TP 3 – Matrices, pivot de Gauss et RSA

▶ **Prérequis** : *Chapitre 3 du polycopié.*

1 Matrices

On peut construire des matrices et des vecteurs de la façon suivante

```
A = matrix([[1,2,3],[4,5,6]])
B = matrix(2,3,[1,2,1/2,-1,0,5])
v = vector([0,1,-1])
```

- Comment récupérer les coefficients de `A`, de `B`, de `v` ? Que donnent les commandes `v*A`, `A*v`, `B*v` ? Que donne `A.transpose()` ? Que donnent `A.parent()`, `B.parent()`, `v.parent()` ? Que donnent `B.nrows()` et `B.ncols()` ?

On peut aussi construire des matrices ou des vecteurs à coefficients dans un anneau donné.

- Essayer les commandes suivantes et préciser les objets qu'elles définissent. On appliquera en particulier la méthode `parent` à chacun d'entre eux :

```
w = vector(CC,[0,5])
C = matrix(QQ,A)
D = matrix(GF(5),A)
M23 = MatrixSpace(RR,2,3), E = M23([1,2,3,4,5,6])
M4=MatrixSpace(CC,4)
F = M4(1), H=M4(0)
identity_matrix(10)
```

On peut enfin construire des matrices dont les coefficients vérifient une formule donnée.

- Essayer les commandes suivantes

```
G = matrix(QQ,4, lambda i,j: i-j)
v = vector(RR,[i^2 for i in range(3)])
```

- Construire la matrice de Hilbert de taille 5, c'est-à-dire la matrice carrée de taille 5 dont le coefficient en position (i, j) est donné par $\frac{1}{i+j-1}$, vue comme matrice à coefficients rationnels.
- Que donne `G[[0,2,3],1]` ? Que donne `parent` appliqué à cet objet ? Essayer des variantes, comme par exemple la commande `G[[1,0],[0]+[2]]`.

On donne à présent un certain nombre de commandes `Sage` qu'il peut être utile de connaître et de maîtriser !

- Comment calculer la somme, le produit de deux matrices ? Le déterminant et l'inverse d'une matrice ? Tester les commandes `kernel`, `right_kernel`, `image`, `row_module`, `column_module` sur une matrice. On comparera les résultats pour la matrice `A` ci-dessus et la même matrice vue comme matrice à coefficients rationnels.
- Créer une matrice `A` et un vecteur `v` et résoudre le système linéaire `Ax=v` grâce à la commande `A.solve_right()`. Que se passe-t-il si `A` n'est pas injective ? Si l'équation n'a pas de solution ?
- Soit `A=matrix(QQ,3,[1,2,3,3,2,1,1,1,1])`. Tester

```
characteristic_polynomial, minimal_polynomial, eigenvalues,
 eigenvectors_right, eigenspaces_right, eigenmatrix_right, jordan_form
```

  sur `A`.
- Même question avec `B=matrix(QQ,2,[0,1,2,0])`. Que se passe-t-il ? Réessayer ces commandes en voyant `B` comme une matrice à coefficients dans $K = \mathbf{Q}(\sqrt{2})$.

On termine cette section avec un petit exercice d'application.

- On définit l'espace vectoriel $V = \mathbf{F}_2^4$ et on tire trois vecteurs $u, v, w \in V$ aléatoirement suivant une loi uniforme. Comment définir le sous-espace $W = \text{Vect}(u, v, w)$? Comment en donner une base ? Tirer un autre élément aléatoirement dans V et tester s'il appartient à W. Construire le quotient de V par W et donner sa dimension. Construire de même un deuxième sous-espace aléatoire W' à l'aide de deux vecteurs de V et donner une base de $W \cap W'$.

2 Pivot de Gauß

- On se fixe un anneau A. Rappeler le coût, en termes d'opérations dans A et en fonction des dimensions des matrices, des calculs suivants : somme de deux matrices, produit[1] de deux matrices, d'une matrice et d'un vecteur colonne, élévation d'une matrice carrée à une puissance m donnée.

2.1 Le cas d'un corps

- Implémenter un algorithme naïf pour calculer le déterminant d'une matrice carrée. Le tester et le comparer à la fonction `det` de `Sage`.
- On suppose ici que l'on travaille sur un **corps** k. Implémenter l'algorithme du pivot de Gauß du polycopié en une fonction `pivot` qui prend en argument une matrice carrée et renvoie cette matrice sous forme échelonnée. Quelle est la complexité de cet algorithme ? Pourquoi a-t-on eu besoin de l'hypothèse que k est un corps ?
- Implémenter un algorithme de résolution d'un système linéaire de la forme $Ax = v$ avec A une matrice et v un vecteur. Préciser le coût de cet algorithme.
- Implémenter un algorithme fournissant l'inverse d'une matrice carrée. Préciser le coût de cet algorithme. On suppose qu'on a plusieurs systèmes linéaires de la forme $MX_i = B_i$ à résoudre. À partir de combien de tels systèmes est-il plus rentable de calculer M^{-1} que de résoudre les systèmes un par un ?
- Implémenter un algorithme calculant le déterminant d'une matrice carrée, le rang d'une matrice, une base de son noyau, de son image. Préciser le coût de ces algorithmes.

2.2 Un algorithme dans le cas de $\mathbb{Z}$

On donne à présent une méthode alternative pour calculer le déterminant d'une matrice carrée à coefficients entiers.

- Établir *l'inégalité de Hadamard* pour une matrice carrée M à coefficients complexes

$$|\det(M)| \leqslant \prod_{i=1}^{n} \left(\sum_{j=1}^{n} |m_{ij}|^2 \right)^{\frac{1}{2}}.$$

  On suppose désormais que M est à coefficients entiers. Montrer qu'on peut calculer le déterminant de M de la façon suivante et implémenter cet algorithme :

  (i) Trouver des nombres premiers $p_1, \dots, p_r$ tels que $p_1 p_2 \cdots p_r > 2|\det(M)|$;

  (ii) Calculer $\det(M)$ modulo p_i pour tout $1 \leqslant i \leqslant r$ et en déduire $\det(M)$.

2.3 Un algorithme sans division

On ne suppose plus que A est un corps et on suppose seulement qu'il s'agit d'un anneau commutatif[2]. On pose $M^{(0)} = M$, puis pour k allant de 0 à $n-2$, on effectue :

(i) Si les lignes numéro k à $n-1$ de la matrice $M^{(k)}$ sont nulles, l'algorithme est terminé ;

(ii) a) Si la ligne k de la matrice $M^{(k)}$ est nulle mais qu'il existe une ligne de numéro $> k$ non nulle, échanger la ligne k avec une ligne non nulle de numéro $> k$;

b) Si $m_{k,k}^{(k)} = 0$, échanger la colonne k avec une colonne de numéro $j > k$ tel que $m_{k,j}^{(k)} \neq 0$;

(iii) Pour j allant de k à $n-1$, remplacer dans $M^{(k)}$ la ligne L_j par $m_{k,k}^{(k)} L_j - m_{j,k}^{(k)} L_k$;

(iv) La matrice obtenue à la suite de ces opérations est appelée $M^{(k+1)}$.

- Exprimer le déterminant de $M^{(k+1)}$ en fonction de celui de $M^{(k)}$.
- En déduire un algorithme pour calculer le déterminant de M et l'implémenter.
- En déduire un algorithme pour calcule le polynôme caractéristique d'une matrice M à coefficients dans un corps et l'implémenter.
- On suppose que les coefficients de M sont de taille majorée par t. Donner une majoration de la taille des coefficients de $M^{(k)}$ en fonction de ceux de M et faire afficher ces coefficients sur plusieurs exemples. Que constatez-vous ?

La méthode précédente fait apparaître des coefficients qui peuvent devenir très grands. On présente donc une méthode, dite de Gauß-Bareiss, également sans division mais pour laquelle les coefficients croissent moins vite. On fixe une suite (c_k) d'éléments de A non nuls (dépendant de M) et on définit une suite $\left(M^{(k)}\right)$ de matrices comme précédemment, sauf qu'on remplace l'étape **(iii)** par

1. On pourra se renseigner sur l'algorithme de Strassen pour une amélioration de cette complexité dans le même veine que Karatsuba.
2. On pourrait travailler dans le corps de fractions de A lorsque A.

(iii)' Pour j allant de k à n, remplacer dans $M^{(k)}$ la ligne L_j par $c_k^{-1}(m_{k,k}^{(k)}L_j - m_{j,k}^{(k)}L_k)$;

Noter que le cas $c_k = 1$ correspond à l'algorithme précédent et le cas $c_k = m_{k,k}^{(k)}$ au pivot de Gauß. On cherche alors une suite (c_k) telle que $M^{(k)}$ reste à coefficients dans A mais que ces coefficients ne deviennent pas trop gros. On pose $\delta^{[0]}(1,1) = 1$ et

$$\delta^{[k]}(i,j) = \det \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,k} & m_{1,j} \\ m_{2,1} & & \cdots & & m_{2,j} \\ \vdots & \vdots & & \vdots & \vdots \\ m_{k,1} & m_{k,2} & \cdots & m_{k,k} & m_{k,j} \\ m_{i,1} & m_{i,2} & \cdots & m_{i,k} & m_{i,j} \end{pmatrix}.$$

- Soit $N = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ une matrice par blocs avec A, D carrées et A inversible. Montrer que $\det(N) = \det(A)\det(D - CA^{-1}B)$.
- En déduire que si $i, j > k+1$, on a

$$\delta^{[k+1]}(i,j)\delta^{[k+1]}(k+1,k+1) = \delta^{[k]}(k+1,k+1)\delta^{[k]}(i,j) - \delta^{[k]}(i,k+1)\delta^{[k]}(k+1,j).$$

  On pose alors $c_k = \delta^{[k]}(k+1,k+1)$ et on note $M^{(k)}$ la matrice obtenue par l'algorithme ci-dessus avec l'étape **(iii)'**.
- Montrer que l'algorithme est bien défini et que $M^{(k)}$ est bien à coefficients dans A. On pourra vérifier que pour $i, j > k$, on a $M^{(k)}[i,j] = \delta^{[k]}(i,j)$.
- Montrer que pour calculer c_k, on n'a en fait pas besoin de calculer $\delta^{[k]}(k+1,k+1)$. Implémenter cet algorithme pour calculer le déterminant et le polynôme caractéristique d'une matrice.
- On suppose ici que $A = \mathbf{Z}$ et que tous les coefficients de M sont de taille majorée par t. Montrer que tous les coefficients de $M^{(k)}$ ont une taille majorée par $n\left(\frac{1}{2}\log(n) + t\right) + 1$. Quel est le coût de l'algorithme précédent ?

2.4 Compléments sur le polynôme caractéristique

- Justifier que la connaissance de $\det(M - xI_n)$ pour $x \in \{0, \ldots, n-1\}$ permet de déterminer χ_M. Implémenter cet algorithme et en préciser le coût.
- Rappeler les relations entre les sommes de Newton et les fonctions symétriques élémentaires d'une famille de n éléments de A. Expliquer comment calculer χ_M à partir des traces des M^i pour $i \in \{1, \ldots, n\}$. Implémenter cet algorithme [3] et en préciser le coût.

3 RSA

Pour tester vos différents programme dans cette partie, on peut produire des nombres premiers de grandes tailles grâce aux commandes `next_prime` ou `previous_prime` mais aussi via `P=Primes()` et `P.unrank(54)`.

3.1 RSA

- Écrire un programme qui prend en entrée la clé publique (n, e) d'un cryptosystème RSA et un message en clair m, puis renvoie le message chiffré $m^e \pmod n$.
- Écrire un programme de déchiffrement, connaissant les facteurs premiers de n.
- On peut utiliser le théorème chinois pour accélerer le calcul de $c^d \pmod n$ (où c est le message chiffré, d l'exposant de déchiffrement), en calculant $c^{d(\bmod p)-1} \pmod p$ et $c^{d(\bmod q)-1} \pmod q$. Justifier cette approche et l'implémenter.
- Comparer les temps d'exécution en faisant varier e (pour n fixé).

3.2 Quelques attaques

- Supposons que $n = pq$ avec p et q "proches". Montrer que l'on peut écrire $n = r^2 - s^2$ avec r de l'ordre de grandeur de $\sqrt{n}$ (et donc s petit). Expliquer comment déterminer facilement si un entier est le carré d'un entier et en déduire un programme qui prend en entrée n et renvoie (p, q) en temps rapide si p et q sont proches.
- Écrire un programme permettant de retrouver un message en clair m en prenant en entrée ses chiffrés pour deux clés publiques (n, e_1) et (n, e_2) où e_1, e_2 sont premiers entre eux.

On présente à présent l'attaque de Wiener. Soit $n = pq$ avec p et q premiers tels que $p \in]q, 2q[$.

3. Dit de Le Verrier.

- Montrer que si, pour une clé RSA publique (n, e), on a $d < \frac{1}{3}\sqrt[4]{n}$, alors un attaquant peut calculer d en temps polynomial à partir de n et e. On pourra utiliser pour cela le théorème suivant :
  Théorème 1.– *Soient $\frac{a}{b}$ et $\frac{c}{d}$ deux fractions sous forme irréductible telles que $\left|\frac{a}{b} - \frac{c}{d}\right| \leqslant \frac{1}{2d^2}$. Alors $\frac{c}{d}$ est une des réduites du développement de $\frac{a}{b}$ en fractions continues.*
- Pour éviter cette attaque, on donne (n, e') comme clé RSA publique où $e' = e + t\varphi(n)$ avec t grand. Montrer que si $e' > n\sqrt{n}$, alors même si $d < \frac{1}{3}\sqrt[4]{n}$, l'attaque ci-dessus ne peut pas être effectuée.
- Montrer que si $ed = 1 + k\varphi(n)$ et $d < \frac{1}{3}\sqrt[4]{n}$, alors $\frac{k}{d}$ est l'une des réduites du développement de $\frac{e}{n}$ en fraction continue.
- En déduire un programme qui prend en entrée (n, e) et renvoie d en temps rapide si $d < \frac{1}{3}\sqrt[4]{n}$.

3.3 Factorisation

- Écrire un programme qui prend en entrée n et k, puis renvoie `Echec` si $k \neq \varphi(n)$ ou si n n'est pas le produit de deux nombres premiers, et renvoie les deux facteurs premiers de n sinon.

Supposons que $n = pq$. On note e, premier avec $\varphi(n)$, l'exposant public d'un système RSA de module n.

- Montrer comment, à partie de e, d et n, on peut construire un multiple B de $\varphi(n)$.
- On note $m = \text{ppcm}(p-1, q-1)$. Montrer que pour tout $a \in (\mathbf{Z}/n\mathbf{Z})^\times$, on a $a^m = 1$. Montrer que $a^{m/2}$ peut prendre quatre valeurs et que deux de ces valeurs permettent de factoriser n.
- On pose H l'ensemble des éléments de $(\mathbf{Z}/n\mathbf{Z})^\times$ tels que $a^{m/2} = \pm 1 \pmod{n}$. Montrer qu'il s'agit d'un sous-groupe de $(\mathbf{Z}/n\mathbf{Z})^\times$.
- Montrer qu'il existe $b \in (\mathbf{Z}/n\mathbf{Z})^\times$ tel que b soit d'ordre $p-1$ modulo p et d'ordre $\frac{q-1}{2}$ modulo q.
- On pose $p - 1 = 2^{\nu_p} p'$ et $q - 1 = 2^{\nu_q} q'$ avec p', q' impairs et on suppose, sans perte de généralité, que $\nu_p \geqslant \nu_q$. Exprimer $\frac{m}{2}$ en fonction de ν_p et du ppcm de p' et q'. En déduire que b n'appartient pas à H. Si l'on prend x au hasard dans $(\mathbf{Z}/n\mathbf{Z})^\times$, montrer que la probabilité que x n'appartienne pas à H est supérieure ou égale à $\frac{1}{2}$.
- Montrer que m divise B et en déduire qu'il existe un entier naturel k tel que pour tout $x \in (\mathbf{Z}/n\mathbf{Z})^\times$, on a $x^{\frac{m}{2}} \equiv x^{B/2^{k+1}} \pmod{n}$.
- En déduire et implémenter un algorithme polynomial qui factorise n étant donnés n, e et d et donner sa probabilité de succès.