Algèbre linéaire.

February 9, 2001

L'algèbre linéaire se prête bien au calcul sur machine : on aborde ici la réduction sous forme échelonnée (de Gauß) des matrices et les méthodes de réduction des endomorphismes (diagonalisation).

1 Réduction et factorisation de matrices

1.1 Méthode de Gauß

Remarques :
- On veut résoudre le système $AX = Y$. Quand A est inversible ce système est équivalent à $X = A^{-1}Y$, mais le calcul direct de A^{-1} est couteux et inadapté (car cela nécessite la résolution de n systèmes linéaires $A\epsilon_i = e_i$).
- Si A est triangulaire supérieure la résolution de $AX = Y$ est immédiate : c'est la méthode des remontées.
La méthode de Gauß:
Elle consiste à déterminer une matrice inversible M telle que $MA = U$ soit triangulaire supérieure et à résoudre $UX = MY$ par la méthode des remontées.
On fait apparaître des zéros sous la diagonale principale de A par une combinaison de lignes c'est à dire en multipliant A à gauche par des matrices $M_{i,j}$.
Exemple :
Soit $A = a_{i,j}$ inversible et supposons $a_{1,1} \neq 0$ (si ce n'est pas le cas, A étant inversible, un des $a_{i,1}$ au moins est non nul, on échange alors la ligne i correspondante avec la première ligne ce qui revient à multiplier A à gauche par une matrice P_1).
Exercice 1
Déterminer P_1

$a_{1,1}$ est alors appelé le premier pivot.
On définit la matrice M_1 par :
$m_{i,i} = 1$
$m_{i,1} = -\dfrac{a_{i,1}}{a_{1,1}}$ $(i > 1)$
$m_{i,j} = 0$ $(j > 1, j \neq i)$
M_1 est inversible, $\det(M_1) = 1$ et $M_1 A$ a alors comme première colonne $a_{1,1}, 0, ...0$.
On continue le processus en choisisant le deuxième pivot : si $M_1 A = \alpha_{i,j}$ ce sera $\alpha_{2,2}$ (ou $\alpha_{i,2}$ avec $(i > 2)$ si $\alpha_{2,2} = 0$ etc...

Exercice 2
Faire les différentes étapes décrites ci-dessus pour :

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

En déduire le déterminant de A.

1.2 Factorisation LU

Lorsqu'il est inutile de faire des échanges de lignes pour choisir le pivot, la méthode de Gauß nous donne :
$MA = U$ avec $M = M_{n-1}M_{n-2}...M_1$.
On a donc :
$A = LU$ avec $L = M^{-1} = M_1^{-1}...M_{n-1}^{-1}$

Exercice 3
Calculer L en calculant le produit $M_1^{-1}...M_{n-1}^{-1}$ (on remarquera que M_k^{-1} a des 1 sur la diagonale, des zéros ailleurs, sauf à la k ième colonne, où sous la diagonale on trouve : $\frac{a_{j,k}}{a_{k,k}}$ $j = k+1...n$)

2 Réduction des endomorphismes par la méthode traditionnelle

On factorise le polynôme caractéristique $\det(\lambda I - A)$ et on résoud $(\lambda I - A)v = 0$ pour les racines λ trouvées.
Définitions :
matrice caractéristique de A c'est $\lambda I - A$
polynôme caractéristique de A c'est $\det(\lambda I - A)$=$P(\lambda)$
matrice adjointe de A est la matrice des cofacteurs de la transposée de A.
On a donc si $B(\lambda)$ = matrice adjointe de $\lambda I - A$:
$B(\lambda) * (\lambda I - A) = (\lambda I - A) * B(\lambda) = P(\lambda) * I$

3 Réduction exacte des endomorphismes par le polynome minimal

3.1 Calcul du polynôme minimal

Soit A une matrice $n \times n$. Nous allons déterminer le polynôme minimal de A, c'est-à-dire un polynôme non nul de degré minimal qui annule A. On considère pour cela A comme un *vecteur* de l'espace des matrices ayant n^2 coordonnées et on va chercher par la méthode du pivot de Gauß une relation non triviale entre les puissances successives de A (on peut se limiter à I, A, .., A^n car le polynôme caractéristique de A annule A).

Exercice 4
Soit la matrice A définie à l'exercice 2 :
Calculer A^2, A^3, A^4 et appliquer le pivot de Gauß (sans faire LU) sur les 5 lignes constituées par I, A, A^2, A^3, A^4 (si une colonne a des zéros sur et sous la diagonale, on passe à la colonne suivante en cherchant le pivot à partir de la même ligne).
N'oubliez pas de garder une trace des opérations effectuées en notant en bout de lignes l'expression des lignes $(I, A...A^4)$.
Trouver le polynôme minimal de A.

Trouver par la même méthode, le polynôme minimal de la matrice définie par :

$$B = \begin{pmatrix} 1 & 1 & 1 & 3 \\ 1 & 1 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 3 & 1 & 1 & 1 \end{pmatrix}$$

3.2 Recherche de vecteurs propres

On suppose que le polynôme minimal P_A est sans facteur carré (*square-free*) ce qui se vérifie en cherchant le pgcd de P_A avec sa dérivée. À l'aide du polynôme minimal, il est facile de déterminer les vecteurs propres de A correspondant aux racines de P_A que l'on sait déterminer car si $P_A(X) = (X - \lambda_1) \times Q(X)$ on a :

$$\text{Im } Q(A) = \text{Ker } (A - \lambda_1 I)$$

En effet puisque $(A - \lambda_1 I) \times Q(A) = P_A(A) = 0$ on a :

$$\text{Im } Q(A) \subset \text{Ker } (A - \lambda_1 I)$$

$(X - \lambda_1)$ et $Q(X)$ sont premiers entre eux car P_A est sans facteur carré donc d'après Bézout, il existe deux polynômes U et V tels que :
$U(X) \times (X - \lambda_1) + Q(X) \times V(X) = 1$ on a donc :
$U(A) \times (A - \lambda_1 I) + Q(A) \times V(A) = I$ donc :

$$\text{Im } Q(A) \supset \text{Ker } (A - \lambda_1 I)$$

Lorsque le polynôme minimal n'est pas *square-free*, l'identité de Bézout donne les projecteurs sur les *espaces caractéristiques* ($Q(A)$ envoie K^n sur $Ker(A - \lambda_1 I)^{p_1}$ lorsque λ_1 est d'ordre p_1).

Exercice 5

Soit :

$$B = \begin{pmatrix} 1 & 1 & 1 & 3 \\ 1 & 1 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 3 & 1 & 1 & 1 \end{pmatrix}$$

Trouver une base de vecteurs propres en utilisant la méthode ci-dessus.

4 Réduction des endomorphismes

4.1 Calcul du polynôme caractéristique et de la matrice adjointe

4.1.1 Les formules de Newton et méthode de Leverrier

On note P le polynôme caractéristique de A défini par :
$P(\lambda) = \det(\lambda I - A) = (\lambda - \lambda_1)(\lambda - \lambda_2)...(\lambda - \lambda_n) = \lambda^n - p_1\lambda^{n-1} - p_2\lambda^{n-2} ... - p_n$
On pose :
$s_1 = \text{tr}(A) = \Sigma\lambda_i = \Sigma_{i=1}^n a_{i,i}$
$s_k = \text{tr}(A^k) = \Sigma\lambda_i^k \ (k = 1..n)$
On montre alors les formules de Newton :
$p_1 = s_1$

$2.p_2 = s_2 - p_1.s_1$
$k.p_k = s_k - p_1.s_{k-1} - ... - p_{k-1}.s_1$ $(k = 1..n)$
Ces formules permettent de calculer les coefficients p_k $(k = 1..n)$ du polynôme caractéristique de A lorsqu'on a calculé les traces des matrices A $A^2..A^n$.
La matrice adjointe $B(\lambda)$ est un polynôme en λ de degré $n-1$ à coefficients matriciels :
$B(\lambda) = \lambda^{n-1}I + \lambda^{n-2}B_1 + ...B_{n-1}$
puisque $B(\lambda) \times (\lambda I - A) = P(\lambda)I$ on a la relation de récurrence :
$B_0 = I,\ B_k = AB_{k-1} - p_kI$
donc $B_k = A^k - p_1A^{k-1} - p_2A^{k-2}... - p_kI$

4.1.2 La méthode de Faddeev

La méthode de Faddeev permet le calcul simultané des coefficients p_i $(i = 1..n)$ du polynôme caractéristique $P(\lambda) = \det(\lambda I - A)$ et des coefficients matriciels B_i $(i = 1..n-1)$ du polynôme en λ donnant la matrice adjointe.
Au lieu de calculer les traces des puissaces de A, Faddeev calcule les traces des matrices A_i $(i = 1..n)$ définies par :
$A_1 = A,\ p_1 = \mathrm{tr}(A),\ B_1 = A_1 - p_1I$
$A_2 = AB_1,\ p_2 = \frac{1}{2}\mathrm{tr}(A_2),\ B_2 = A_2 - p_2I$
$A_k = AB_{k-1},\ p_k = \frac{1}{k}\mathrm{tr}(A_k),\ B_k = A_k - p_kI$
On montre facilement que :
$B_n = A_n - p_nI = 0$
les nombres $p_1, p_2...p_n$ et les matrices $B_1, B_2, ...B_n$ ainsi définis sont bien les coefficients de $P(\lambda)$ et de $B(\lambda)$ puisque on retrouve les formules de Newton en prenant les traces des égalités : $A_k = A^k - p_1A^{k-1} - ... - p_{k-1}A$
$B_k = A^k - p_1A^{k-1} - ... - p_{k-1}A - p_kI$

4.1.3 Démonstration de la méthode de Faddeev

Théorème : $P'(\lambda)$, la dérivée du polynôme caractéristique, est égal à la trace de $B(\lambda)$ ($B(\lambda)$ est la matrice adjointe de $\lambda I - A$ et on note $b_{i,j}(\lambda)$ ses coefficients).
Démonstration :
si on note $V_1(\lambda), ...V_n(\lambda)$ les vecteurs colonnes de $\lambda I - A$, on a : $\det(\lambda I - A) -$
$\det(\lambda_0 I - A) = \det(V_1(\lambda) - V_1(\lambda_0), V_2(\lambda), , ..., V_n(\lambda))+$
$\det(V_1(\lambda_0), V_2(\lambda) - V_2(\lambda_0), ..., V_n(\lambda)) + ... + \det(V_1(\lambda_0), V_2(\lambda_0), ..., V_n(\lambda) - V_n(\lambda_0))$
On a donc :
$P'(\lambda_0) = \lim_{\lambda \to \lambda_0} \dfrac{P(\lambda) - P(\lambda_0)}{\lambda - \lambda_0} = \det(V_1'(\lambda_0), V_2(\lambda_0), ..., V_n(\lambda_0)) +$
$\det(V_1(\lambda_0), V_2'(\lambda_0), ..., V_n(\lambda_0)) + ... + \det(V_1(\lambda_0), V_2(\lambda_0), ..., V_n'(\lambda_0))$
Pour finir la démonstration, il suffit de remarquer que :
$V_i'(\lambda_0) = e_i$
$\det(V_1(\lambda_0), V_2(\lambda_0), ..., V_i'(\lambda_0), ..., V_n(\lambda_0)) = b_{i,i}(\lambda_0)$
et donc que $P'(\lambda_0) = \sum_{i=1}^{n} b_{i,i}(\lambda_0) = \mathrm{trace}\ (B(\lambda_0)$

Les relations de récurrence :
On note :
$P(\lambda) = \lambda^n - p_1\lambda^{n-1} - p_2\lambda^{n-2}... - p_n$
$B(\lambda) = \lambda^{n-1}I + \lambda^{n-2}B_1 + ... + B_{n-1}$
Donc :
$P'(\lambda) = \mathrm{trace}\ (B(\lambda)) = \lambda^{n-1}\mathrm{trace}\ (I) + \lambda^{n-2}\mathrm{trace}\ (B_1) + ... + \mathrm{trace}\ (B_{n-1})$
ou encore
$n\lambda^{n-1} - p_1(n-1)\lambda^{n-2} - p_2(n-2)\lambda^{n-3}... - p_{n-1} = \lambda^{n-1}\mathrm{trace}\ (I) + \lambda^{n-2}\mathrm{trace}\ (B_1) +$
$... + \mathrm{trace}\ (B_{n-1})$
ou encore $\mathrm{trace}\ (B_i) = -p_i(n-i)$

Comme $B(\lambda) \times (\lambda I - A) = P(\lambda) I$ on a les relations de récurrence :
$B_0 = I$,.., $B_i = AB_{i-1} - p_i I$
donc trace $(B_i) = -p_i(n-i) =$ trace $(AB_{i-1}) - np_i$
$p_i = \frac{\text{trace } (AB_{i-1})}{i}$
On calcule donc :
$p_1 =$ trace (AB_0) et $B_1 = AB_0 - p_1 I$
$p_2 = \frac{\text{trace } (AB_1)}{2}$ et $B_2 = AB_1 - p_2 I$
....
$p_i = \frac{\text{trace } (AB_{i-1})}{i}$ et $B_i = AB_{i-1} - p_i I$

4.2 Exercice 6

Soit la matrice :

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

Déterminer $p_1, p_2, p_3 p_4$ et B_1, B_2, B_3, B_4 :
- par la méthode de Leverrier
- par la méthode de Faddeev
En déduire le polynôme caractéristique de A et calculer A^{-1}.
Vérifier votre résultat à l'aide des commandes `MuPAD`.

5 Méthode de la puissance et des itérations inverses.

La méthode de la puissance est une méthode numérique qui permet de déterminer la valeur propre de module maximal (en supposant que A possède une seule valeur propre de module maximal). On prend un vecteur colonne v au hasard et on calcule la suite récurrente:

$$v_0 = v \ , \ v_{n+1} = Av_n / ||Av_n||$$

Si la composante de v_0 sur l'espace propre correspondant à la valeur propre de plus grand module n'est pas nulle, v_n tend vers un vecteur (normé) de cet espace propre.
En effet, si la base propre est $w_1, ..., w_n$ (avec $Aw_i = \lambda_i w_i$ avec $|\lambda_1| > |\lambda_2| > ... > |\lambda_n|$) on a :
$v_0 = a_1 w_1 + a_2 w_2 + ... + a_n w_n$ donc
$Av_0 = a_1 \lambda_1 w_1 + a_2 \lambda_2 w_2 + ... + a_n \lambda_n w_n$
et on pose $||Av_0||^{-1} = \mu_1$
$v_1 = \mu_1 Av_0$ donc
$Av_1 = \mu_1 A^2 v_0 = \mu_1 (a_1 \lambda_1^2 w_1 + a_2 \lambda_2^2 w_2 + ... + a_n \lambda_n^2 w_n)$
et on pose $||Av_1||^{-1} = \mu_2$ donc
$Av_2 = \mu_2 \mu_1 (a_1 \lambda_1^2 w_1 + a_2 \lambda_2^2 w_2 + ... + a_n \lambda_n^2 w_n)$
.....
si on pose $||Av_{k-1}||^{-1} = \mu_k$ et $C_k = \mu_k .. \mu_1$ on a :
$v_k = \mu_k Av_{k-1}$ donc
$v_k = \mu_k .. \mu_1 A^k v_0 = \mu_k .. \mu_1 (a_1 \lambda_1^k w_1 + a_2 \lambda_2^k w_2 + ... + a_n \lambda_n^k w_n)$
$v_k = C_k \lambda_1^k (a_1 w_1 + a_2 (\frac{\lambda_2}{\lambda_1})^k w_2 + ... a_n (\frac{\lambda_n}{\lambda_1})^k w_n) \simeq C_k \lambda_1^k a_1 w_1$
On a bien pour k assez grand v_k colinéaire au vecteur propre w_1 donc $Av_k \simeq \lambda_1 v_k$

et puisque v_k est de norme 1, $|\lambda_1| \simeq ||Av_k||$ et si la première coordonnée $(v_k)_1$ de v_k est non nulle, on a $\lambda_1 \simeq \frac{(Av_k)_1}{(v_k)_1}$

Exercice 7 Calcul numérique de la valeur propre de norme maximale par la méthode de la puissance : essayez avec une matrice aléatoire. Pour générer une matrice aléatoire:

- Avec `MuPAD` :
  `r:=random(-9..9); M:=Matrix()(4,4,r());`
  génère une matrice aléatoire dont les coefficients sont compris entre -9 et 9.
- HP49: `RAND({4,4})`, TI89/92: `RandMat(4,4)`.

Remarque : pour trouver les autres valeurs propres/vecteurs propres, il faut pouvoir éliminer la valeur propre trouvée.
On sait en particulier le faire quand A est symétrique car il suffit de remplacer A par $A - \lambda_1 v_k^t v_k$.

La méthode des itérations inverses consiste lorsque l'endomorphisme f est inversible à appliquer l'algorithme de la puissance à f^{-1} en effet:
si λ est une valeur propre de f et si w est un vecteur propre associé à λ on a :
$f(w) = \lambda w \Leftrightarrow f^{-1}w = \frac{1}{\lambda}w$
La méthode des itérations inverses permet donc de trouver la valeur propre de plus petit module (à condition d'avoir inversé la matrice A associée à f).
La méthode des itérations inverses est utile lorsqu'on connait une valeur approchée τ d'une valeur propre λ.
On peut alors améliorer τ en utilisant des itérations inverses, puisqu'alors la matice $B = A - \tau I$ est inversible et possède $\lambda - \tau$ (qui est très petit) comme valeur propre.
On cherche l'inverse de $B = A - \tau I$, puis on pose :
$y_0 = B^{-1}b$ où b est un vecteur aléatoire (y_0 est alors proche d'un vecteur propre correspondant à $\lambda \approx \tau$.
On itère ensuite la procédure.

6 Expérimentation

6.1 MuPAD

Ne pas oublier de taper l'instruction `export(linalg)` dans votre session `MuPAD`.
Quelques instructions d'algèbre linéaire:

1. `charPolynomial(A,x)`: polynôme caractéristique
2. `det(A)`: déterminant d'une matrice A,
3. `scalarProduct(A,B)`: produit scalaire des deux vecteurs A et B,
4. `eigenValues(A)`: valeurs propres de la matrice A. Si vous voulez des valeurs numériques (et si A est à valeurs numériques) écrivez au moins un des coefficients de A comme nombre réel ou complexe.
5. `eigenVectors(A)`: vecteurs propres de A,
6. `linearSolve(A,b)`: résout l'équation linéaire $Ax = b$,
7. `f:=(i,j)->(1/(i+j-1));Dom::Matrix()(2,2,f)`: pour créer une matrice de Hilbert.

8. `(i,j) -> if i=j then 1; else 0; end_if;`: utile pour créer la matrice identité.
9. `ncols(A)`: nombre de colonnes d'une matrice, `tr(A)`: trace de A
10. `transpose(A)`: transposée de A
11. `ludecomp(A)`: donne la décomposition LU de A.
12. `?linalg` pour la liste des commandes d'algèbre linéaire

Exercice 8
Programmation des méthodes de Fadeev et du calcul du polynome minimal : Sauvegardez votre programme dans un fichier texte, par exemple `polmin.mu` puis lancez-le dans une session `MuPAD` en tapant :
`read("polmin.mu")`.

Rappelons qu'une procédure peut être exécutée en pas à pas (debug mode) en lançant le debugger (dans `emacs`, tapez sur la touche `Echap`, puis tapez `x mdx`) puis:
`read("polmin.mu")`
pour lire la procédure dans le fichier `name` et finalement:
`debug(minimal_poly(A))`

6.2 Calculatrices

Les calculatrices formelles Casio ne disposent pas de commande d'algebre linéaire symbolique. On présente ici les commandes pour les HP49 et TI92/89.

6.2.1 Commandes de la HP49

Pour entrer une matrices A vous pouvez utiliser le MatrixWriter (`MTRW` sur le clavier). On peut aussi créer une matrice dont l'élément $m_{i,j}$ est donné par une fonction $(i,j) \to m_{i,j}$ à l'aide de `LC2M`.

Pour les commandes utiles se réferrer à "La HP49G en résumé".

6.2.2 TI92/89

Commandes utiles:

- `identity(n)` et `randmat(n,m)` pour créer des matrices identité et aléatoire
- `rref` : réduction de Gauß
- `egv` (TI89 et 92+ seulement) : valeurs propres/vecteurs propres. Attention, le calcul est numérique uniquement. Sur tous les modèles on peut bien sur calculer le polynome caractéristique avec `det(A-xI)` ou I désigne la matrice identité, puis factoriser avec `factor` et calculer les vecteurs propres à la main en résolvant le système.
- `lu` : factorisation LU.

7 Autres exercices à traiter (bonus)

Calcul d'un noyau/image d'une application linéaire et d'une intersection d'espaces vectoriels: vérifiez un exemple pris dans un de vos TD de Deug 1ère année ou par exemple noyau/image de l'endomorphisme de matrice dans la base canonique:

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

Intersection des sous-espaces vectoriels de R^3 d'équations $x+y-3z=0$ et $2x+y-z=0$.

Résolution de système linéaire à paramètres: comment faire la discussion: reprenez un exemple de vos TD de 1ère année ou résolvez le système:

$$\left\{\begin{array}{rcl} x+ay+a^2z & = & d \\ x+by+b^2z & = & e \\ x+cy+c^2z & = & f \end{array}\right.$$

Factorisation `LU`, vérification expérimentale de la méthode de diagonalisation numérique utilisant LU (prendre des matrices aléatoires, pour la diagonalisation numérique, effectuez quelques itérations de LU UL et comparez avec les instructions de diagonalisation numérique du logiciel. Pour que la stabilité numérique soit meilleure, testez d'abord avec des matrices symétrique définies positives, pour en obtenir une de manière aléatoire, il suffit de faire le produit d'une matrice aléatoire par sa transposée).

Diagonalisation exacte d'endomorphismes par les différentes méthodes proposées: par exemple pour la matrice avec des 1 partout sauf sur l'antidiagonale où on met des m.

Déterminer le polynôme minimal puis, regardez comment se comporte la diagonalisation numérique pour :

$$A=\left(\begin{array}{ccc} 3 & -1 & 1 \\ 2 & 0 & 1 \\ 1 & -1 & 2 \end{array}\right),$$

Cas des matrices symétriques définies positives. Vérifiez qu'on peut choisir une base orthonormale de vecteurs propres.

8 Programmes pour le polynôme minimal.

8.1 Avec `MuPAD`.

Ce qui suit `//` est un commentaire (comme en C++).

```
// illustration du polynome minimal
// utiliser read("minimal_poly") a l'interieur de MuPAD

// fonction iequalj utile pour definir la matrice identite
iequalj:=(l,k)->(if (l=k) then 1; else 0; end_if;);

// definition de la procedure
// en argument, on place la matrice A dont on cherche le
// polynome minimal, la procedure retourne la matrice reduite
// ou on lit le polynome minimal
minimal_poly:=proc(A)
local n,Id,B;                          // variables locales
begin
 n:=ncols(A);                          // dimension de la matrice A

 Id:=Dom::Matrix()(n,n,iequalj);       // matrice identite

 B:=Dom::Matrix()(n+1,n^2+1);          // cree la matrice resultat

// on remplit la 1ere ligne avec les coordonnees de l'identite
```

```
 for i from 1 to n^2 do
   B[1,i]:=Id[((i-1)div n)+1,((i-1) mod n)+1]:
 end_for:

// on remplit les lignes 2 a n+1 avec
// les coordonnees des puissances de A
 for j from 1 to n do
  for i from 1 to n^2 do
    B[j+1,i]:=(A^j)[((i-1)div n)+1,((i-1) mod n)+1]:
  end_for;
 end_for;

// on remplit la derniere colonne avec les puissances de A
 for i from 1 to n+1 do B[i,n^2+1]:=a^(i-1): end_for;

// on reduit la matrice
 gaussElim(B);
end_proc:
```

Pour utiliser cette procedure, tapez par exemple les commandes:

```
export(linalg);
read("minimal_poly");
A:=Dom::Matrix()(2,2,[[1,2],[3,4]]);
minimal_poly(A);
```

8.2 Avec la HP49G mode RPN

Le caractère `@` permet d'introduire un commentaire.

```
<<                              @ A (la matrice A doit etre sur la pile)
  DUP IDN                       @ A A^j (initialise a Id pour j=0)
  DUP SIZE 1 GET                @ A A^j n
  -> A AJ N                     @ (stocke A A^j et n en variables locales)
  <<
     0 N FOR J                  @ boucle J variant de 0 a N
       AJ ARRY-> EVAL * ->LIST  @ cree un vecteur avec les coordonnees de A^j
       'a' J ^ +                @ rajoute en derniere colonne a^j
       A 'AJ' STO*              @ multiplie A^j par A et stocke le resultat
     NEXT                       @ fin de boucle
  >>
  N 1 + ->LIST                  @ cree une matrice a partir des vecteurs
>> 'MINP' STO
```

On stocke ce programme dans une variable (par exemple `'MINP' STO`). Pour exécuter le programme, on place la matrice sur la pile, puis on tape `MINP`. Remarque: il est possible d'exécuter ce programme en mode pas-à-pas en tapant `'MINP'` puis la touche `PRG`, menu `RUN` et menu `DEBUG`.

9 Programmes pour la méthode de Fadeev.

9.1 Avec MuPAD

```
// fonction iequalj utile pour definir la matrice identite
iequalj:=(k,l)->(if (k=l) then 1; else 0; end_if;);
```

```
fadeev:=proc(A)
local Aj,AAj,Id,coef,n,pcara,lmat;
begin
 n:=ncols(A);
 Id:=Dom::Matrix()(n,n,iequalj);  // matrice identite
 Aj:=Id;
 lmat:=[];                        // liste vide
 pcara:=[1];                       // coefficient de plus grand degre
 for j from 1 to n do
  lmat:=append(lmat,Aj):          // rajoute Aj a la liste de matrices
  AAj:=Aj*A:
  coef:=-tr(AAj)/j:
  pcara:=append(pcara,coef):        // rajoute coef au polynome caracteristique
  Aj:=AAj+coef*Id:
 end_for;
 lmat,pcara;                       // resultat
end_proc;
```

Exemple d'utilisation:

```
export(linalg);
read("fadeev");
A:=Dom::Matrix()(2,2,[[1,2],[3,4]]);
res:=fadeev(A);
(res[1])[2]*A;
```

9.2 Avec la HP49G mode RPN

```
<<
  DUP IDN DUP           @ A Id A_j (initialise a Id pour j=0)
  DUP SIZE 1 GET        @ A Id A_j n
  { } { 1 }             @ A Id A_j n lmat pcara (liste des matrices et poly car)
  -> A ID AJ N LMAT PCARA
  <<
     1 N FOR J          @ (boucle J de 1 a N)
       LMAT AJ +        @ lmat
       'LMAT' STO       @ (rajoute A_j a la liste des matrices)
       AJ A *           @ A A*A_j
       DUP TRACE        @ A*A_j trace
       J NEG /          @ A*A_j -trace/j
       PCARA OVER +      @ A*A_j -trace/j pcara
       'PCARA' STO       @ A*A_j -trace/j (ajoute le coefficient a pcara)
       ID * +           @ A_j+1
       'AJ' STO         @ (stocke le resultat)
     NEXT
     LMAT PCARA          @ rappelle lmat et pcara sur la pile
  >>
>> 'FADEEV' STO
```

Pour obtenir le polynome caracteristique sur la pile sous forme symbolique tapez
`X PEVAL`.

10 Méthode de la puissance.

Programme pour effectuer une itération:

```
export(linalg);
M:=Dom::Matrix()(2,2,[[1,2],[3,4]]);
v:=Dom::Matrix()(2,1,[1.2,2.3]);
iter:=proc()
begin
 v:=normalize(M*v);
end_proc;
```

Modifier M et v puis tapez `iter();`.

```
<<
  M SWAP *
  DUP ABS /
>> 'ITER' STO
```

Sur la `49G mode RPN` , on stocke la matrice dans une variable (`'M' STO`) puis on tape un vecteur, puis `ITER` plusieurs fois.

11 Quelques références

- Press et al.:
  Numerical Recipies in Pascal (exite aussi en C)
- Davenport, Siret, Tournier:
  Calcul formel: Systèmes et algorithmes de manipulations algébriques
- Gantmacher:
  Théorie des matrices