L2 Calcul formel Tp 2 : Systèmes d'équations linéaires entières et à paramètres polynomiaux

```
> restart;
```

N'oubliez pas de charger la bibliothèque 'linalg'!

```
> with(linalg):
>
```

Algorithme de réduction des matrices entières et polynomiales

Matrices entières

Tester l'algorithme présenté en cours sur les matrices entières M suivantes. Vous devez donc trouver une suite d'opérations élémentaires sur les lignes et les colonnes pour "diagonaliser" la matrice M.

Les fonctions arithmétiques **irem** et **iquo** pourront vous être utiles.
Dans Maple, les opérations élémentaires sur les lignes et les colonnes sont implémentées dans les fonctions **addrow**/**addcol** et **swaprow**/**swapcol** (en anglais, to swap=échanger).

Pour la suite, il est utile de se souvenir des opérations élémentaires effectuées, ou du moins de trouver des matrices entières inversibles g (comme gauche) et d (comme droite) telles que gMd soit diagonale.
Pour vous aider, je fournis des variantes **opC**/**opL** et **ech** qui construisent les matrices g et d à mesure que vous faîtes vos opérations élémentaires. Leur syntaxe est la suivante:
-**opC(i,r,j)** change Ci en Ci+r*Cj
-**opL(i,r,j)** change Li en Li+r*Lj.
-**ech(i0,j0,i1,j1)** effectue un échange de lignes et un échange de colonnes pour que les coefficients d'indices (i0,j0) et (i1,j1) soient échangés.
En prime, une fonction **annule()** permet de revenir en arrière si l'on s'est trompé.
Lorsque vous avez fini, la fonction **affiche(M)**, vous permet de récupérer les valeurs des matrices g et d construites et de vérifier votre résultat.
Si vous avez besoin de récupérer des coefficients dans la
Si vous le souhaitez, vous pouvez regarder rapidement comment fonctionnent les fonctions suivantes, mais ce n'est pas indispensable.

```
> init:=proc(M)
  local n,p;
  global niveau,sauveG, sauveD, sauveM;
  n:=rowdim(M);
  p:=coldim(M);
  niveau:=1:
  sauveG[0]:=matrix(n,n, (i,j)-> if i=j then 1 else 0 fi):
  sauveD[0]:=matrix(p,p, (i,j)-> if i=j then 1 else 0 fi):
  sauveM[0]:=copy(M):
  end:

> transvL:=proc(i,j,r)
  local T;
  T:=copy(sauveG[0]);
  T[i,j]:=r;
  evalm(T);
  end:

  transvC:=proc(i,j,r)
  local T;
  T:=copy(sauveD[0]);
  T[i,j]:=r;
  evalm(T);
  end:

  permL:=proc(i,j)
  local E;
  E:=copy(sauveG[0]);
  E[i,i]:=0; E[j,j]:=0; E[i,j]:=1; E[j,i]:=1;
  E;
  end:

  permC:=proc(i,j)
  local E;
  E:=copy(sauveD[0]);
  E[i,i]:=0; E[j,j]:=0; E[i,j]:=1; E[j,i]:=1;
  E;
  end:


  opL:=proc(i,r,j)
  local R;
  global niveau,sauveM,sauveG,sauveD,N;
  R:=map(expand,addrow(sauveM[niveau-1],j,i,r));
  sauveM[niveau]:=R;
  sauveD[niveau]:=sauveD[niveau-1];
```

```
sauveG[niveau]:=multiply(transvL(i,j,r),sauveG[niveau-1]);

niveau:=niveau+1;
N:=evalm(R);
end:

opC:=proc(j,r,i)
local R;
global niveau,sauveM,sauveG,sauveD,N;
R:=map(expand,addcol(sauveM[niveau-1],i,j,r));
sauveM[niveau]:=R;
sauveD[niveau]:=multiply(sauveD[niveau-1],transvC(i,j,r));

sauveG[niveau]:=sauveG[niveau-1];
niveau:=niveau+1;
N:=evalm(R);
end:

ech:=proc(i0,j0,i1,j1)
local R;
global niveau,sauveM,sauveG,sauveD,N;
R:=swaprow(swapcol(sauveM[niveau-1],j0,j1),i0,i1);
sauveM[niveau]:=R;
sauveD[niveau]:=multiply(sauveD[niveau-1],permC(j0,j1)) ;
sauveG[niveau]:=multiply(permL(i0,i1),sauveG[niveau-1]) ;
niveau:=niveau+1;
N:=evalm(R);
end:

annule:=proc()
global niveau,N;
niveau:=niveau-1:
N:=evalm(sauveM[niveau-1]);
end:

affiche:=proc(M)
global niveau,g,d;
g:=sauveG[niveau-1];
d:=sauveD[niveau-1];
print('g'=map(sort,map(expand,evalm(g))));
print('d'=map(sort,map(expand,evalm(d))));
print('gMd'=map(sort,map(expand,multiply(g,M,d))));
end:
> M := matrix([[9, -36, 30], [-36, 192, -180], [30, -180,
180]]); init(M):
```

$$M := \begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix}$$

```
>
>
>
>
>
```

```
> M1:=matrix([[41, -58, -90, 53, -1], [94, 83, -86, 23,
  -84], [19, -50, 88, -53, 85]]);
  init(M1):
```

$$M1 := \begin{bmatrix} 41 & -58 & -90 & 53 & -1 \\ 94 & 83 & -86 & 23 & -84 \\ 19 & -50 & 88 & -53 & 85 \end{bmatrix}$$

```
>
```

Comparez vos résultats avec ceux de la fonction `ismith` de Maple.

Matrices polynomiales

Même question, avec des matrices à coefficients polynomiaux.
Attention: la division euclidienne dans les polynômes se fait avec les fonctions `rem` et `quo`.

```
> M3:=matrix([[6*x^8+30*x^7+33*x^6+72*x^5+28*x^4-15*x^3-48*x
  ^2-81*x-23,
  12*x^8+40*x^7+40*x^6+56*x^5+15*x^4-38*x^3-30*x^2-63*x-32,
  2*x^5+6*x^4+3*x^3+2*x^2-5*x-8],
  [21*x^8+31*x^7+70*x^6+75*x^5+14*x^4-46*x^3-91*x^2-61*x-13,
  42*x^8+62*x^7+63*x^6+48*x^5-20*x^4-54*x^3-69*x^2-55*x-17,
  7*x^5+8*x^4+2*x^3-4*x^2-9*x-4],
  [3*x^5+x^4+5*x^3+2*x^2-8*x-3, 6*x^5+2*x^4-x^3+2*x^2-5*x-4,
  x^2-1]]);
  init(M3):
```

```
>
```

```
>
```

Comparez vos résultats avec ceux de la fonction `smith` de Maple.

Dans le temps qu'il vous reste, essayez d'automatiser vos fonctions. L'idéal serait d'arriver à reprogrammer la fonction smith.

Solutions entières de systèmes d'équations entières

Exercice 1: un système à résoudre

Utilisez la méthode présentée en cours pour décrire l'ensemble des solutions entières du système suivant (vous avez le droit d'utiliser la fonction ismith).

```
> syst:={-50*x-76*y-79*z+23*t+40*u=0,-92*x-16*y+32*z+34*t-34
  *
  u=0,4*x-72*y+12*z+92*t-74*u=0};
```

La commande Maple permettant de résoudre des équations diophantiennes (non nécessairement linéaires) s'appelle isolve. Vérifiez que votre résultat coïncide avec celui de Maple.

Exercice 2: base d'un réseau

Paramétrer l'ensemble des combinaisons linéaires **rationnelles** des vecteurs v1, v2, v3 qui sont entières.

```
> v1:= vector([2,2/3,1,1/2,1]);
  v2:= vector([3,1/2,0,1/4,1]);
  v3:= vector([2,1,1,4/3,1/3]);
```

Par exemple, le vecteur w=1/2*v1+1/3*v2+1/2*v3 est une combinaison linéaire rationnelle qui est entière.

```
> w:=evalm(1/2*v1+1/3*v2+1/2*v3);
```

Exercice 3: réactions chimiques

Déterminer une base de l'ensemble des réactions chimiques possibles entre les molécules suivantes:

CH4, CO, CO2, O2, H2O, N2, H2, NH3, N2O5.

```
> Mol:=[C+4*H, C+O, C+2*O, 2*O,2*H+O,2*N,2*H,N+3*H,
  2*N+5*O];
> At:=[C,H,O,N];
```

Systèmes à paramètres

Exercice 1: rang d'une matrice à paramètres

Déterminer en fonctions des valeurs du paramètre α le rang de la matrice suivante:

```
> M:=matrix(6,6,(i,j)-> cos((2*i+j)*alpha)):
```

Comparer avec la réponse de Maple.

```
> rank(M);
```

Exercice 2: systèmes d'équations

Trouver l'ensemble des polynômes A, B et C en l'indéterminée x qui vérifient l'équation suivante:

A*P1+ B*P2+C*P3=1, où P1, P2 et P3 sont les polynômes suivants:

```
> P1:=(x^6-x^5+x^4-x^3+x^2-x+1)*x*(x-1);
> P2:=(x^8-x^6+x^4-x^2+1)*x*(x+1);
> P3:=(x^12-x^11+x^9-x^8+x^6-x^4+x^3-x+1)*(x^2-1);
```

Remarque subsidiaire sur ismith

Comparer le résultat de l'algorithme présenté en cours et celui de la fonction ismith sur les matrices suivantes:

```
> M1:=diag(2,3,5);
  M2:=diag(2,3,4);
> N1:=ismith(M1,U1,V1);
  N2:=ismith(M2,U2,V2);
```

Voyez-vous une famille d'opérations élémentaires sur les lignes et les colonnes qui permette de passer de votre résultat à celui de Maple?
A quelle condition sur le vecteur entier B=vector([b1,b2,b3]) le système M1 &*X =B admet-il des solutions entières?
Comment reformuler ce résultat en utilisant la matrice N1?
Explication?