C. Charignon

Table des matières

Première partie
Cours

1 Rappels de mathématique

Dans cette partie, nous fixons $\mathbb{K}$ un corps et $(n,p) \in \mathbb{N}^2$.

1.1 Vocabulaire

Soit $M \in \mathcal{M}_{n,p}(\mathbb{K})$.

- La matrice M est dite *échelonnée en lignes* lorsque chaque ligne est nulle ou contient strictement plus de zéros consécutifs au début que la précédente.
  Si M est échelonnée en ligne, le premier coefficient non nul de chaque ligne s'appelle un *pivot* .
- La matrice M est dite *réduite en ligne* lorsqu'elle est échelonnée et que chaque pivot vaut 1, et chaque pivot est le seul coefficient non nul de sa colonne.
- Opérations sur les lignes :
  - ◇ Pour tout $(i,j) \in [\![1,n]\!]^2$, nous appellerons l'opération $L_i \leftrightarrow L_j$ une *transposition* sur les lignes de M ;
  - ◇ Pour tout $i \in [\![1,n]\!]$ et $\lambda \in \mathbb{K}^*$, l'opération $L_i \leftarrow \lambda L_i$ sera appelée une *dilatation* ;
  - ◇ Pour tout $(i,j) \in [\![1,n]\!]^2$ tel que $i \neq j$ et tout $\lambda \in \mathbb{K}$, l'opération $L_i \leftarrow L_i + \lambda L_j$ sera appelée une *transvection*.
- Si N est une autre matrice de format $n \times p$, on dit que N est *équivalente* à M en lignes, et on note $M \underset{l}{\sim} N$ lorsqu'il existe une suite de dilatations, transvection, et permutations des lignes de M permettant de la transformer en N.

Le théorème suivant du à Gauss affirme qu'il est toujours possible de réduire une matrice, mais aussi que le résultat est unique :

Théorème 1.1. *Toute matrice M est équivalente en ligne à une* unique *matrice réduite en ligne.*

Définition 1.2.

Soit $M \in \mathcal{M}_{n,p}(\mathbb{K})$.

- L'unique matrice réduite équivalente en lignes à M s'appelle la réduite de Gauss (en lignes) de M.
- On appelle « rang » de M le nombre de lignes non nulles dans la réduite de M.

On démontre en particulier que M est inversible ssi sa réduite est I_n.

Proposition 1.3.

Soit $M \in \mathcal{M}_{n,p}(\mathbb{K})$. Le rang de M est également égal au nombre de pivot dans *n'importe quelle* matrice échelonnée obtenue par opérations sur les lignes à partir de M.

Vocabulaire : Nous dirons que nous effectuons un pivot de Gauss partiel sur M lorsque nous obtenons seulement *une* matrice échelonnée. Nous dirons que nous effectuons un pivot total lorsque nous obtenons *la* matrice réduite de M.

1.2 Utilisations du pivot de Gauss

Une matrice peut être utilisée pour décrire (liste non exhaustive) :

- un système d'équations
  - ◇ système homogène de n équations et p inconnues,
  - ◇ ou un système de n équations et $p-1$ inconnues, avec un second membre. Dans ce cas la dernière colonne contient le second membre, et on dit que M est la matrice « augmentée » du système.
- une famille de vecteurs, contenant p vecteurs dans un espace de dimension n ;
- une application linéaire, entre un espace de dimension p et un espace de dimension n.

Rappelons les utilisations principale du pivot de Gauss dans ces trois cas.

- **pivot partiel :** Un pivot partiel permet de calculer le rang de M. À partir de là :
  - ◇ si M est la matrice d'un système d'équations (S), on déduit combien de solution (S) admet (aucune, une unique, ou une infinité, auquel cas le nombre de paramètres pour décrire l'ensemble des solutions est « nombre d'inconnues − rang »).
  - ◇ si M est la matrice d'une application linéaire ϕ, celle-ci est injective ssi $\text{rg}(M) = p$ et surjective ssi $\text{rg}(M) = n$. Plus généralement, on a $\text{rg}(M) = \text{rg}(\phi) = \dim(\text{Im}(\phi))$ et le théorème du rang permet d'obtenir la dimension du noyau de ϕ, via la formule $\dim(\text{Ker}(\phi)) = p - \text{rg}(\phi)$.
  - ◇ si M est la matrice d'une famille de vecteurs $\mathcal{F}$ dans un espace vectoriel E, celle-ci est libre ssi $\text{rg}(M) = p$ et génératrice de E ssi $\text{rg}(M) = n$; plus généralement le rang de M est égal à $\dim(\text{Vect}(\mathcal{F}))$.
- **pivot total :**
  - ◇ si M est la matrice d'un système d'équations (S), on obtient les solutions de (S) ;
  - ◇ si M est la matrice d'une application linéaire ϕ, on obtient son noyau.
  - ◇ si M est la matrice d'une famille de vecteurs $\mathcal{F}$, en effectuant un pivot total sur les lignes de M, on trouve les relations de dépendance linéaire entre les vecteurs de $\mathcal{F}$. Cela peut servir si on souhaite utiliser le théorème de la base extraite (si on veut retire les vecteurs « superflus » dans $\mathcal{F}$) .
- dans certains cas, il peut aussi être utile d'effectuer un **pivot sur les colonnes** :
  - ◇ si M est la matrice d'une application linéaire ϕ on obtient une base de l'image de ϕ.
  - ◇ si M est la matrice d'une famille de vecteurs $\mathcal{F}$, on obtient une base de l'espace qu'elle engendre.

2 Implémentation du pivot de Gauss

Voici une implémentation possible de la méthode du pivot de Gauss.

2.1 Opérations élémentaires sur les lignes

On commence par programmer les opérations élémentaires sur les lignes.

- `echangeLigne`, complexité en $O(1)$.
- `transvection`, complexité en $O(p)$.
- `dilatation`, complexité en $O(p)$.

```python
def afficheMat(M):
    """ Affiche une matrice de manière un tout petit peu plus lisible."""
    for l in M:
        print(l)

def echangeLigne(M, i1, i2):
    """ Effectue l'opération Li1 <-> Li2 """
    M[i1], M[i2] = M[i2], M[i1]

def dilatation(M, i, alpha):
    """ Effectue l'opération Li <- alpha*Li """
    p=len(M[i])
    for j in range(p):
        M[i][j] = alpha * M[i][j]

def transvection(M,i1,i2,alpha):
    """ Effectue l'opération M[i1] <- M[i1] + alpha*M[i2]"""
    p=len(M[i1])
    for j in range(p):
        M[i1][j]=M[i1][j]+alpha*M[i2][j]
```

Attention : La fonction `echangeLigne` ne marche pas sur un tableau numpy ! Pour commencer, détaillons la manière dont notre code est interprété :

```
def echangeLigne(M, i1, i2):
    """ Effectue l'opération Li1 <-> Li2 """
    sauv=M[i1]
    M[i1]=M[i2]
    M[i2] = sauv
```

Maintenant, il faut savoir que lors de l'instruction `M[i1]=M[i2]` le tableau `M[i1]` est *modifié* en y mettant le contenu de `M[i2]`, alors que le comportement du type `list` classique de Python est de faire pointer `M[i1]` vers le tableau `M[i2]`.

Ainsi, numpy interprète la ligne `M[i1]=M[i2]` en :

```
for i in range(len(M[i1])):
    M[i1][j]=M[i2][j]
```

Ainsi, à partir du moment où cette instruction *modifie* `M[i1]`, puisque `M[i1]==sauv`, elle modifie aussi `res` et on comprend la suite...

Voici une version de `echangeLigne` qui marche aussi avec les tableaux numpy :

2.2 AnnuleColonne

On en déduit alors la procédure `annuleColonne` qui prend en entrée une matrice M et deux indices i et j et qui utilise le coefficient $M_{i,j}$ comme pivot pour annuler tous les autres coefficients de la colonne j de M (c'est-à-dire de `M[:,j]`).

On suppose que le pivot $M_{i,j}$ est égal à 1 : si ce n'est pas le cas, utiliser au préalable la fonction `dilatation`.

```
def annuleCol(M,i0,j):
    """ précondition : M[i,j]==1.
     Utilise M[i,j] comme pivot pour annuler le reste de la colonne M[:,j]"""
    assert M[i0,j]==1
    for i in range(len(M)):
        if i!=i0:
            transvection(M,i,i0,-M[i,j]) # Li <- Li - M[i,j]*Mi0
```

Exemple : Réduisons les matrices suivantes en utilisons nos opérations élémentaires sur les lignes :

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

2.3 L'algorithme principal

La méthode du pivot de Gauss en lignes classique consiste à traiter chaque colonne l'une après l'autre.

Pour chaque colonne, on cherche un élément non nul, on le place en position de pivot, et on l'utilise pour annuler les autres coefficients de la colonne.

Si il n'y a aucun élément non nul dans la colonne, c'est que cette colonne n'a pas besoin d'être traitée : passer simplement à la colonne suivante.

On va écrire une fonction `cherchePivot` prenant en entrée une matrice M, un indice de colonne j, et un indice de ligne i et cherchant un pivot dans la colonne $M[:,j]$ en dessous de la case $M[i,j]$. Cette fonction renverra `None` lorsqu'aucun pivot n'aura été trouvé.

```
def cherchePivot(M,i,j):
    n=len(M)
    res=i
    while res<n and M[res,j]==0:
        res+=1
    if res == n : return None
```

```
    else: return res
```

Ce qui nous mène au programme final. Il y a juste deux points à retenir :

- on procède colonne après colonne : le cœur du programme est donc une boucle `for j in range(p)`, où p est le nombre de colonnes.
- il nous faut garder en mémoire un indice $i0$ qui est le numéro de ligne ou nous voudrons placer le prochain pivot.

L'invariant de boucle que nous allons maintenir au cours de la procédure est :

À la fin de l'itération j, les j premières colonnes de M forment une matrice échelonnée, dont les dernier pivot est à la ligne $i0 - 1$.

Remarque : En fait la plupart du temps, les matrices auxquelles ont appliquera le pivot de Gauss seront de rang maximal, ce qui fait qu'on aura toujours $i0 = j$. On pourrait donc s'épargner quelques subtilités dans la programmation.

```
def pivot(M):
    n, p = len(M), len (M[0])
    i0=0
    for j in range(p):
        i=cherchePivot(M,i0,j)
        if i!=None:
            echangeLigne(M,i,i0) # On place le pivot au bon endroit
            dilatation(M,i0, 1/M[i0,j]) # On transforme le pivot en 1
            annuleColonne(M,i0,j)
            i0+=1
        #Sinon : rien à faire, la colonne j est déjà réduite
```

N.B. Le programme `pivot` que nous venons d'écrire est une procédure : il ne renvoie rien. Par contre, il modifie la matrice passée en argument.

3 Erreur d'arrondi et amélioration

(Présenté sous forme d'exercice)

On fixe $\epsilon \in]0,1[$, et on définit le système d'équations $(S) : \begin{cases} \epsilon x + y = 0{,}5 \\ x + y = 1{,}0 \end{cases}$ d'inconnue $(x, y) \in \mathbb{R}^2$.

1. Résoudre (S).
2. Vers quelles limites tendent les solutions lorsque $\epsilon \to 0$? Cela semble-t-il cohérent ?
3. Tester le programme `pivot` pour de petites valeur de ϵ. Que se passe-t-il ?
4. Écrire la matrice augmentée de (S), et suivre pas à pas la résolution par l'algorithme du pivot de Gauss. Ne pas simplifier les formules obtenues afin de bien voir les opérations effectuées par l'ordinateur.
   Expliquer ce qui s'est passé. Le problème est lors du calcul de $\dfrac{1 - \dfrac{0{,}5}{\epsilon}}{1 - \dfrac{1}{\epsilon}}$.
5. En supposant que l'ordinateur utilisé a un processeur à 64 bits, et que les flottants sont représentés comme d'ordinaire à l'aide de 1 bit de signe, 11 d'exposant, et 52 de mantisse, à partir de quelle valeur de ϵ observe-t-on ce comportement aberrant ?
6. À présent, on présente le système (S) en échangeant les deux lignes : $(S') : \begin{cases} x + y = 1 \\ \epsilon x + y = 0{,}5 \end{cases}$.
   Suivre de nouveau l'application de l'algorithme. Quel résultat obtiendra-t-on lorsque ϵ est très proche de 0 ? Commentaires ?

On constate sur cet exemple qu'il est plus prudent de choisir le pivot le plus grand possible. C'est pourquoi on propose cette nouvelle version du programme `cherchePivot` :

```python
def meilleurPivot(M,i0,j):
    meilleurActuel = i0, 0 #indice, module du pivot
    for i in range(i0,len(M)):
        if abs(M[i,j]) > meilleurActuel[1]:
            meilleurActuel = i, abs(M[i,j])
    return meilleurActuel
```

La fonction principale peut alors être réécrite ainsi :

```python
def pivotLigne(M):
    n,p=M.shape
    i0=0 # ligne où on veut mettre le prochain pivot
    for j in range(p):
        ipivot, val=meilleurPivot(M,i0,j) # (indice, valeur absolue) du pivot
        if val!=0:
            echangeLigne(M,ipivot,i0)
            dilatation(M,i0, 1/M[i0, j])
            annuleCol(M,i0,j)
            i0+=1
        #sinon : pas de pivot, donc rien à faire!
```

N.B. En pratique, comme on l'a déjà dit, le test de savoir si un coefficient est nul est vain sur une matrice de flottants. Donc en réalité le cas où en renvoie `None` est inutile. Ce qui signifie qu'un ordinateur est incapable en pratique de détecter à coup sûr une matrice qui n'est pas de rang maximal.

4 Quelques applications

4.1 Résoudre un système

Soit (S) un système de n équations linéaires à p inconnues. Pour le résoudre grâce à notre programme, il suffit de l'appliquer à sa matrice augmentée M. Comme notre programme ne modifie M que par des opérations élémentaires sur les lignes, nous sommes certains que le système contenu dans M après l'exécution est équivalent au système (S) initial. Voyons quelques exemples :

- *Cramer* Soit $(S1) : \begin{cases} x+2y+3z=1 \\ 2x+z=2 \\ x+y+z=3 \end{cases}$. Notons $M1$ sa matrice augmentée, donc $M1 = \begin{pmatrix} 1 & 2 & 3 & 1 \\ 2 & 0 & 1 & 2 \\ 1 & 1 & 1 & 3 \end{pmatrix}$. Après exécution du programme `pivotLigne`, nous voyons que `M1` contient `[[1.0, 0.0, 0.0, 2.333333333333333], [0.0, 1.0, 0.0, 3.333333333333333], [-0.0, -0.0, 1.0, -2.6666666666666665]]`, ce qui signifie que :

$$M_1 \underset{l}{\sim} \begin{pmatrix} 1 & 0 & 0 & \frac{7}{3} \\ 0 & 1 & 0 & \frac{10}{3} \\ 0 & 0 & 1 & -\frac{8}{3} \end{pmatrix}$$

Comme les opérations sur les lignes mènent à des systèmes équivalent au système de départ, nous déduisons :

$$(S1) \Leftrightarrow \begin{cases} x = \frac{7}{3} \\ y = \frac{10}{3} \\ z = -\frac{8}{3} \end{cases}$$

On voit que de réduire la matrice augmentée fournit immédiatement la solution du système.

- *Exemple avec moins d'équations que d'inconnues :* Soit $(S2) : \begin{cases} x+2y+3z=1 \\ 2x+z=2 \end{cases}$, et M_2 sa matrice augmentée, donc $M_2 = \begin{pmatrix} 1 & 2 & 3 & 1 \\ 2 & 0 & 1 & 2 \end{pmatrix}$. On trouve que $M_2 \underset{l}{\sim} \begin{pmatrix} 1 & 0 & \frac{1}{2} & 1 \\ 0 & 1 & \frac{5}{4} & 0 \end{pmatrix}$. Ainsi :

$$(S2) \Leftrightarrow \begin{cases} x + \frac{1}{2}z = 1 \\ y + \frac{5}{4}z = 0 \end{cases}$$

Il suffit de «faire passer les paramètre (ici z) de l'autre côté :

$$(S2) \Leftrightarrow \begin{cases} x = -\frac{1}{2}z + 1 \\ y = -\frac{5}{4}z \end{cases}$$

Et nous pouvons immédiatement exprimer le résultat sous forme paramétrée :

$$\mathcal{S}_{(S2)} = \left\{ (1 - \frac{1}{2}z, -\frac{5}{4}z, z) \ ; \ z \in \mathbb{R} \right\}$$

Ou encore donner une base de type «point + espace vectoriel directeur» :

$$\mathcal{S}_{(S2)} = (1, 0, 0) + \text{Vect}\left((-\frac{1}{2}, -\frac{5}{4}, 1) \right)$$

Dans les situations suivantes, la méthode est moins fiable à cause des erreurs sur les flottants.

- *Système incompatible*
- *Infinité de solutions*
- *Solution unique mais pas Cramer*

Remarque : Dans numpy se trouve `np.linalg.solve` pour résoudre un système d'équations. La syntaxe est `np.linalg.solve(A, B)` où A est la matrice du système et B le second membre, mais mis en ligne et non en colonne.

4.2 Inverser une matrice

Soit $A \in \mathcal{G}l_n(\mathbb{C})$. Pour inverser A, on utilise la méthode de Gauss-Jordan. Sur un papier, cette méthode consiste à écrire I_n à côté de A, à réduire A par opérations sur les lignes, et à appliquer également à I_n chaque opération effectuée sur A.

Pour utiliser notre programme, il suffit de créer une matrice M de format $n \times 2n$, dont la partie gauche contiendra initialement A et la partie droite I_n. Lorsque nous appliquerons un pivot total à M, la partie gauche deviendra I_n, et les opérations effectuées sur les lignes auront automatiquement été effectuées aussi sur la partie droite, qui contiendra donc à la fin de la procédure A^{-1}.
Remarque : Dans numpy, utiliser `np.linalg.inv`

4.3 Calculer un rang

Pour calculer le rang d'une matrice, on peut la réduire, puis compter le nombre de lignes non nulles à la fin.

Cependant, compte tenu des erreurs inhérentes aux flottants, une ligne qui aurait du être nulle aura toutes les chances de ne pas l'être exactement...

Ce genre de difficultés dépasse le cadre de ce cours, et en pratique, nous resterons dans des situations où les matrices sont de rang maximal.
Remarque : Dans numpy : `nump.linalg.matrix_rank`.

Deuxième partie
Exercices

Exercices : pivot de Gauss

1 Autour du programme principal

Exercice 1. **! Utilisation directe de l'algorithme du pivot de Gauss

1. Écrire une fonction pour résoudre un système d'équation lorsqu'il est de Cramer. La fonction prendra en entrée la matrice augmentée du système et renverra le tableau des solutions.
2. Écrire une fonction qui applique la méthode de Gauss-Jordan pour calculer l'inverse d'une matrice.

Exercice 2. * Réduction en colonne

En utilisant le programme vu en cours pour réduire une matrice en ligne, et une fonction qui calcule la transposée, obtenir un programme pour réduire une matrice en colonnes.

Exercice 3. * Matrice réduite ou échelonnée**

Écrire un prédicat qui indique si une matrice est réduite en ligne ou échelonnée mais non réduite, ou non échelonnée.

2 Applications

2.1 Diététique

Exercice 4. * Repas équilibré

Voici les valeurs nutritionnelles de quelques aliments. L'énergie est en kJ par 100 g, et les autres sont exprimé en pourcentages massiques :

	énergie	glucides	lipides	protéines
Chips classiques natures	2188	46.7	34	5.9
Moutarde à l'ancienne	664	2.3	11	7.7
Amandes enrobées de chocolat et cacao	2324	48.1	35.3	10.1
Curry madras	1605	43.5	13.9	13.9

Par ailleurs les recommandations de la directive européenne 1990/496, remplacée par le réglement 1169/2011 sont les suivantes, pour une journée :

- Énergie : 8400 kJ
- Glucides : 260 g
- Lipides : 70 g
- Protéines : 50 g

Expliquer comment obtenir un repas équilibré à l'aide de chips classiques nature, de moutarde à l'ancienne, de curry madras et d'amandes enrobées de chocolat et cacao.

2.2 Maths

Exercice 5. ** Calcul de déterminant

1. Quelle serait la complexité du calcul d'un déterminant par l'application directe de sa définition ?
2. Quelle serait la complexité du calcul d'un déterminant par la formule du développement selon une ligne (ou une colonne) ?

   N.B. Il s'agit d'un programme récursif : établir une relation de récurrence sur la complexité.
3. Finalement, écrire un programme basé sur une modification du programme du pivot de Gauss. Préciser sa complexité.

Exercice 6. * Rang

1. Écrire une fonction qui calcule le rang d'une matrice échelonnée.
2. Soit $\mathcal{F} = \left(X^2+X+1, 2X^2-X-1, X+2, X^3-2\right)$. En réduisant $\mathrm{Mat}_{\mathrm{can}}\mathcal{F}$, déterminer le rang de $\mathcal{F}$. Est-elle libre ? Engendre-t-elle $\mathbb{R}_2[X]$?
3. Soit $\phi : \begin{array}{ccc} \mathbb{R}_2[X] & \to & \mathbb{R}^2 \\ P & \mapsto & \big(P(2)+P'(1), P''(0)-P(1)\big) \end{array}$. En réduisant $\mathrm{Mat}_{\mathrm{can,can}}(\phi)$, déterminer $\mathrm{rg}(\phi)$. Cette application linéaire est-elle injective ? Surjective ?
4. Que penser de l'utilité pratique de la fonction rang ?

2.3 Chimie

Exercice 7. * Acide citrique

L'acide citrique, de formule $C_6H_8O_7$ (présent par exemple dans le citron) peut être utilisé pour détartrer les bouilloires. En effet, il réagit avec le calcaire, de formule $CaCO_3$, en produisant de l'eau, du gaz carbonique, et du $Ca_3\left(C_6H_5O_7\right)_2$ qui se dissout dans l'eau.

Une autre utilité de cette réaction est de reconnaître le calcaire. En effet si quelques gouttes de jus de citron sur un caillou produisent un dégagement de gaz carbonique, alors ce caillou contient du calcaire.

Équilibrer cette réaction, sans faire les calculs vous-même.

Exercice 8. ** ! Triacide

On considère le tri-acide H_3PO_4. Les trois couples acide/base correspondants sont $(H_3PO_4/H_2PO_4^-)$, $(H_2PO_4^-/HPO_4^{2-})$ et (HPO_4^{2-}/PO_4^{3-}), de pKa respectifs 2,1, 7,2 et 12,4.

On part d'une concentration de 1 mol/L qu'on notera C_0.

On notera c_1, c_2, c_3, c_4 les concentrations en H_3PO_4, $H_2PO_4^-$, HPO_4^{2-} et $P0_4^{3-}$ respectivement.

1. On suppose le pH de la solution égal à 6.
   (a) Écrire l'équation liant pH, pKa, et les concentrations acide/base pour chacun des trois couples. Mettre le tout sous la forme d'un système de trois équations linéaires en $\ln(c_1)$, $\ln(c_2)$, $\ln(c_3)$, $\ln(c_4)$.
   (b) Réduire ce système au moyen du programme du pivot de Gauss vu en cours.
   (c) En utilisant une dernière équation venant de la conservation de la quantité de phosphore, achever de déterminer c_1, c_2, c_3 et c_4.
   (d) Refaire le calcul en négligeant ce qui vous paraît digne de l'être comme on le fait généralement en chimie et comparer les résultats.
2. Plus généralement, écrire un programme calculant chaque concentration en fonction du pH.
3. Tracer les courbes donnant chaque concentration en fonction du pH.
4. Une célèbre boisson folklorique américaine aromatisée au cola a un pH de 2.5, obtenu par adjonction d'acide phosphorique. Quel est la forme dominante de ce tri-acide dans cette boisson ?