathcal{F}(E)$ l'ensemble des fonctions de E dans E.
On dit que f est inversible à gauche s'il existe une fonction g telle que $g \circ f = \mathrm{Id}_E$.
Montrez que f inversible à gauche si, et seulement si, f est injective puis que f est inversible à droite si, et seulement si, f est surjective.

Recherche 3 - 15 Notation polonaise inverse

La notation polonaise inverse ou notation postfixée consiste à placer les opérateurs <u>après</u> les opérandes, en cela il diffère de la notion infixe. Par exemple :

Notation infixe	Notation postfixe
$1 + 2$	$1\ 2\ +$
$10 \div 0.5$	$10\ 0.5\ \div$
2×3	$2\ 3\ \times$
$(1 + 2) \times 3$	$1\ 2\ +\ 3\ \times$
$1 + (2 \times 3)$	$1\ 2\ 3\ \times\ +$
$(-2) - 3$	$2\ -_1\ 3\ -_2$
$-(2 - 3)$	$2\ 3\ -_2\ -_1$

Remarques sur la notation postfixe.

— Pas besoin de parenthèses

— Il faut distinguer le $-$ unaire (noté ici $-_1$) du moins binaire (noté ici $-_2$). Le premier transforme un nombre en son opposé (exemple : -2) le second calcule la différence entre deux termes (exemple : $1 - 3$).

1. Convertir les expressions suivantes en notation polonaise inverse.
   - i. $1 + 2 + 3 + 4 + 5 + 6$
   - ii. $7 + (1 + 3) \times (2 + 3)$
   - iii. $2 \times 3 \times 5 \div 7$
   - iv. $3 + 9 \times (5 \times (7 + 2) + 2)$
2. Convertir en notation usuelle les expressions postfixes suivantes.

i. $2\ 3\ -_1\ -_2$

ii. $1\ 2\ 3\ +\ \times$

iii. $1\ 2\ 3\ +\ -_1\ \times$

iv. $7\ 5\ 3\ 2\ 1\ +\ \times\ \div\ +$

TP 3 - 1

Essayez de déterminer des valeurs « flottantes » a, b et c pour lesquelles $(a + b) + c \neq a + (b + c)$. Expliquez votre raisonnement. Vérifiez avec l'interpréteur Python.

TP 3 - 2

Est-ce que les opérateurs Python `and` et `or` sont commutatifs ?

TP 3 - 3

Explorez le code suivant...

```
1   class C() :
2
3       def  __init__(self, chaine = ' ') :
4           self.car = chaine
5
6       def __add__(self, other):
7           return chr( ord(self.car) + ord(other.car) - 32)
8
9       def __repr__(self) :
10          return self.car
11
12  class Ens() :
13
14      def __init__(self, elements = set([])) :
15          self.elms = elements
16
17      def __add__(self, other) :
18          U = Ens()
19          U.elms = set.union(self.elms, other.elms)
20          return U
21
22      def __repr__(self):
23          return str(self.elms)
24
25  class Nat() :
26
27      def __init__(self, ent = 0) :
28          self.nat = abs(ent)
29
30      def __add__(self, other) :
31          if self.nat >= other.nat :
32              return Nat(self.nat)
33          return Nat(other.nat)
34
35      def __repr__(self):
36          return str(self.nat)
37
38  class Panier() :
39
40      def __init__(self, articles = set([])) :
41          self.contenu = articles
42
43      def __add__(self, other) :
44          return Panier(self.contenu | other.contenu)
45
46      def __repr__(self) :
47          return str(self.contenu)
```

TP 3 - 4 HP15C

Créez une calculatrice qui effectue les calculs arithmétiques à la mode polonaise inversée.
On pourra se contenter d'une version simplifiée : on donne comme argument la chaîne correspondant à l'opération à effectuer sous forme postfixée. La fonction renvoie le résultat de l'opération. Par exemple :

```
In [1]: seq = "2 3 4 + *"

In [2]: npi(seq)
Out[2]: 14
```

On pourra utiliser les fonctions du module `operator`.
Par exemple, `operator.add(2,3)` renvoie 5.
On peut alors créer un dictionnaire des opérations :

```
operations = {
    '+':  operator.add, '-':  operator.sub,
    '*':  operator.mul, '/':  operator.truediv,
    '%':  operator.mod, '**': operator.pow,
    '//': operator.floordiv
}
```

et on obtient l'opération d'addition à partir de la chaîne '+' :

```
In [3]: operations['+']
Out[3]: <function _operator.add>
```

Par exemple :

```
In [1]: npi("3 2 7 9 - * 3 + 2 / +")
Out[1]: 2.5

In [2]: npi("3 2 7 9 - * 3 + 2 /")
  File "<string>", line unknown
SyntaxError: Expression non valide
```

Vous pouvez ensuite proposer une calculatrice plus « interactive » :

```
In [1]: hp_inter()
->  |
rentrer un nombre ou un opérateur: 1
-> 1 |
rentrer un nombre ou un opérateur: 2
-> 2, 1 |
rentrer un nombre ou un opérateur: 3
-> 3, 2, 1 |
rentrer un nombre ou un opérateur: 4
-> 4, 3, 2, 1 |
rentrer un nombre ou un opérateur: '+'
-> 7, 2, 1 |
rentrer un nombre ou un opérateur: '-'
-> -5, 1 |
rentrer un nombre ou un opérateur: '+'
-> -4 |
rentrer un nombre ou un opérateur: 3
-> 3, -4 |
rentrer un nombre ou un opérateur: '*'
-> -12 |
rentrer un nombre ou un opérateur: 'fin'
```

TP 3 - 5

Définissez une fonction `big(op, neutre)` qui prend un opérateur associatif et son neutre et renvoie une fonction qui prend une collection (`tuple`, `list`, `set`,...) et renvoie la réduction de la collection par l'opération :
`neutre op x1 op x2 op...op xn`

TP 3 - 6

Les fonctions `map` et `filter` de Python sont en fait des cas particuliers de `reduce`. Définissez-les en utilisant `reduce` au lieu d'itération ou de récursion.

TP 3 - 7

Définissez une version de `reduce` qui omet le troisième argument (le neutre).
Cette fonction sera utilse pouur travailler sur des magmas associatifs qui n'ont pas de neutre.

Groupes

Recherche 3 - 16

1. $\mathbb{B}_2$ muni des opérations suivantes est-il un groupe :
   i. $\wedge$? ii. $\vee$? iii. $\oplus$?
2. Résoudre dans $\mathbb{B}_2$ l'équation $a \oplus x = b$.

Recherche 3 - 17

Soit $\langle G, \star \rangle$ un groupe et a et b deux éléments de G. Exprimez l'inverse de $a \star b$ en fonction des inverses de a et de b.

Recherche 3 - 18

Les tables ci-dessous définies sur $E = \{a, b, c\}$ définissent-elles des magmas associatifs ? des monoïdes ?

$\star$	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

$\perp$	a	b	c
a	a	c	c
b	b	c	a
c	c	a	a

Notons $A = \{a, b, c\}$. Montrez que $\langle A, \star \rangle$ a une structure de groupe. Comparer ce groupe avec

— $\langle \mathbb{Z}/3\mathbb{Z}, \overline{+} \rangle$
— $\left\langle \{1, e^{2i\pi/3}, e^{-2i\pi/3}\}, \times \right\rangle$.

Recherche 3 - 19 Isomorphisme

Montrez que $\langle \mathbb{F}_2, + \rangle$ et $\langle \mathbb{Z}/4\mathbb{Z}^\star, \cdot \rangle$ sont isomorphes.
Est-ce que $\langle \mathbb{Z}/4\mathbb{Z}, + \rangle$ et $\langle \mathbb{Z}/5\mathbb{Z}^\star, \cdot \rangle$ sont isomorphes ?
Est-ce que $\langle \mathbb{Z}/4\mathbb{Z}, + \rangle$ et $\langle \mathbb{Z}/8\mathbb{Z}^\star, \cdot \rangle$ sont isomorphes ?
Est-ce que $\langle \mathbb{F}_2 \otimes \mathbb{F}_2, + \rangle$ et $\langle \mathbb{Z}/8\mathbb{Z}^\star, \cdot \rangle$ sont isomorphes ?

Recherche 3 - 20 Déplacement à partir du pavé numérique

À chaque touche du pavé numérique, on fait correspondre un déplacement élémentaire d'un point sur l'écran : 6 pour un déplacement unitaire à droite, 2 pour un déplacement unitaire vers le bas, etc.
On note D l'ensemble de ces dix déplacements et $D^\star$ l'ensemble des séquences de déplacements.
Associer à $D^\star$ une loi pour en faire un monoïde. Montrez que c'est un groupe abélien.
Que se passe-t-il si au lieu de déplacements on considère la trace laissée par ces déplacements ?

Recherche 3 - 21 Caractérisation d'un sous-groupe

La programmation, c'est l'efficacité... La définition d'un sous-groupe n'est pas efficace : pouvez-vous en donner une caractérisation plus synthétique en trouvant des propriétés minimales à vérifier ?

Recherche 3 - 22

Est-ce que l'ensemble $\left\{x+y\sqrt{2} \mid (x,y) \in \mathbb{Q}^2\right\}$ muni de la multiplication des réels a une structure de groupe ?

Recherche 3 - 23 Chiffrement par blocs

Commençons par une définition :

Définition 3 - 1

Cryptosystème (ou système de chiffrement ou chiffre)
Un cryptosystème est un quintuplet $\langle \mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ tel que :

- L'ensemble $\mathcal{P}$ est l'espace des messages en clair (*plaintext* en anglais). En général, c'est l'ensemble des chaînes de bits.
- L'ensemble $\mathcal{C}$ est l'espace des messages chiffrés (*cyphertext* en anglais). En général, c'est aussi l'ensemble des chaînes de bits.
- L'ensemble $\mathcal{K}$ est l'espace des clés (*key*) ;
- $\mathcal{E}$ est la famille des fonctions de chiffrement, qui vont de $\mathcal{P}$ dans $\mathcal{C}$;
- $\mathcal{D}$ est la famille des fonctions de déchiffrement, qui vont de $\mathcal{C}$ dans $\mathcal{P}$;

Pour chaque élément $e \in \mathcal{K}$, il existe un élément $d \in \mathcal{K}$ tel que, pour tout message clair m de $\mathcal{P}$, il existe $D_d \in \mathcal{D}$ et $E_e \in \mathcal{E}$ telles que $D_d(E_e(m)) = m$. Les fonctions D_d et E_e sont des applications injectives. Il est entendu que d doit rester secret...

1. On travaille sur des mots formés des 26 minuscules non accentuées de notre alphabet latin et d'une espace qui sont modélisés par les entiers de $E = \{0, 1, 2, \ldots, 26\}$.

   Dans ce contexte, on remplace une « lettre » x par $kx[27]$ avec $k \in E$. Définit-on ainsi un cryptosystème ?

2. Un cryptosystème est un *chiffrement par blocs* si $\mathcal{P} = \mathcal{C} = \{0,1\}^n$. Est-ce que le chiffrement ROT-13 est un chiffrement par blocs ?

3. Pourquoi les fonctions de chiffrement d'un chiffrement par blocs sont des permutations ?

Depuis 2001, le chiffrement par blocs « officiel » est l'AES qui opère par blocs de 128 bits. Une présentation de l'algorithme de chiffrement associé, Rijndael, et de sa résistance aux attaques est présentée ici :

http://www.cryptis.fr/assets/files/Canteaut-25ans-Cryptis.pdf.

Recherche 3 - 24 Mode ECB

ECB : *Electronic CodeBook*. C'est le plus simple...et le plus vulnérable des modes de chiffrement par blocs.

Voyons sur un exemple. On considère une chaîne de bits quelconque que l'on découpe en blocs de longueur fixe, par exemple 7 (plus pratique pour ensuite utiliser l'ASCII : pourquoi ?). On rajoute éventuellement des zéros en bout de chaîne.

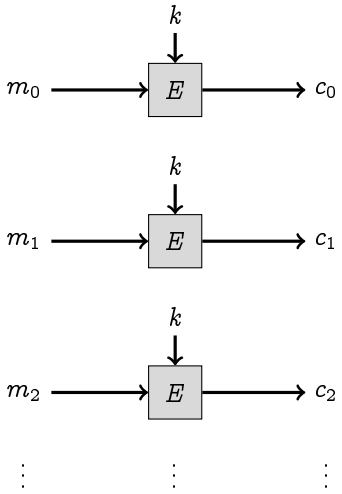
Considérons $m = 1001001111011$. On rajoute un 0 en bout de chaîne :

$$m' = 1001001\ 1110110 = \beta_1\beta_2$$

avec $\beta_1 = 1001001$ et $\beta_2 = 1110110$.

On choisit une clé de chiffrement dans S_7 car $\mathcal{K} = S_7$: $\pi = \begin{pmatrix} 2 & 0 & 3 & 1 & 5 & 4 & 6 \end{pmatrix}$

Alors $E_\pi(\beta_1) = 0110001$ et $E_\pi(\beta_2) = 1101110$.

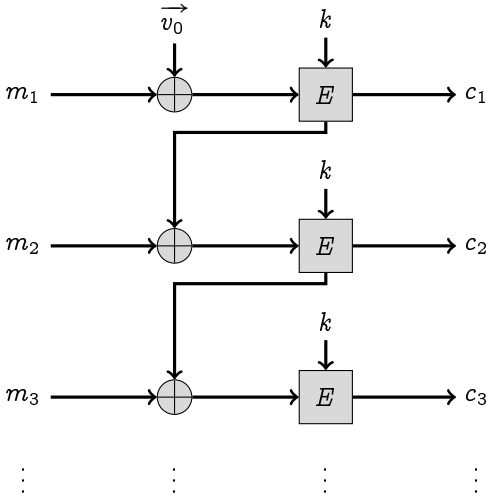


1. Que se passe-t-il si un bloc chiffré est mal transmis ou perdu ?
2. Quelle est la principale faiblesse du mode ECB ?
3. Chiffrez « maman » avec ce cryptosystème en transformant la chaine de lettres en chaînes de bits constituée des codes ASCII de ses caractères. Le tableau ASCII standard est-il satisfaisant ? Comment y remédier ?
4. Quelle est la fonction de déchiffrement associée à l'exemple précédent ?
5. Déchiffrez 10110111001111 sachant que la clé de chiffrement est $k = \begin{pmatrix} 1 & 2 & 0 & 6 & 5 & 4 & 3 \end{pmatrix}$
6. Déchiffrez « papa » sachant que c'est le cryptosystème du début qui a été employé.

Recherche 3 - 25 Mode CBC

CBC : *Cipher-Block Chaining*. Ce mode a été inventé par IBM en 1976. Pour éviter la faiblesse de l'ECB, les blocs sont maintenant chaînés : chaque bloc en clair est « XORé » ou « OUEXé » avec le bloc crypté précédent avant d'être crypté lui-même.
Pour le premier bloc, on utilise un *vecteur d'initialisation* public. Avec les notations habituelles, on a donc

$$c_i = E_k(m_i \oplus c_{i-1}), \quad c_0 = \overrightarrow{v_0}$$



1. Quel est le lien entre OUEX et $\mathbb{F}_2$?
2. Cryptez l' exemple traité avec ECB en prenant $\overrightarrow{v_0} = 1010101$.
3. Faites de même avec « Maman ».
4. Donnez une formule de déchiffrement. Déchiffrer le cryptogramme 101101110011111 sachant que la clé est $k = \begin{pmatrix} 1 & 0 & 2 \end{pmatrix}$ et que $\overrightarrow{v_0} = 001$.
5. Déchiffrez « Papa » sachant que c'est le cryptosystème de la question 2. qui a été employé.

Ce mode est efficace mais nécessite l'utilisation de fonctions de chiffrement et de déchiffrement différentes ce qui peut ralentir la procédure.

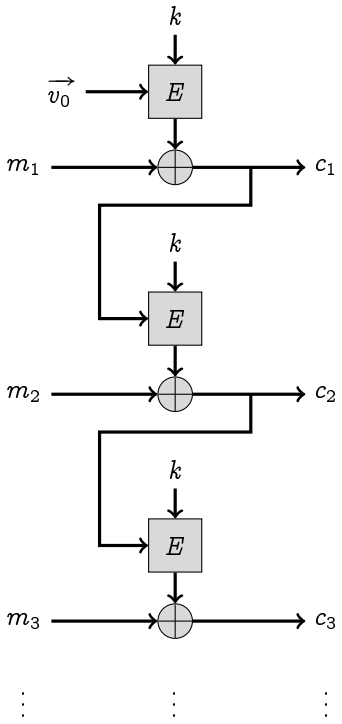
Recherche 3 - 26 Mode CFB

CFB : *Cipher-FeedBack*. C'est un mode dérivé de CFB qui est utilisé par OpenPGP, format pour l'échange sécurisé de données (paiements sécurisés par exemple) largement utilisé actuellement mais est susceptible d'être attaqué, même s'il est très difficile de mettre en œuvre concrètement cette attaque :

http://www.cert-ist.com/fra/ressources/Publications_ArticlesBulletins/Autres/FailledanslechiffrementCFBdOpenPGP/

Le mode CFB utilise un registre de décalage de taille r inférieure à celle de la clé.
On a besoin d'un vecteur d'initialisation $\overrightarrow{v_0}$. Le message clair est découpé en blocs de longueur r. Ensuite on procède comme suit, sachant que les m_j sont les blocs en clair de longueur r :

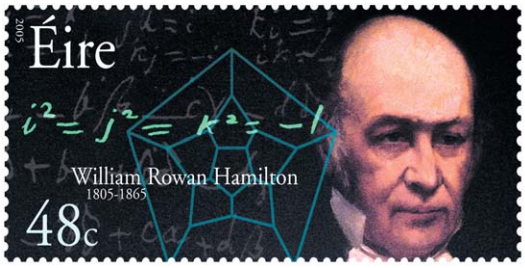
— $t_0 = \overrightarrow{v_0}$;
— $s_j = E_k(t_j)$;
— g_j est la chaîne constituée des r bits les plus à gauche de s_j. Quelle est l'opération arithmétique associée ?
— $c_j = m_j \oplus g_j$;
— $t_j = (2^r t_{j-1} + c_{j-1}) \bmod 2^n$ (attention ! Il faut travailler en base 2).



Le déchiffrement fonctionne de manière identique en échangeant les rôles de m_j et c_j à la quatrième étape : démontrez-le !

1. Chiffrez « *Papa is cool* » sachant que la clé est $k = \begin{pmatrix} 3 & 0 & 2 & 1 \end{pmatrix}$, que $\overrightarrow{v_0} = 0101$ et que $r = 3$.
2. Déchiffrer le cryptogramme 101101110011111 sachant que la clé est $k = \begin{pmatrix} 1 & 0 & 2 \end{pmatrix}$, que $r = 2$ et que $\overrightarrow{v_0} = 001$.

Recherche 3 - 27 Hamilton et les quaternions



Après avoir travaillé sur les nombres complexes comme nous allons le voir bientôt, Sir William HAMILTON (il a été anobli en 1835) voulut étendre ses travaux à la dimension 3. Il n'arrivait cependant pas à imaginer une multiplication qui puisse satisfaire des critères intéressants, notamment la conservation des normes.
Un jour qu'il se promenait avec son épouse dans un parc dublinois, il eut soudain un éclair qu'il s'empressa de graver sur la pierre du pont le plus proche :

$$i^2 = j^2 = k^2 = ijk = -1$$

Remplissez alors la table de multiplication suivante :

×	1	i	j	k
1				
i				
j				
k				

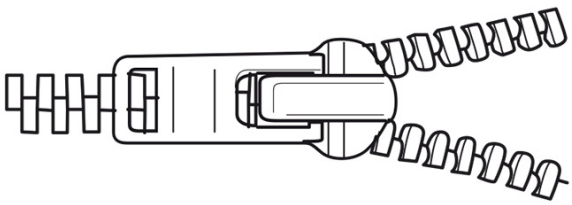
HAMILTON définit alors un *quaternion* par la donnée de quatre réels a, b, c et d car un quaternion peut s'écrire de manière unique comme combinaison des quatre quaternions élémentaires :

$$q = a1 + bi + cj + dk$$

Le nombre a est la partie scalaire et le triplet (b, c, d) est la partie vectorielle (c'est HAMILTON qui a en effet introduit le terme de *vecteur* en mathématique).
Quelles propriétés possède ou ne possède pas la multiplication des quaternions ?
Il y a énormément de choses à dire sur les quaternions mais nous manquons un peu de recul mathématique... Point de vue informatique, il faut cependant noter qu'ils sont largement utilisés en infographie, en traitement du signal et dans tout domaine utilisant des rotations et entraînant de nombreux calculs.
En physique, ces calculs sont traditionnellement traités par du calcul matriciel mais nous verrons bientôt que celui-ci induit de nombreuses erreurs d'arrondis qui peuvent être atténuées en passant par les quaternions.
Un site à visiter : http://www.alcys.com/. Les quaternions apparaissent sous vos yeux ébaubis....

TP 3 - 8

Déterminez une fonction **braguette(v1, v2)** qui renvoie la liste des couples formés par les éléments de deux listes pris dans l'ordre.



Si les listes sont de longueurs différentes, on s'arrête à la plus petite.

```
In [38]: braguette([1, 2, 3, 4],['a', 'b', 'c'])
Out[38]: [(1, 'a'), (2, 'b'), (3, 'c')]
```

Déduisez-en une fonction **prod_scal(v1, v2)** qui calcule le produit scalaire de deux listes avec un petit **reduce**.

```
In [39]: prod_scal([1,2,3],[4,5,6])
Out[39]: 32
```

En fait, on aurait pu gagner du temps en créant une fonction **braguette_avec(xs, ys, op)** qui ferme la braguette en effectuant une opération sur chaque dent.

```
In [44]: braguette_avec("abcd", "ABCD", lambda x, y: x + y)
Out[44]: ['aA', 'bB', 'cC', 'dD']
```

Comment un simple **braguette_avec** va alors nous permettre d'implémenter l'algorithme d'Euclide étendu en trois lignes ?

Rappel :

k	u_k	v_k	r_k	q_k	
0	1	0	19	/	L_0
1	0	1	15	1	L_1
2	1	−1	4	3	$L_2 \leftarrow L_0 - 1 \times L_1$
3	−3	4	3	1	$L_3 \leftarrow L_1 - 3 \times L_2$
4	4	−5	1	3	$L_4 \leftarrow L_2 - 1 \times L_3$
5			0		$L_5 \leftarrow L_3 - 3 \times L_4$

Donnez ensuite une autre version de cet algorithme avec une boucle while :

```
1 def bezout(a,b) :
2     U, V = ????, ????
3     while ???? :
4         U, V = ????
5     return ?
```

C'est celle que nous utiliserons pour construire notre classe de calcul modulaire.

TP 3 - 9 Permutations

Définition 3 - 2 — **Permutation**
Soit E un ensemble. Une permutation de E est une application bijective de E sur E. On note $S(E)$ l'ensemble des permutations de E.

Généralement, on note les permutations sous forme d'une matrice où la première ligne correspond aux éléments de E et où la deuxième ligne correspond aux images des éléments de E.
Par exemple :

$$\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 2 & 0 & 1 & 4 & 3 \end{pmatrix}$$

est une permutation de $E = \{0, 1, 2, 3, 4\}$.
On peut la résumer à la donnée de $[2, 0, 1, 4, 3]$ ou la donnée de la fonction π en extension.

Définition 3 - 3 — Soit $n \in \mathbb{N}$, alors S_n désigne l'ensemble des permutations de $\{0, 1, 2, 3, \ldots, n-1\}$.

Théorème 3 - 1 — **Groupe des permutations**
L'ensemble S_n muni de la loi $\circ$ de composition des applications a une structure de groupe.

Théorème 3 - 2 — **Nombre de permutations**
Le groupe S_n est d'ordre $n!$.

1. Faites le lien entre la composition des applications et le *pipe* (« paillepeu ») des commande système. Écrivez $f \circ g(x)$ en notations sytème.
2. Soit par exemple $\pi' = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \end{pmatrix}$
   Déterminez $\pi \circ \pi'$, $\pi' \circ \pi$, $\pi \circ \pi$.
3. Quelle peut être la réciproque d'une permutation pour la composition ?

4. Voici quelques éléments sur les fonctions `zip` et `sorted` :

```
In [39]: sorted([1,6,5,4])
Out[39]: [1, 4, 5, 6]

In [40]: sorted([(1,2),(4,1),(3,2),(2,0)])
Out[40]: [(1, 2), (2, 0), (3, 2), (4, 1)]

In [41]: zip([1,3,2],['a','b','c'])
Out[41]: <zip at 0x7f82a884f488>

In [42]: list( zip([1,3,2],['a','b','c']) )
Out[42]: [(1, 'a'), (3, 'b'), (2, 'c')]

In [43]: sorted( list( zip([1,3,2],['a','b','c']) ) )
Out[43]: [(1, 'a'), (2, 'c'), (3, 'b')]
```

Comment dit-on braguette en anglais ?

Alors que dire de :

```
class Perm :

    def __init__(self, images = []) :
        self.images = images

    def __repr__(self) :
        return (' ').join( [str(x) for x in self.images] )

    def __neg__(self) :
        im = self.images
        n = len(im)
        return Perm( map(lambda couple : couple[1], sorted( zip(im, range(n)) ) ) )
    def __add__(self, other) :
        """ fabrique la composée self o other """
        return Perm( ??????? )

    def permute(self,xs) :
        """ si xs = [xs[0], xs[1], xs[2],...], p.permute(xs) = [xs[p(0)], xs[p(1)], xs[p(2)],...] """
        return ??????
```

qui permet d'obtenir :

```
In [47]: p = Perm([0,2,3,1])

In [48]: p
Out[48]: 0 2 3 1

In [49]: -p
Out[49]: 0 3 1 2
```

5. Une *permutation circulaire gauche* « décale » les éléments d'un nombre fixé de « rangs ». Par exemple, voici une permutation circulaire gauche :

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 3 & 4 & 0 & 1 & 2 \end{pmatrix}$$

Décrire $\pi(k)$ dans le cas d'une permutation circulaire de bits de i rangs vers la gauche.

On définit de même des permutations circulaires droite.

Combien y a-t-il de permutations circulaires dans S_n ?

Écrire une méthode **circulaire(self, rangs)** qui renvoie `self` permutée de **rangs** rangs circulairement.

```
def circulaire(self, rangs) :
        """ permutation circulaire de rangs rangs """
        im = self.images
        n = len(im)
        return Perm( ?????? )
```

TP 3 - 10 Mode ECB : la programmation

Il reste à programmer tout ça...
On aura d'abord besoin de la fonction `reduce` de la bibliothèque **functools**.
Complétez alors le squelette suivant :

```
def decoupe_en_blocs(n, bs) :
    """découpe une liste en liste de blocs de longueur n"""
    ???????????

def ecb(cle, bs) :
    """renvoie la liste des permutations des blocs contenus dans une liste"""
    ????

def string2bin(s, n) :
    """ transforme un caractère en sa représentation sur n bits sous forme de liste"""
    ?????

def bin2string(bs) :
    """ transforme une liste de bits en l'entier décimal correspondant"""
    ????

def cats(xs) :
    """concatène une liste de listes ou de chaînes"""
    ????

def code_bloc_string(code_bloc, cle, n_bits, chaine) :
    """applique un chiffrement par bloc sur une chaîne en donnant le nom du
       chiffrement, sa clé et le nb de bits sur lequel les caractères sont codés """
    ?????
```

Par exemple :

```
In [5]: bs = [1,0,0,1,0,0,1,1,1,1,0,1,1]

In [6]: decoupe_en_blocs(7, bs)
Out[8]: [[1, 0, 0, 1, 0, 0, 1], [1, 1, 1, 0, 1, 1, 0]]

In [9]: ecb(Perm([2,0,3,1,5,4,6]), bs)
Out[9]: [[0, 1, 1, 0, 0, 0, 1], [1, 1, 0, 1, 1, 1, 0]]

In [10]: string2bin('(', 7)
Out[12]: [0, 1, 0, 1, 0, 0, 0]

In [13]: bin2string([0, 1, 0, 1, 0, 0, 0])
Out[20]: 40

In [21]: cats(['a', 'u', ' ', 's', 'e', 'c', 'o', 'u', 'r', 's'])
Out[21]: 'au secours'

In [22]: cats([[0,1], [2,3]])
Out[22]: [0, 1, 2, 3]

In [23]: code_bloc_string(ecb, Perm([2,0,3,1,5,4,6]), 7, 'Help')
Out[23]: '0+:h'
```

Corps

Recherche 3 - 28 C'est fini

Soit $E = \{1, 2, 3, 5, 6, 10, 15, 30\}$.
Pour tout couple $\langle a, b\rangle$ d'éléments de E, on note $a \wedge b$ le PGCD de a et b et $a \vee b$ le PPCM de a et b.
Démontrez que $(a \vee b)(a \wedge b) = ab$. Quelle est la structure de $\langle E, \wedge, \vee\rangle$? De $\langle E, \vee, \wedge\rangle$?

Recherche 3 - 29 Hamilton et les isomorphismes

Le mathématicien irlandais William Rowan HAMILTON (1805-1865) publie en 1837 *Theory of Conjugate Functions, or Algebraic Couples* : (http://www.maths.tcd.ie/pub/HistMath/People/Hamilton/PureTime/PureTime.pdf).
En voici un extrait :

> *On the Addition, Substraction, Multiplication, and Division, of Number-Couples, as combined with each other.*
> *6. Proceeding to operations upon number-couples, considered in combination with each other, it is easy now to see the reasonableness of the following definitions, and even their necessity, if we would preserve in the simplest way, the analogy of the theory of couples to the theory of singles :*
>
> $$\langle b_1, b_2\rangle + \langle a_1, a_2\rangle = \langle b_1 + a_1, b_2 + a_2\rangle; \tag{52.}$$
>
> $$\langle b_1, b_2\rangle - \langle a_1, a_2\rangle = \langle b_1 - a_1, b_2 - a_2\rangle; \tag{53.}$$
>
> $$\langle b_1, b_2\rangle\langle a_1, a_2\rangle = \langle b_1, b_2\rangle \cdot \langle a_1, a_2\rangle = \langle b_1 a_1 - b_2 a_2, b_2 a_1 + b_1 a_2\rangle; \tag{54.}$$
>
> $$\frac{\langle b_1, b_2\rangle}{\langle a_1, a_2\rangle} = \left\langle \frac{b_1 a_1 + b_2 a_2}{a_1^2 + a_2^2}, \frac{b_2 a_1 - b_1 a_2}{a_1^2 + a_2^2} \right\rangle. \tag{55.}$$
>
> *Were these definitions even altogether arbitrary, they would at least not contradict each other, nor the earlier principles of Algebra, and it would be possible to draw legitimate conclusions, by rigorous mathematical reasoning, from premises thus arbitrarily assumed : but the persons who have read with attention the foregoing remarks of this theory, and have compared them with the Preliminary Essay, will see that these definitions are really not arbitrarily chosen, and that though others might have been assumed, no others would be equally proper.*
> *With these definitions, addition and subtraction of number-couples are mutually inverse operations, and so are multiplication and division ; and we have the relations,*
>
> $$\langle b_1, b_2\rangle + \langle a_1, a_2\rangle = \langle a_1, a_2\rangle + \langle b_1, b_2\rangle, \tag{56.}$$
>
> $$\langle b_1, b_2\rangle \cdot \langle a_1, a_2\rangle = \langle a_1, a_2\rangle \cdot \langle b_1, b_2\rangle, \tag{57.}$$
>
> $$\langle b_1, b_2\rangle(\langle a_1', a_2'\rangle + \langle a_1, a_2\rangle) = \langle b_1, b_2\rangle\langle a_1', a_2'\rangle + \langle b_1, b_2\rangle\langle a_1, a_2\rangle : \tag{58.}$$
>
> *we may, therefore, extend to number-couples all those results respecting numbers, which have been deduced from principles corresponding to these last relations. For example,*
> $(\langle b_1, b_2\rangle + \langle a_1, a_2\rangle) \cdot (\langle b_1, b_2\rangle + \langle a_1, a_2\rangle) = \langle b_1, b_2\rangle\langle b_1, b_2\rangle + 2\langle b_1, b_2\rangle\langle a_1, a_2\rangle$
> $+ \langle a_1, a_2\rangle\langle a_1, a_2\rangle,$ (59.)
> *in which*
>
> $$2\langle b_1, b_2\rangle\langle a_1, a_2\rangle = \langle 2, 0\rangle\langle b_1, b_2\rangle\langle a_1, a_2\rangle = \langle b_1, b_2\rangle\langle a_1, a_2\rangle + \langle b_1, b_2\rangle\langle a_1, a_2\rangle; \tag{60.}$$
>
> *for, in general, we may mix the signs of numbers with those of number-couples, if we consider every single number a as equivalent to a pure primary number-couple,*
>
> $$a = \langle a, 0\rangle. \tag{61.}$$
>
> *When the pure primary couple $\langle 1, 0\rangle$ is thus considered as equivalent to the number 1, it may be called, for shortness, the primary unit ; and the pure secondary couple $\langle 0, 1\rangle$ may be called in like manner the secondary unit.*
> *We may also agree to write, by analogy to notations already explained,*
>
> $$\begin{aligned} \langle 0, 0\rangle + \langle a_1, a_2\rangle &= +\langle a_1, a_2\rangle, \\ \langle 0, 0\rangle - \langle a_1, a_2\rangle &= -\langle a_1, a_2\rangle; \end{aligned} \tag{62.}$$
>
> *and then $+\langle a_1, a_2\rangle$ will be another symbol for the number-couple $\langle a_1, a_2\rangle$ itself, and $-\langle a_1, a_2\rangle$ will be a symbol for the opposite number-couple $\langle -a_1, -a_2\rangle$. The reciprocal of a number-couple $\langle a_1, a_2\rangle$ is this other number-couple,*
>
> $$\frac{1}{\langle a_1, a_2\rangle} = \frac{\langle 1, 0\rangle}{\langle a_1, a_2\rangle} = \left\langle \frac{a_1}{a_1^2 + a_2^2}, \frac{-a_2}{a_1^2 + a_2^2} \right\rangle = \frac{\langle a_1, -a_2\rangle}{a_1^2 + a_2^2}. \tag{63.}$$

It need scarcely be mentioned that the insertion of the sign of coincidence = between any two number-couples implies that those two couples coincide, number with number, primary with primary, and secondary with secondary ; so that an equation between number-couples is equivalent to a couple of equations between numbers.

Vous ferez de ce texte un commentaire composé puis détaillerez l'isomorphisme entre $\mathbb{R}^2$ et $\mathbb{C}$ évoqué en cours.

TP 3 - 11 Rationnels - Relations d'équivalences

Qu'est-ce qu'un rationnel ? *Ratio* est un mot latin, qui désignait un calcul mais qui a donné en français le mot *raison*. Le mot *ratio* est passé en anglais pour désigner un indice en comptabilité : le rapport entre deux quantités de même nature. Ce mot a été importé de l'anglais de spécialité par les financiers.
En mathématique, un *nombre rationnel* est « **construit** » à partir d'un couple d'entiers relatifs, le deuxième étant non nul. Le premier est appelé *numérateur*, le second *dénominateur*.
Oui mais comment s'effectue cette « construction » ? On a une idée intuitive de la réponse : on veut qu'un *ration*nel représente une proportion. Par exemple, si Daniel a 5 billes bleues parmi ses 8 billes et si Fabienne a 10 billes bleues parmi ses 16 billes, les deux enfants ont la même proportion de billes bleues. On voudrait alors que cette proportion soit désignée par le même nombre rationnel.
On voudrait donc mettre en *relation* tous les couples ⟨ nombre de billes bleues, nombre total de billes ⟩ qui « représentent » la même proportion de billes bleues.
On va donc modéliser cela en considérant la relation :

$$\Big(\forall\langle a,b\rangle \in \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})\Big)\Big(\forall\langle c,d\rangle \in \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})\Big)\Big(\langle a,b\rangle\,\mathcal{R}\,\langle c,d\rangle \leftrightarrow ad=bc\Big)$$

Deux couples en relation vont donc représenter des fractions *égales* : c'est une *relation d'équivalence* car elle est :

- réflexive : $\forall x.x\mathcal{R}x$
- symétrique : $\forall x.\forall y.x\mathcal{R}y \Longrightarrow y\mathcal{R}x$
- transitive : $\forall x.\forall y.\forall z.x\mathcal{R}y \wedge y\mathcal{R}z \Longrightarrow x\mathcal{R}z$

On va alors pouvoir définir cette égalité dans une classe Python.
Une proportion est alors une classe modulo cette relation d'équivalence.
On désigne par $\mathbb{Q}$ l'ensemble quotient de $\mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})$ modulo $\mathcal{R}$.
Cette notation a été introduite par le mathématicien italien Guiseppe Peano en 1895 car c'est l'initiale du mot italien « *quoziente* ».

$$\mathbb{Q} = \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})/\mathcal{R} = \left\{\left[\langle n,d\rangle\right]_{\mathcal{R}} \mid \langle n,d\rangle \in \mathbb{Z}\otimes(\mathbb{Z}\setminus\{0\})\right\}$$

Un nombre rationnel est donc une classe d'équivalence, c'est-à-dire un ensemble !!!
Pour définir une nouvelle égalité sur Python, il faut vérifier qu'on peut définir une relation d'équivalence.

```
class Ratio():

    def __init__(self, num, den) :
        assert den != 0, "Rationnel non défini ! (dénominateur nul)"
        self.num = num
        self.den = den

    def __eq__(self, other) :
        return self.num * other.den == self.den * other.num
```

On peut alors utiliser l'opérateur **==** pour comparer des rationnels :

```
In [5]: r1 = Ratio(1,3)
In [6]: r2 = Ratio(4,12)
In [7]: r1 == r2
Out[7]: True
In [8]: r3 = Ratio(2,0)
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-8-73704c7d78e0> in <module>()
----> 1 r3 = Ratio(2,0)
~/PROG/PYTHON/IntroAlg/monoides.py in __init__(self, num, den)
AssertionError: Rationnel non défini ! (dénominateur nul)
```

On peut rajouter une petite méthode pour représenter les rationnels de manière sympathique :

```
    def __repr__(self) :
        return str(self.num) + '/' + str(self.den)
```

Par exemple :

```
In [13]: Ratio(2,4)
Out[13]: 2/4
```

Mais on peut faire mieux pour avoir le représentant irréductible : comment ?

```
In [17]: Ratio(3,6)
Out[17]: 1/2
```

Définissez de même les opérations usuelles sur les corps :

```
    def __add__(self, other) :

    def __neg__(self) :

    def __sub__(self, other) :

    def __mul__(self, other) :

    def __div__(self, other) :
```

On rajoutera une méthode mystérieuse `__hash__(self)` qui permettra de faire par exemple des ensembles de rationnels. Vous comprendrez pourquoi en seconde année dans le cours d'algo avancée.

```
def __hash__(self) :
        return hash(self.num)^hash(self.den)
```

Imaginez ensuite une classe `Complex` pour les nombres complexes.

TP 3 - 12 Calcul modulaire

Créez une classe Modulo afin de calculer dans $\mathbb{Z}/n\mathbb{Z}$.

```
In [140]: Modulo(6,3)
Out[140]: 3%6

In [141]: Modulo(6,3) + Modulo(6,26)
Out[141]: 5%6

In [142]: Modulo(6,3) * Modulo(6,26)
Out[142]: 0%6

In [143]: Modulo(6,3).est_inversible()
Out[143]: False

In [144]: Modulo(6,5).est_inversible()
Out[144]: True

In [145]: Modulo(6,13) == Modulo(6,25)
Out[145]: True
```

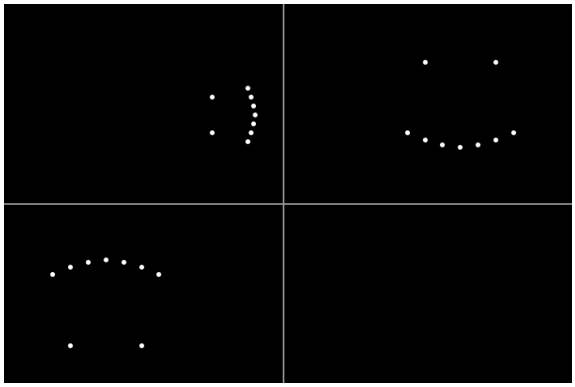
TP 3 - 13

Récupérez sur Madoc le fichier **plotting.py**
Créez dans le même répertoire un fichier **complex.py** :

```
from plotting import plot

S = {2+2j, 3+2j, 1.75+1j, 2+0.9j, 2.25+0.83j, 2.5+0.79j, 2.75+0.83j, 3+0.9j, 3.25+1j}
```

Puis lancez **plot(S)**
Il s'agit ensuite de faire tourner S, de translater S, de réduire ou agrandir S, de combiner les transformations sur une seule figure : comment ?



TP 3 - 14 Corps

Nous allons avoir besoin de plusieurs coprs pour nos expériences sur les vecteurs.

Vous allez créer (en reprenant le travail déjà fait auparanvant), une classe **R** des réels, une classe **Cplx** des complexes, une classe **Ratio** des rationnels, une classe **Mod** des entiers modulo un entier qui auront toutes le squelette suivant :

```
class X:
    def __init__(self,val) :

    def __hash__(self) :
        return hash(val)

    def zero(self) :
        """le neutre de l'addition"""

    def un(self) :
        """le neutre de la multiplication"""

    def __eq__(self, other) :

    def __str__(self) :

    def __repr__(self) :

    def __add__(self, other) :

    def __mul__(self, other) :

    def __neg__(self) :

    def __sub__(self, other) :

    def __truediv__(self, other) :

    def inv(self) :
```

CHAPITRE

4

Algèbre linéaire

Vecteurs

Recherche 4 - 1 Sous-espace vectoriel engendré par une famille de vecteurs

Combien y a-t-il de vecteurs dans $\text{Vect}(\langle 1,1\rangle, \langle 0,1\rangle)$ considéré comme $\mathbb{F}_2$-espace vectoriel ? comme $\mathbb{R}$-espace vectoriel ?

Recherche 4 - 2 Famille génératrice

On travaille dans le $\mathbb{F}_2$-espace vectoriel $\mathbb{F}_2^5$. Quels sont tous les vecteurs de $\text{Vect}(\langle 1,1,1,0,0\rangle, \langle 0,1,1,1,0\rangle, \langle 0,0,1,1,1\rangle)$?

Définition 4 - 1

Famille génératrice
Soit $\mathcal{V}$ un ensemble de vecteurs. Soit $(v_0, v_1, v_2, \ldots, v_n)$ une famille de vecteurs telle que $\mathcal{V} = \text{Vect}(v_0, v_1, v_2, \ldots, v_n)$ alors on dit que $(v_0, v_1, v_2, \ldots, v_n)$ est une **famille génératrice** de $\mathcal{V}$.

Comment traduire alors le résultat initial en terme de famille génératrice ?

Recherche 4 - 3 Famille génératrice de $\mathbb{R}^3$

Comment démontrer que $(\langle 3,0,0\rangle, \langle 0,0,1\rangle, \langle 0,2,0\rangle)$ est une famille génératrice de $\mathbb{R}^3$? Pourquoi le problème est différent du cas d'un $\mathbb{F}_2$-espace vectoriel ?

Recherche 4 - 4 Les canons

Montrez que $\mathbb{K}^n$ a une structure de $\mathbb{K}-$espace vectoriel.
Comment organiseriez-vous, de manière très informelle, une classe « $\mathbb{K} - EV$ » ?
Il sera utile, surtout pour des informaticiens, de garder cette image en tête.

Recherche 4 - 5

On travaille dans $\mathbb{K}^3$ considéré comme $\mathbb{K}$-espace vectoriel muni des lois canoniques.
Résolvez dans $\mathbb{R}^3$ puis dans $\mathbb{F}_2^3$ l'équation :

$$\langle 2,-3,4\rangle = x\langle 1,1,1\rangle + y\langle 1,1,0\rangle + z\langle 1,0,0\rangle$$

Recherche 4 - 6 RVB/CJMN

Comment modéliseriez-vous les systèmes RVB et CJMN en terme d'espace vectoriel ?
Comment passer en niveau de gris ? En codage 24 bits ?

Recherche 4 - 7 Espaces vectoriels pour les nuls

Avec les notations usuelles, que pensez-vous des égalités suivantes :

1. $0_+ \cdot \boldsymbol{u} = \mathbf{0}$
2. $\lambda \cdot \mathbf{0} = \mathbf{0}$
3. $(\lambda \cdot \boldsymbol{u} = \mathbf{0}) \rightarrow (\lambda = 0 \vee \boldsymbol{u} = \mathbf{0})$

Recherche 4 - 8

Soit $(E, +_E, \cdot)$ un $(\mathbb{K}, +_\mathbb{K}, \times_\mathbb{K})$-espace vectoriel. Est-ce que vous pouvez imaginer un moyen « économique » d'assurer qu'un sous-ensemble F de E ait une structure d'espace vectoriel avec les mêmes lois ?

Recherche 4 - 9 R-espaces vectoriels ?

Déterminez si les ensembles suivants munis des opérations indiquées ont une structure de $\mathbb{R}$-espace vectoriel. N'hésitez pas à voir ces exercices comme des problèmes d'instances de classes.

1. Soit $V = \{\langle a,b,c,0\rangle \mid \langle a,b,c\rangle \in \mathbb{R}^3\}$ muni de l'addition canonique des quadruplets et du produit canonique d'un quadruplet par un réel.

2. Soit $V = \{\langle a,b,c,2\rangle \mid \langle a,b,c\rangle \in \mathbb{R}^3\}$ muni de l'addition canonique des quadruplets et du produit canonique d'un quadruplet par un réel.
3. On munit $V = \mathbb{R}_+^\star$ de l'addition définie par $a \oplus b = a \times b$ et $\mathbb{R}$ opère sur le groupe $\langle V, \oplus\rangle$ par le produit externe défini par $\lambda \otimes \boldsymbol{x} = \boldsymbol{x}^\lambda$.
4. $V = \{\langle a,b,c\rangle \mid \langle a,b,c\rangle \in \mathbb{R}^3 \wedge a\cdot b\cdot c = 0\}$ avec les opérations canoniques.
5. V est l'ensemble des fonctions dérivables sur $\mathbb{R}$.
6. V est l'ensemble des suites arithmétiques réelles.

Recherche 4 - 10 K-espaces vectoriels ?

Les exemples suivants permettent-ils de définir des $\mathbb{K}$-espaces vectoriels ?

1. $\mathbb{K} = \mathbb{C}$, l'ensemble des nombres complexes, $V = \mathbb{C}$ muni de l'addition habituelle des complexes, et le produit externe est défini par :
$$(\star) : \begin{array}{ccc} \mathbb{C}\otimes\mathbb{C} & \to & \mathbb{C} \\ \langle \lambda, \boldsymbol{z}\rangle & \mapsto & \lambda^2 \cdot \boldsymbol{z} \end{array}$$
2. Soit $\mathbb{K}$ un corps quelconque et $V = \mathbb{K}^2$ muni de l'addition canonique des couples. On définit le produit externe ainsi :
$$(\star) : \begin{array}{ccc} \mathbb{K}\otimes\mathbb{K}^2 & \to & \mathbb{K}^2 \\ \langle \lambda, \langle x, y\rangle\rangle & \mapsto & \langle \lambda x, 0\rangle \end{array}$$
3. $\mathbb{K} = \mathbb{Q}$ et $V = \mathbb{R}$ munis des opérations canoniques.
4. L'ensemble des vecteurs de $\mathbb{F}_2^5$ qui ont un nombre pair de 1.
5. L'ensemble des vecteurs de $\mathbb{F}_2^5$ qui ont un nombre impair de 1.

Recherche 4 - 11

Soit V un espace vectoriel muni de tout ce qu'il faut.
Soit $\langle \boldsymbol{u_i}\rangle_{0\leqslant i\leqslant p}$ une famille de vecteurs de V. Montrez que $\mathcal{V}\text{ect}\{\boldsymbol{u_0}\cdots\boldsymbol{u_p}\}$ a une structure d'espace vectoriel.
Voilà pourquoi on appellera maintenant $\mathcal{V}\text{ect}\{\boldsymbol{u_0}\cdots\boldsymbol{u_p}\}$ le **sous-espace vectoriel engendré** par la famille $\langle \boldsymbol{u_i}\rangle_{0\leqslant i\leqslant p}$.
On dit que $\langle \boldsymbol{u_i}\rangle_{0\leqslant i\leqslant p}$ est une **famille génératrice** de $\mathcal{V}\text{ect}\{\boldsymbol{u_0}\cdots\boldsymbol{u_p}\}$.

Recherche 4 - 12

Voici des messages de 7 bits considérés comme des vecteurs de $\mathbb{F}_2^7$:

$$e_1 = 1100000,\ e_2 = 0110000,\ e_3 = 0011000,\ e_4 = 0001100,\ e_5 = 0000110,\ e_6 = 0000011$$

Est-ce que les vecteurs suivants peuvent être des combinaisons linéaires des e_i :
$u = 0010010$? $v = 0100010$?

Recherche 4 - 13

Trouvez un vecteur de $\mathbb{F}_2^4$ satisfaisant les conditions suivantes : $1100\odot x = 1,\ 1010\odot x = 1, 1111\odot x = 1$ avec $\odot$ le produit scalaire.

Recherche 4 - 14

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

1. $u = \langle 6, -9\rangle$ et $v = \langle -10, 15\rangle$. Donner des CL de la famille $\langle u\rangle$, de la famille $\langle u, v\rangle$.
   Pensez à un pauvre compilateur : comment décider si des espaces sont égaux ? Comparer tous les éléments ? Avoir une approche plus formelle ? Comment ?
   Il pourrait être utile par exemple, comme pour les rationnels, d'avoir une forme *normalisée* des vecteurs pour mieux les comparer : des idées ?
2. Que représente $\mathcal{V}\text{ect}\{\mathcal{F}\}$ avec $\mathcal{F} = \langle u, v\rangle$?
3. Démontrer que $w = \langle 2, -3\rangle \in \mathcal{V}\text{ect}\{\mathcal{F}\}$.
4. u et v sont-ils colinéaires ?
5. Démontrer que $\mathcal{V}\text{ect}\{\mathcal{F}\} = \mathcal{V}\text{ect}\{\langle w\rangle\}$.
6. Démontrer que $\langle 1, 2\rangle \notin \mathcal{V}\text{ect}\{\mathcal{F}\}$.

Recherche 4 - 15

On note $H = \left\{\langle x, y, z\rangle \in \mathbb{R}^3 \mid x + 2y + 3z = 0\right\}$, démontrer que H est un sev de $\mathbb{R}^3$ en déterminant u et v, deux vecteurs de $\mathbb{R}^3$, de sorte que $H = \mathcal{V}\text{ect}\{\langle u, v\rangle\}$. On note $w = (1, 2, 3)$. Est-ce que $w \in H$?

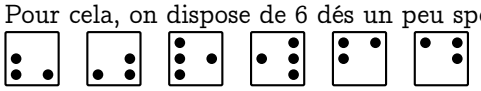
Recherche 4 - 16

1. Démontrer que $H = \left\{\langle x, y, z\rangle \in \mathbb{R}^3 \mid x - z = 0\right\}$ est un sev de $\mathbb{R}^3$.
2. Démontrer que $H = \left\{\langle x, y, z\rangle \in \mathbb{R}^3 \mid x = 0\right\}$ est un sev de $\mathbb{R}^3$.
3. $H = \left\{\langle x, y, z\rangle \in \mathbb{R}^3 \mid ax + by + cz = 0\right\}$ où a, b et c sont des réels fixés. Démontrer que H est un sev de $\mathbb{R}^3$.

Recherche 4 - 17

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

Recherche 4 - 18 Des dessins de dés

On voudrait dessiner ce dé : 
Pour cela, on dispose de 6 dés un peu spéciaux :

On peut les additionner selon la règle suivante : on superpose les dés et un point apparaît si, et seulement si, un et un seul point est présent par position.
En fait, le jeu est constitué de petites lumières. Quand on appuie sur une de ces lumières, cela change l'état (allumé ou éteint) des lumières voisines. Quel lien y a-t-il entre la situation réelle et la modélisation proposée ?
Par exemple, 1  + 1  = 
Quel est le rapport avec notre cours ? Dessinez le dé demandé. Et celui-ci  ? Et celui-là  ?
Pour aller plus loin, aller jouer au *Lights Out* : http://www.jaapsch.net/puzzles/lights.htm
Pourra-t-on résoudre toutes les énigmes ? Est-ce qu'il est vraiment utile d'appuyer sur certaines touches ?

TP 4 - 1 Recherches d'amphi

Finissez de répondre aux problèmes posés en amphi...En particulier, créez les classes de vecteurs (à l'aide de listes puis à l'aide de dictionnaires).

TP 4 - 2 Dés

Comment jouer aux dés de la Recherche 4 - 18 avec Python et des vecteurs ?

TP 4 - 3 Famille génératrice canonique

Travaillons dans un espace vectoriel de type $\mathbb{K}^n$, par exemple $\mathbb{K}^3$.
Prenons n'importe quel vecteur de $\mathbb{K}^3$ et notons-le $\langle x, y, z\rangle$.
Il est assez évident d'écrire que :

$$\langle x, y, z\rangle = x\langle 1, 0, 0\rangle + y\langle 0, 1, 0\rangle + z\langle 0, 0, 1\rangle$$

On dit que $\langle 1, 0, 0\rangle$, $\langle 0, 1, 0\rangle$ et $\langle 0, 0, 1\rangle$ sont les générateurs canoniques de $\mathbb{R}^3$.
Écrivez alors une fonction **canonique(dim, un, zero)** qui, étant donné le nombre de composantes des vecteurs, l'unité et le zéro du corps, retourne la liste des générateurs canoniques en utilisant la classe de vecteurs définis à partir de séquences.

```
In [94]: canonique1(5, Mod(1,2))
Out[94]:
[[1%2, 0%2, 0%2, 0%2, 0%2],
 [0%2, 1%2, 0%2, 0%2, 0%2],
```

```
 [0%2, 0%2, 1%2, 0%2, 0%2],
 [0%2, 0%2, 0%2, 1%2, 0%2],
 [0%2, 0%2, 0%2, 0%2, 1%2]]
```

Faites de même avec les vecteurs définis à l'aide de dictionnaires :

```
In [97]:  canonique2({'pou', 'chou', 'caillou', 'genou', 'hibou'}, Ratio(1,1))
Out[97]:
[Vecteur{'hibou': 1, 'genou': 0, 'pou': 0, 'chou': 0, 'caillou': 0},
 Vecteur{'hibou': 0, 'genou': 1, 'pou': 0, 'chou': 0, 'caillou': 0},
 Vecteur{'hibou': 0, 'genou': 0, 'pou': 1, 'chou': 0, 'caillou': 0},
 Vecteur{'hibou': 0, 'genou': 0, 'pou': 0, 'chou': 1, 'caillou': 0},
 Vecteur{'hibou': 0, 'genou': 0, 'pou': 0, 'chou': 0, 'caillou': 1}]
```

Vecteurs fonctions

Voici une troisième modélisation des vecteurs : maintenant que vous êtes chauds, vous pouvez compléter cette classe sans autre commentaire...

```
class V :
    def __init__(self, dim, fct ) :
        self.F = fct
        self.D = dim
        self.T = tuple(fct(i) for i in range(dim))

    def __hash__(self) :
        return hash(self.T)

    def __getitem__(self, i) :
        return ??????

    def __str__(self) :
        return str(self.T)

    def __repr__(self) :
        return ??????

    def __len__(self) :
        return ???????

    def __add__(self, other) :
        assert ??????????
        return ??????????

    def __mul__(self, k) :
        return ?????????

    def dot(self, other) :
        assert ????????
        return ????????

    def __eq__(self, other):
        return ????????
```

Sommes

Créez une fonction **select(veclist, k)** qui renvoi la liste des vecteurs de la liste **veclist** dont la composante **k** est nulle.
Par exemple :

```
In [99]: select(vs, 'chou')
Out[99]:
[Vecteur{'hibou': 1%2, 'genou': 0%2, 'pou': 0%2, 'chou': 0%2, 'caillou': 0%2},
 Vecteur{'hibou': 0%2, 'genou': 1%2, 'pou': 0%2, 'chou': 0%2, 'caillou': 0%2},
 Vecteur{'hibou': 0%2, 'genou': 0%2, 'pou': 1%2, 'chou': 0%2, 'caillou': 0%2},
 Vecteur{'hibou': 0%2, 'genou': 0%2, 'pou': 0%2, 'chou': 0%2, 'caillou': 1%2}]
```

Créez ensuite une fonction qui renvoie la somme d'une liste de vecteurs :

```
In [100]: somme(vs)
Out[100]: Vecteur{'hibou': 1%2, 'genou': 1%2, 'pou': 1%2, 'chou': 1%2, 'caillou': 1%2}
```

Et enfin la fonction **som_select(veclist, k)** qui effectue la somme des vecteurs de **veclist** dont la composante k est nulle.

```
In [101]: som_select(vs, 'chou')
Out[101]: Vecteur{'hibou': 1%2, 'genou': 1%2, 'pou': 1%2, 'chou': 0%2, 'caillou': 1%2}
```

TP 4 - 6 Combinaisons de $\mathbb{F}_2$-vecteurs

Construisez une fonction **F2_vect(vectlist)** qui renvoie la liste de toutes les combinaisons linéaires des vecteurs de **vectlist** sur $\mathbb{F}_2$.

```
In [109]: vs = canonique2({'pou', 'chou', 'caillou'},Mod(1,2))

In [110]: F2_vect(vs)
Out[110]:
[Vecteur{'pou': 0%2, 'chou': 0%2, 'caillou': 0%2},
 Vecteur{'pou': 1%2, 'chou': 0%2, 'caillou': 0%2},
 Vecteur{'pou': 0%2, 'chou': 1%2, 'caillou': 0%2},
 Vecteur{'pou': 1%2, 'chou': 1%2, 'caillou': 0%2},
 Vecteur{'pou': 0%2, 'chou': 0%2, 'caillou': 1%2},
 Vecteur{'pou': 1%2, 'chou': 0%2, 'caillou': 1%2},
 Vecteur{'pou': 0%2, 'chou': 1%2, 'caillou': 1%2},
 Vecteur{'pou': 1%2, 'chou': 1%2, 'caillou': 1%2}]
```

TP 4 - 7 Politique américaine

Le Sénat américain est composé de deux sénateurs par état (il y a 50 états...). Vous allez récupérer sur Madoc les statistiques sur 46 votes du 109e congrès (2006). Un 1 signifie un « Yea », un −1 un « Nay » et un 0 une abstention. Il manque les votes de Jon CORZINE, ssénateur du New-Jersey qui a été victime d'un accident de voiture cette année-là (cela fait une exception à gérer :-)).
On utilisera les fonctions **open**, **read**, **split** pour récupérer les données.

1. Créez un dictionnaire vs de type **{Nom : liste des votes}** et un autre, os, de type **Nom : [Parti, État]**. Par exemple :

```
In [18]: os['Obama']
Out[18]: ['D', 'IL']

In [20]: vs['Obama']
Out[20]: [1, -1, 1, 1, 1, -1, -1,...]
```

   On pourra utiliser la fonction **int** et des définitions par compréhension.

2. Comment utiliser le produit scalaire pour mesurer l'accord ou le désaccord entre deux sénateurs ?
3. Écrivez une fonction **compare(sen1, sen2)** qui mesure le degré d'accord de deux sénateurs. Par exemple :

```
In [21]: compare('Obama', 'McCain')
Out[21]: 16
```

4. Écrivez une fonction **senat_parti(parti)** qui renvoie l'ensemble des noms des sénateurs d'un parti donné. Créez aussi une fonction qui renvoie le nom des sénateurs par état, le nom des états par parti.
5. Écrivez une fonction **compare_sen_etat(etat)** qui mesure la cohérence des votes des deux sénateurs d'un état donné.

```
In [23]: compare_etat('WA')
Out[23]: (39, 'WA', ('Cantwell', 'Murray'))
```

   Classez les états dans l'ordre croissant des cohérences.

6. Créez une fonction **plus_eloigne(sen, ens)** qui renvoie le nom et la mesure de l'écart du sénateur d'un ensemble donné le plus en désaccord avec un sénateur donné.

```
In [33]: plus_eloigne('Obama',senat_parti('D'))
Out[33]: (22, 'Nelson2')

In [34]: plus_eloigne('Obama',senat_parti('R'))
Out[34]: (7, 'Sununu')
```

   Faites de même avec le plus en accord.

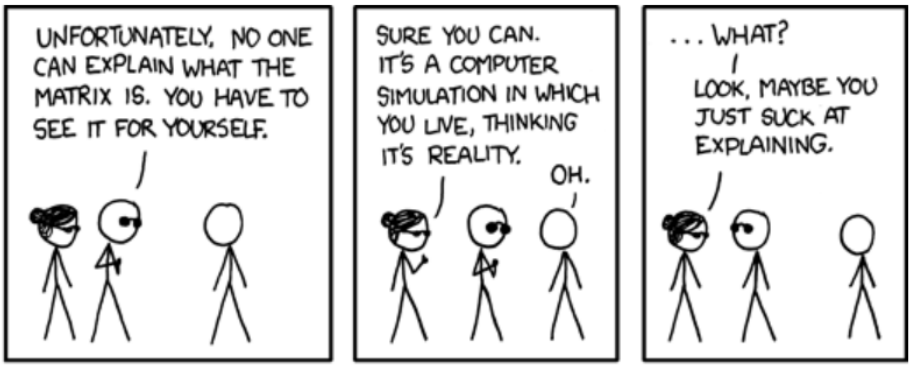
7. Quel est le parti le plus « cohérent » ? Comment le déterminer à l'aide d'une fonction calculant une moyenne de cohérence.
   Par souci d'efficacité, on pourra se souvenir que le produit scalaire est distributif sur l'addition des vecteurs.
8. Écrivez une fonction **moins_d_accord(ens)** qui renvoie le couple de sénateurs les moins d'accord d'un ensemble donné :

```
In [50]: moins_d_accord(senateurs)
Out[50]: (('Feingold', 'Inhofe'), -3)
```

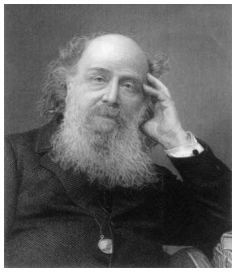
   Combien de calculs et de comparaisons sont effectués ? Pensez-vous pouvoir être plus efficace ?

9. Qui est le sénateur le plus Républicain ? L'État le plus Démocrate ? Classez les sénateurs/États selon ces critères.
10. Ouvrez le fichier **ONU_vote.txt** et étudiez-le d'une manière similaire à ce qui vient d'être fait.

La matrice



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 Python.
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 »...

Qu'est-ce que c'est ?

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** (le plus souvent commutatif) :

— $(\mathbb{A}, \boxplus)$ 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 $(\mathbb{Z}, \cdot)$?

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 $\mathbb{A}$:

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

On notera en particulier $-u = (-1_{\mathbb{A}})u$.
On peut multiplier deux vecteurs composante par composante :

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

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 $\mathbb{A}$.

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

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.

Matrices particulières

— On peut définir des **matrices par blocs** :

$$\left(\begin{array}{|c|cc} a_{11} & a_{12} & a_{13} \\ \hline a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}\right)$$

pourra s'écrire :

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$$

avec $A \in \mathbb{A}^{1\times1}$, $B \in \mathbb{A}^{1\times2}$, $C \in \mathbb{A}^{2\times1}$ et $D \in \mathbb{A}^{2\times2}$

— Les **matrices diagonales** : $(a_{ij} = 0)$ pour $i \neq j$. On les notera :

$$A = \text{diag}\left[a_{11}, a_{22}, ..., a_{mm}\right]$$

Avec $A \in \mathbb{A}^{n\times p}$ et $m = \min(n, p)$.

— Les **matrices tridiagonales** telles que $a_{ij} = 0$ pour $|i - j| > 1$.

— Les **matrices triangulaires supérieures** telles que $a_{ij} = 0$ pour $i > j$.

— Les **matrices triangulaires inférieures** telles que $a_{ij} = 0$ pour $i < j$.

— Les **matrices de Hessenberg supérieures** telles que $a_{ij} = 0$ pour $i > j + 1$.

— Les **matrices de Hessenberg inférieures** telles que $a_{ij} = 0$ pour $j > i + 1$.

— Une **matrice colonne** est une matrice de $\mathbb{A}^{n\times1}$.

— Une **matrice ligne** est une matrice de $\mathbb{A}^{1\times n}$.

— Les **matrices Attila** sont les matrices remplies de 1.

— Les **matrices nulles** sont les matrices remplies de 0.

— Algorithmiquement parlant, il peut être intéressant de traiter à part les matrices contenant proportionnellement beaucoup de 0 : on les appelle les **matrices creuses**.

Donnez des exemples de matrices de chacun de ces types.

Opérations sur les matrices

Égalité

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

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...

Multiplication par un scalaire

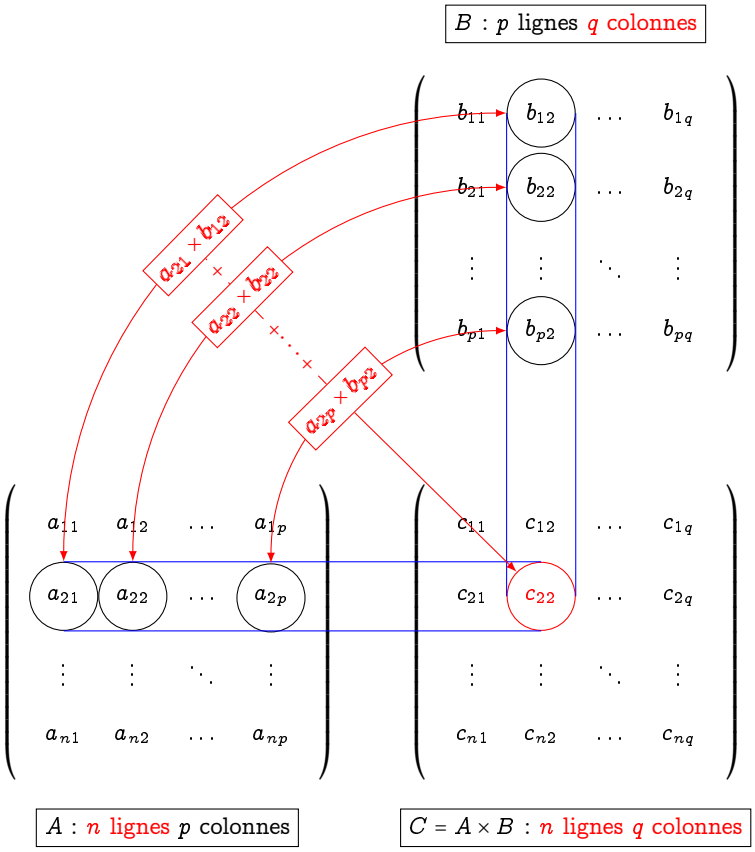
Soit $k \in \mathbb{A}$ et $M \in \mathbb{A}^{n\times p}$ alors la matrice $P = kM$ est définie par :

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

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.

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 ?

Recherche

Est-ce qu'on ne pourrait pas parler de produit matriciel en utilisant le produit scalaire ?...

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 $T = {}^{t}A$.
En pratique, on échange lignes et colonnes.

Recherche

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

Définition 4 - 2

Une matrice telle que ${}^{t}A = A$ est dite **symétrique**.
Une matrice telles que ${}^{t}A = -A$ est dite **antisymétrique**

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 ?

Recherche 4 - 19

L'ensemble des matrices à coefficients dans un corps $\mathbb{K}$ de la forme :

$$\begin{pmatrix} a & a+b \\ a+b & b \end{pmatrix}$$

muni de l'addition matricielle ainsi que de la multiplication par un scalaire est-il un espace vectoriel ?
Même question avec les matrices de la forme $\begin{pmatrix} a & 0_{\mathbb{K}} \\ 1_{\mathbb{K}} & b \end{pmatrix}$ et celles de la forme $\begin{pmatrix} 0_{\mathbb{K}} & a \\ b & 0_{\mathbb{K}} \end{pmatrix}$

Recherche 4 - 20

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, $A \times G$, $G \times A$.

Recherche 4 - 21

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.

Recherche 4 - 22 Puissances de matrices

1. Soit $A = \frac{1}{3}\begin{pmatrix} 0 & -2 & -2 \\ 2 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}$.
   i. Calculer A^2, A^3 et A^4.
   ii. Montrez que $\{A, A^2, A^3, A^4\}$ est un groupe pour le produit matriciel de $M_3(\mathbb{R})$.
2. Pour chacune des matrices A suivantes, calculer A^n pour tout $n \in \mathbb{N}$.
   i. $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$ avec $a \in \mathbb{R}$.
   ii. La matrice J_p de $M_p(\mathbb{R})$ dont tous les coefficients sont égaux à 1.

Recherche 4 - 23 Matrices et relations

Considérons par exemple la relation $\mathcal{R}$ définie par son dictionnaire d'association :

De	Vers
a	γ
b	β,δ
c	γ
d	δ

Donnez sa représentation matricielle.
Donnez la représentation matricielle d'une relation incluse dans $\mathbb{R}$.
À votre avis, pourquoi le nom « transposée » a-t-il été bien choisi pour désigner la relation réciproque ?
Donnez la représentation matricielle de se transposée. Généralisez.
Quelle est la représentation matricielle de la négation de $\mathcal{R}$?
Soit la relation $\mathcal{R}'$:

De	Vers
a	α, γ
b	α, β, δ
c	β, δ
d	α, β, γ

Quelle est la représentation matricielle de $\mathcal{R} \cap \mathcal{R}'$? De $\mathcal{R} \cup \mathcal{R}'$?
Quelle est le rapport entre composition de relations et calcul matriciel ?
Quelle est la représentation matricielle de $\mathcal{R} \circ {}^{t}\mathcal{R}'$?

Recherche 4 - 24

1. Donner la représentation matricielle des relations suivantes définies sur $\{1,2,3,4\}$:
   - **i.** $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$;
   - **ii.** $\{(1,1),(1,4),(2,2),(3,3),(4,1)\}$;
   - **iii.** $\{(1,2),(1,3),(1,4),(2,1),(2,3),(2,4),(3,1),(3,2),$ $(3,4),(4,1),(4,2),(4,3)\}$;
   - **iv.** $\{(2,4),(3,1),(3,2),(3,4)\}$
2. Quelles sont, parmi ces relations, celles qui sont réflexives ? irréfléxives ? symétriques ? antisymétriques ? transitives ? Donner des algorithmes qui répondent à ces questions en prenant les relations en arguments et en effectuant des calculs sur les matrices.

Recherche 4 - 25

Soit $\mathcal{E} = \{1,2,3,4,5\}$ et $\mathcal{R}$ une relation sur $\mathcal{E}$ qui contient les couples $(1,3)$, $(2,4)$, $(3,1)$, $(3,5)$, $(4,3)$, $(5,1)$, $(5,2)$ et $(5,4)$. Déterminez :

1. $\mathcal{R}^2$
2. $\mathcal{R}^3$
3. $\mathcal{R}^4$
4. $\mathcal{R}^5$
5. $\mathcal{R}^6$

Utilisez le calcul matriciel.

Recherche 4 - 26

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

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

Recherche 4 - 27

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

Recherche 4 - 28

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

Recherche 4 - 29

$A = (a_{ij}) \in \mathbb{R}^{n\times n}$. Expliciter la matrice A dans les cas suivants :

1. $a_{ij} = i + j$ si $i > j$ et $a_{ij} = 0$ sinon.
2. $a_{ij} = j$ si $i \leq j$ et $a_{ij} = i$ si $i > j$.
3. $a_{ij} = |i - j - 1|$

Recherche 4 - 30

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

1. Démontrer que B est symétrique.
2. $B = (b_{i,j})$, calculer $b_{i,j}$ en fonction des $a_{k,l}$.

Recherche 4 - 31

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

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

Recherche 4 - 32

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

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

Recherche 4 - 33

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

Recherche 4 - 34

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

1. Calculer A^2.
2. Calculer $A^2 - 3A + 2I_3$.
3. En déduire que A est inversible et déterminer A^{-1}.

Recherche 4 - 35

a) Soit $U = \begin{pmatrix} 0 & 0 & 5 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix}$. Exprimer de façon simple la matrice U^3 comme combinaison linéaire 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}.

Recherche 4 - 36 Espace vectoriel de matrices

Nous désignerons par E_{ij} la matrice carrée de $\mathbb{R}^{2\times 2}$ dont tous les coefficients sont nuls sauf le coefficient (i, j) qui vaut 1. Par exemple, $E_{12} = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$ dans $\mathbb{R}^{2\times 2}$.
Dans toute la suite de l'exercice, E désignera le $\mathbb{R}$-espace vectoriel $\mathbb{R}^2 = \{(x, y) \mid x \in \mathbb{R} \wedge y \in \mathbb{R}\}$ muni des lois usuelles.

1. Démontrez que $E = \mathrm{Vect}(E_{11}, E_{12}, E_{21}, E_{22})$.

2. Démontrez que la famille de vecteurs $\mathcal{F} = (E_{11}, E_{12}, E_{21}, E_{22})$ est libre. Qu'en déduisez-vous sur cette famille ?

3. On note $S = \left\{ \begin{pmatrix} a & c \\ c & b \end{pmatrix} \mid (a,b,c) \in \mathbb{R}^3 \right\}$. Démontrez que $\mathcal{C} = (E_{11}, E_{22}, E_{12} + E_{21})$ est une base de S.

4. On note φ la fonction de $\mathbb{R}^3$ dans S définie par :

$$\varphi(x,y,z) = \begin{pmatrix} x & z \\ z & y \end{pmatrix}$$

Démontrez que $\varphi \in \mathcal{L}(\mathbb{R}^3, S)$, ensemble des applications linéaires de $\mathbb{R}^3$ dans S.

5. *Question bonus* : déterminez $\text{mat}_{\mathcal{B},\mathcal{C}}\varphi$ avec $\mathcal{B} = \big\{(1,0,0),(0,1,0),(0,0,1)\big\}$.

Recherche 4 - 37

Soit la matrice $A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$.

a) Déterminer A^n pour tout $n \in \mathbb{N}$.

b) On considère les deux suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ définies par la donnée de u_0 et v_0 et les relations de récurrence

$$\begin{cases} u_{n+1} &= u_n - v_n \\ v_{n+1} &= -u_n + v_n \end{cases}$$

(i) On pose $W_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$. Etablir une relation entre W_{n+1}, A et W_n.

(ii) En déduire une expression de u_n et v_n en fonction de u_0, v_0 et n pour tout $n \in \mathbb{N}$.

Recherche 4 - 38 Dynamique des populations

Dans une île vendéenne, on trouve des tyrannosaures (prédateurs) et des pintades (proies). On note les populations de tyrannosaures et de pintades après k années par T_k et P_k. Un biologiste a avancé que les populations varient d'année en année selon le modèle suivant :

$$\begin{cases} T_k = 0,6T_{k-1} + 0,3P_{k-1} \\ P_k = -0,5T_{k-1} + 1,4P_{k-1} \end{cases}$$

Soit $M_k = \begin{pmatrix} T_k \\ P_k \end{pmatrix}$ la matrice des populations de tyrannosaures et de pintades après k années et $M_0 = \begin{pmatrix} 10 \\ 30 \end{pmatrix}$ la matrice de la population initiale.

On note $A = \frac{1}{10}\begin{pmatrix} 6 & 3 \\ -5 & 14 \end{pmatrix}$

1. Quelle est la population initiale des tyrannosaures ?

2. Justifiez le fait que $a_{1,0} < 0$ et $a_{1,1} > 0$.

3. Exprimez la matrice des populations après k années en fonction de celle des populations de l'année précédente.

4. Exprimez la matrice des populations après k années en fonction de celle des populations initiales.

5. Quelle sera la population de tyrannosaures après 3 ans ?

6. Proposez une résolution de la question précédente à l'aide de la classe **`Mat`** introduite dans l'exercice précédent.

Recherche 4 - 39

Dans une ferme du bocage vendéen, un fermier élève des pintades.
Les pintades se reproduisent selon le schéma suivant :

$$p_{n+2} = p_{n+1} + p_n$$

avec p_n le nombre de pintades au bout de n mois sachant que $p_0 = p_1 = 1$.
On veut connaître le nombre de pintades au bout de 1111 mois.

1. Estimez le nombre d'opérations arithmétiques élémentaires pour calculer p_{1111}. Combien de temps cela prendra-t-il au fermier d'effectuer le calcul sur son PC équipé d'un processeur I5 de 60 GFLOPs ?

2. Comment le calcul matriciel peut sauver le fermier ?

Recherche 4 - 40

Dans toute la suite U désigne une suite de 4 bits qui peut être représentée par la matrice

$$U = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}$$

et la transposée de U est notée X :

$${}^tU = X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

et il est bien entendu que les éléments x_i sont considérés comme des éléments du corps $\mathbb{F}_2$ en ce sens que $1 + 1 = 0$. Pour améliorer la sécurité de la transmission des chaînes de 4 bits on décide de transformer le message U en

$$V = U \times B$$

où B est une matrice (judicieusement choisie) à coefficients dans $\mathbb{F}_2$ ou, si l'on préfère, on transforme le message X en $Y = A \times X$

1. Quelle relation existe-t-il entre Y et V ? Entre A et B ?

2. Si on veut que $V = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_2 \end{pmatrix}$, donner la matrice B.

3. Si on veut que $Y = \begin{pmatrix} x_1 \\ x_1 + x_2 \\ x_3 \\ x_{1+}x_4 \\ x_2 \\ x_3 \end{pmatrix}$, donnner la matrice A.

4. On veut rajouter au message U un bit de parité paire (pour un bit de parité impaire, ce qui suit est impossible), décrire alors V et donner alors B ?

5. On veut que $V = \begin{pmatrix} x_4 & x_3 & x_2 & x_1 & x_1 & x_2 & x_3 & x_4 \end{pmatrix}$, déterminer B.

6. Quelqu'un décide de choisir

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

i. Calculer les images des éléments suivants :

U	V
1111	
1100	
0011	
0110	
0101	
0010	
1010	
0100	

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

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

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

Recherche 4 - 41

Une matrice est dite **orthogonale** si, et seulement si, sa transposée est égale à son inverse.

1. La matrice I_n est-elle orthogonale ?

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

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

Recherche 4 - 42 Calculs complexes

On note i le nombre de carré -1. On note $e^{it} = \cos t + i \sin t$. On note $C = \{e^{it} \mid t \in \mathbb{R}\}$

1. Démontrez que C muni de la multiplication des nombres complexes a une structure de groupe.

2. On considère deux réels t et t'. Calculez la partie réelle et la partie imaginaire de $e^{it} \times e^{it'}$. Déduisez-en que

$$\cos(t+t') = \cos t \cos t' - \sin t \sin t'$$

. Déterminez une formule similaire pour $\sin(t+t')$.

3. On considère la matrice $A = \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix}$.

Calculez A^2 et A^3.

4. Donnez une forme simple donnant A^k pour tout entier naturel k et démontrez votre résultat.

5. Calculez $A \times {}^t A$ en donnant la forme la plus simple possible.

6. *Question bonus* : on note $R = \left\{ \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix} \mid t \in \mathbb{R} \right\}$.

Démontrez que R muni de la multiplication matricielle a une structure de groupe.

7. Lien avec les rotations ?

TP 4 - 8 La Matrice

Fabriquons nos outils

Matrice — du mot latin *matrix* (matricis), lui-même dérivé de *mater*, qui signifie « mère » — un élément qui fournit un appui ou une structure, et qui sert à entourer, à reproduire ou à construire.

Création d'une classe « Matrice »

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

```
1 class Mat:
2     """ une matrice sous la forme Mat([nb lignes, nb cols], fonction(i,j))"""
3
4     def __init__(self, dim, f):
5         self.F = f   # fonction (i,j) -> coeff en position i,j
6         self.D = dim # liste [nb de lignes, nb de cols]
```

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

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

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

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

La matrice précédente aurait donc pu être définie par :

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

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

Recherche

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

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

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

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

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

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

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

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

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

Recherche

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

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

Transposée d'une matrice

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

Recherche

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

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

Somme de matrices

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

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

Par exemple, avec la matrice M précédemment introduite :

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

Recherche

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

Produit par un scalaire

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

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

Recherche

Écrivez une implémentation de la méthode prod_par_scal

Produit de matrices

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

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

C'est l'algorithme habituel trois boucles imbriquées).
Nous allons l'abstraire d'un petit niveau : chaque coefficient c_ij est en fait égal au produit scalaire de la ligne i de A et de la colonne j de B.
Commençons donc par construire une fonction qui calcule le produit scalaire de deux itérables.

Pour cela, nous allons utiliser une fonction extrêmement importante qui vient du monde de la programmation fonctionelle : `map`.
map(fonc, iterable(s)) renvoie un itérateur de type map mais où chaque élément sera remplacé par son image par la fonction (si c'est une fonction de une variable) ou par l'image combinée des itérables si c'est une fonction de plusieurs variables.
Par exemple :

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

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

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

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

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

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

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

Avec une fonction de deux variables et deux itérables :

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

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

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

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

Recherche

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

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

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

```
        else:
            return self.prod_par_scal(other)
```

Comparaison de plusieurs produits matriciels

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

Avec les outils standards de Python

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

```
import numpy as np

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

Avec Cython

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

```
In [1]: %load_ext cythonmagic

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

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

    assert ca == rb, 'Tailles non compatibles'

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Puissances d'une matrice

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

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

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

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

```
            if k == 0:
                return acc
            return pui((m*m),k//2,acc if k % 2 == 0 else (m*acc))
        return pui(self,n,unite(r))
```

Recherche Déterminez une version impérative de la méthode précédente.

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 :

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

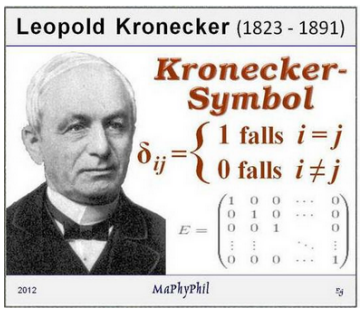
On note alors $N = M^{-1}$.
On remarque (n'est-ce pas) que, d'après cette définition, $(M^{-1})^{-1} = M$.
Lorsque nous aurons étudié les déterminants, nous pourrons démontrer que si A et B sont deux matrices carrées de taille n vérifiant $A \times B = \mathbb{I}_n$, alors elles sont régulières et $A = B^{-1}$.

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 ?

Opérations sur les lignes

2 6 9 a 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 $\mathbb{Z}^{3\times 3}$.



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 ?

2 6 9 b 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 $(\mathbb{I}_n + \lambda E_n^{ij})^{-1} = \mathbb{I}_n - \lambda E_n^{ij}$.

2 6 9 c Dilatations de lignes

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

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

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

2 6 9 d Échange de lignes

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

2 6 9 e Opérations sur les lignes

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

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

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

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

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

2 6 9 f Lien avec les matrices régulières

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

Théorème 4 - 1

φ étant une opération élémentaire sur les lignes,

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

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

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

qui se transforme en

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

et en définitive

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

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

Théorème 4 - 2

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

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

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

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

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

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 $\mathbb{A}^{n\times p}$.

Matrices ligne-équivalentes

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

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

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

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

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

$$N = R \times M$$

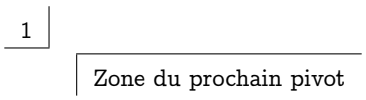
R étant le résultat du produit de matrices du type $\varphi(\mathbb{I}_n)$. Pour obtenir cette matrice R il suffit d'appliquer successivement en parallèle les opérations élémentaires qui transforment M en N sur $\mathbb{I}_n$. Pour cela on utilise un tableau où l'on juxtapose M et $\mathbb{I}_n$, M étant 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	$N = R \times M$

L réduite échelonnée

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

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



Donnons quelques exemples de matrices entières :

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

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

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

Théorème 4 - 3

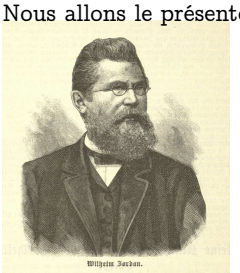
Pour toute matrice $M \in \mathbb{A}^{n\times p}$, il existe une unique matrice ℓ-réduite échelonnée $N \in \mathbb{A}^{n\times p}$ telle que $M \overset{\ell}{\equiv} N$, N est appelée la ℓ-réduite échelonnée de M que l'on peut noter par $N = \ell\text{ré}(M)$.

La démonstration de ce théorème se fait par récurrence sur n et elle est un peu difficile.
Le **rang** de M, noté $\text{rang}(M)$, est le nombre de lignes non nulles de sa ℓ-réduite échelonnée, c'est donc aussi le nombre de pivots da sa ℓré. M est dite de plein rang si son rang est égal à son nombre de lignes.
Nous énoncons les évidences (conséquences directes de la définition) :

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

Algorithme Fang-Tcheng

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



Wilhelm JORDAN
(1842-1899)

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

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

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

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

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

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

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

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

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

— **Deuxième étape.** On repère dans la deuxième colonne (en fait la colonne qui suit la dernière colonne pivot obtenue), à partir de la deuxième ligne (en fait à partir du numéro de ligne qui suit le numéro de la ligne qui a donné le dernier pivot), le plus grand élément en valeur absolue (si on obtient zéro on passe à la colonne suivante, etc···). Ici on obtient 1 qui se trouve sur la deuxième ligne et deuxième colonne et il n'y a pas de permutation de lignes à faire. Maintenant il faut faire apparaître des zéros dans la colonne pivot en ligne 1 et en ligne 3. On utilise les opérations $L_1 \leftarrow L_1 - 2L_2$ et $L_3 \leftarrow L_3 - 0L_2$ (qui ne sert à rien) ; on obtient :

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

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

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

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

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

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

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

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

Théorème 4 - 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.

Résolution de systèmes

Généralités

Gabriel Cramer
(1704-1752)

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

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

Soit aussi

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

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

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

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

soit aussi

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

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

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

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

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

ou bien la matrice nulle $0_{p,1}$ si on note matriciellement cette solution.
Le système $A \times X = B$ est dit **régulier** ou de **Cramer** si, et seulement si, il a autant d'équations que d'inconnues et si A est inversible (le système ayant autant d'équations que d'inconnues on a $n = p$ et A est une matrice carrée). A étant inversible, A^{-1} existe et

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

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

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

En retranchant membre à membre on obtient :

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

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

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

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

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

Systèmes équivalents et résolution

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

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

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

   Pour retrouver l'équation initiale il suffit de multiplier l'équation obtenue par $\frac{1}{\alpha}$ et par conséquent le système obtenu par cette opération est équivalent au système initial.
2. Permutation de deux équations, nous noterons $E_i \longleftrightarrow E_j$ l'opération qui consiste à permuter l'équation numéro i avec l'équation numéro j. Pour retrouver le système initial il suffit d'appliquer de nouveau cette opération, cela nous permet d'affirmer que cette opération transforme un système en un système équivalent.
3. Ajout à une équation une autre équation multipliée par un scalaire : si on ajoute à l'équation E_i l'équation $E_{h \neq i}$ multipliée par β nous noterons $E_i \leftarrow E_i + \beta E_h$

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

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

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

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

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

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

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

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

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

$$A \times U = B$$

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

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

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

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

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

$$A \times U = B$$

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

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

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

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

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

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

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

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

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

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

Vade-mecum

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

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

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

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

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

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

— Une CL de vecteurs c'est un calcul (qui ne comporte que des additions de vecteurs ou des multiplication par un scalaire) où interviennent ces vecteurs.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

on en déduit que

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

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

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

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

$$P \times X' = X$$

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

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

Ce résultat est à retenir.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

— Endomorphismes de $\mathbb{R}^n$:

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

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

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

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

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

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

  ce qui équivaut à écrire

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

  et

$$\begin{cases} f : & E \longrightarrow E \\ & x \longmapsto y = f(x) \end{cases}$$

$$A = \text{Mat}(f, \mathcal{B}, \mathcal{B}) = \text{Mat}_{\mathcal{B}}(f)$$

$$Y = A \times X$$

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

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

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

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

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

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

- Si A est de rang n, cela équivaut à écrire que A est inversible, et donc f est surjective.
- Nous venons de démontrer que les affirmations suivantes sont équivalentes
  - f est injective.
  - $\ker(f) = \{0_E\}$.
  - f est surjective.
  - L'image d'une base de E par f est une base de E.
  - f est bijective.
  - A est inversible.

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

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

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

$$Y = A \times X$$

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

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

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

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

et comme P est inversible

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

ce qui nous donne

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

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

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

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

- Valeurs propres et diagonalisation : on se propose de chercher s'il existe des sev U de sorte que la restriction de f à U soit une homothétie (si $u \in U$, $f(u) = \lambda u$).
  - $\lambda \in \mathbb{R}$ est une valeur propre (vap) de f ssi il existe au moins un vecteur non nul vérifiant $f(u) = \lambda u$.
  - si λ est une vap de f alors $E_\lambda = \{u \in E \mid f(u) = \lambda u\}$ est un sev de E, c'est le sous espace propre (sep) associé à la valeur propre λ.
  - L'ensemble des valeurs propres de f est appelé le spectre de f.
  - La dimension d'un sep est forcément ≥ 1.
  - Si f n'est pas injective ce qui équivaut à dire que $\ker(f)$ n'est pas réduit à $\{0_E\}$ alors 0 est valeur propre et $E_{\lambda=0} = \ker(f)$.
  - La juxtaposition des bases des différents sep est une famille libre. Si cette famille est une base, on dit que cette base est une base de vecteurs propres et la matrice de f dans cette base est diagonale, on dit que l'on a diagonalisé f.

Exercices

Recherche 4 - 43

Déterminer les ℓ-réduites échelonnées (ℓré) des matrices suivantes et donner leur rang :

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

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

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

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

Recherche 4 - 44

Les matrices suivantes sont-elles inversibles ? On cherchera, pour chacune d'elles, l'inverse par la méthode de Gauss.

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

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

3. $C = \begin{pmatrix} 1 & 2 & -1 \\ -1 & -1 & 2 \\ 2 & 1 & 1 \end{pmatrix}$

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

Recherche 4 - 45

Soit $D = [d_{1,1}, d_{2,2}, \ldots, d_{n,n}]$ une matrice diagonale, démontrer que D est inversible si, et seulement si, $\prod_{i=1}^{n} d_{i,i} \neq 0$.

Recherche 4 - 46

Inverser les matrices suivantes :

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

Recherche 4 - 47

Ecrire matriciellement de deux façons les systèmes suivants :

1. $\begin{cases} x - 2y + 3z = 2 \\ 2x - 5y - 4z = -1 \\ 4y - 3z = 8 \end{cases}$

2. $\begin{cases} 3x - z = 6 \\ 5y - 4z = -3 \\ 4x - 3y = 7 \end{cases}$

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

Recherche 4 - 48

Résoudre le système :

$$\begin{cases} x - 2y + z = a \\ 2x - 3y - 2z = b \\ x - y + z = c \end{cases}$$

obligatoirement par la méthode Gauss-Jordan où x, y et z sont les inconnues. En déduire que la matrice $A = \begin{pmatrix} 1 & -2 & 1 \\ 2 & -3 & -2 \\ 1 & -1 & 1 \end{pmatrix}$ est inversible et donner son inverse.

Recherche 4 - 49

Résoudre les systèmes suivants par la méthode de Gauss-Jordan :

1. $\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 2 \\ 7x + 8y + 9z = 3 \end{cases}$

2. $\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 2 \\ 5x + 7y + 9z = 3 \end{cases}$

3. $\begin{cases} x + 2y + 3z + 4t = 1 \\ 4x + 5y + 6z + 2t = 2 \\ 7x + 8y + 9z - t = 3 \end{cases}$

4. $\begin{cases} x + y + z = 1 \\ x - y - 2z = 3 \\ 2x + 3y + 5z = 0 \\ 4x + 3y + 4z = 4 \end{cases}$

5. $\begin{cases} x + y + z = 1 \\ x - y - 2z = 3 \\ 2x + 3y + 5z = 0 \\ 4x + 3y + 4z = 1 \end{cases}$

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

Recherche 4 - 50

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

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

Recherche 4 - 51

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

$$M\begin{pmatrix} x \\ y \end{pmatrix}_{\mathcal{R}}, \quad M\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathcal{R}}$$

Un changement de repère peut concerner un changement d'origine ou un changement de base ou les deux. Soit donc le repère $\mathcal{R}=(O,i,j,j)$ avec $\mathcal{B}=(i,j,k)$ la base associée et $\mathcal{R}'=(O',i',j',k')$ un autre repère avec $\mathcal{B}'=(i',j',k')$ la base associée. Nous notons

$$M\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathcal{R}}, M\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\mathcal{R}'}, O'\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}_{\mathcal{R}} \text{ et } M\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}_{\mathcal{R}_1=(O,i',j',k')}$$

et $P=(p_{i,j})$ la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$, c'est-à-dire la matrice des coordonnées des vecteurs de $\mathcal{B}'$ dans la base $\mathcal{B}$.

1. Quelle sont les coordonnées de O dans le repère $\mathcal{R}$?
2. Démontrer que P est inversible en cherchant les coordonnées des vecteurs de $\mathcal{B}$ dans la base $\mathcal{B}'$.
3. On note $X=\begin{pmatrix} x \\ y \\ z \end{pmatrix}, X'=\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}, X_1=\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$. Démontrer que $X_1=P\times X$.
4. Déterminer X' en fonction de X et de P.

Recherche 4 - 52

Soit $A=\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$, $B_1=\begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$ et $B_2=\begin{pmatrix} 32,1 \\ 22,9 \\ 33,1 \\ 30,9 \end{pmatrix}$
Résolvez les systèmes $A\times X=B_1$ et $A\times X=B_2$: commentaires ?

Recherche 4 - 53

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 ?

TP 4 - 9

Back to code : Inverse d'une matrice par la méthode de Gauss-Jordan

Opérations élémentaires

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

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

Recherche

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

Calcul de l'inverse étape par étape

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

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

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

Complétons-la par la matrice identité de même taille :

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

Recherche

Complétez...

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

        """
        ???
```

On effectue $L_3 \leftarrow L_3 - L_1$:

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

puis $L_3 \leftarrow L_3 - L_2$:

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

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

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

On effectue $L_1 \leftarrow L_1 - L_3$:

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

et enfin $L_2 \leftarrow L_2 - L_3$:

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

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

Recherche

Complétez...

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

    """
    ???
```

alors :

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

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

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

Réduction sous-diagonale d'une matrice

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

```
    def triangle(self):
        """ renvoie la triangulation d'une matrice de haut en bas """
        [r,c] = self.D
        m     = min(r,c)
        S     = self
        T     = zeros(r,c)
        while m > 0:
            NoLigne = 0
            while S[NoLigne, 0] == 0 and (NoLigne < m - 1):
                    NoLigne += 1
            if S[NoLigne, 0] != 0:
                pivot = S[NoLigne,0]
                for k in range(1,m):
                    if S[k,0] != 0:
                        S = S.comb_lignes(pivot, -S[k,0],k,0)
                        print("pivot = "+str(pivot))
                        print("S dans for :")
                        print(S)
            T = T.remplace_ligned(r - m,S.F)
            print("Évolution de T :")
            print(T)
            S = S.dswap(NoLigne)
            m -= 1
        return T
```

Par exemple :

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

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

```
 0  3
Évolution de T :
 1  2  3
 0 -3 -6
 0  0  0
Évolution de T :
 1  2  3
 0 -3 -6
 0  0  3
```

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

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

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

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

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

Calcul du déterminant

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

Recherche Créez une fonction qui calcule le déterminant :

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

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

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

Calcul de l'inverse d'une matrice

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

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

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

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

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

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

Recherche

Construisez diago_triangle

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

Recherche

On considère la matrice

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

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