```
> restart: with(linalg): with(LinearAlgebra):
```

```
Warning, the protected names norm and trace have been redefined and
unprotected

Warning, the assigned name GramSchmidt now has a global binding
```

```
> # Decodage d'une code de Reed-Solomon
> # sur l'alphabet F_8, de longueur 4 et dimension 2.
>
> # Le polynome x^3+x+1 est irreductible sur F_2 et on a
  F_8=F_2[x]/(x^3+x+1).
> # la racine 'a' de ce polynome engendre F_8 sur F_2,
> # et 1,a,a^2 est une base de F_8 en tant que F_2 espace vectoriel.
> alias(a=RootOf(x^3+x+1)):
>
> # On prends 4 elements x1,x2,x3,x4 de F_8 non nuls.
> x1:=1; x2:=1+a; x3:=1+a+a^2; x4:=1+a^2;
```

$$x1 := 1$$

$$x2 := 1 + a$$

$$x3 := 1 + a + a^2$$

$$x4 := 1 + a^2$$

```
> # Le message envoye est m=(1,1), que represente les coefficients
> # d'un polynome f de F_8[x] degre 1
> # On peut reemplacer m par une couple quelconque des elements de F_8.
> f:=1+x;
```

$$f := 1 + x$$

```
> # Le mot recu est (y1,y2,y3,y4) \in F_8.
> # On a un erreur a niveau F_8 (y3 \ne x3).
> # Le code peut corriger cet erreur, car (n-k)/2=1.
> # Si on veut changer le message m, il faut aussi changer le mot recu,
> # a un mot a distance 1 du mot code c(m).
> y1:=eval(f,x=x1) mod 2; y2:=eval(f,x=x2) mod 2; y3:=a; y4:=eval(f,x=x4)
  mod 2;
```

$$y1 := 0$$

$$y2 := a$$

$$y3 := a$$

$$y4 := a^2$$

```
> # Matrice de Berlekamp-Massey dans la forme non reduite.
> BM:=matrix(4,5,[[1,x1,x1^2,y1,y1*x1], [1,x2,x2^2,y2,y2*x2],
  [1,x3,x3^2,y3,y3*x3], [1,x4,x4^2,y4,y4*x4]]);
```

$$BM := \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1+a & (1+a)^2 & a & a(1+a) \\ 1 & 1+a+a^2 & (1+a+a^2)^2 & a & a(1+a+a^2) \\ 1 & 1+a^2 & (1+a^2)^2 & a^2 & a^2(1+a^2) \end{bmatrix}$$

```
> # Matrice dans la forme reduite modulo x^3+x+1 et modulo 2.
> mod2:=x->x mod 2:
> BM:=map(mod2, map(simplify, evalm(BM)));
```

$$BM := \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1+a & 1+a^2 & a & a(1+a) \\ 1 & 1+a+a^2 & 1+a & a & 1+a^2 \\ 1 & 1+a^2 & 1+a+a^2 & a^2 & a \end{bmatrix}$$

```
> # On fait d'abord l'algo de Berlekamp-Massey
> # en calculant la decomposition LU via l'algo de Gauss.
> LUdecomp(map(mod2,BM),L='l',U='u'):
> LGauss:=map(mod2, map(simplify,evalm(l))); UGauss:=map(mod2,
  map(simplify,evalm(u)));
```

$$LGauss := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1+a & 1 & 0 \\ 1 & a & 1 & 1 \end{bmatrix}$$

$$UGauss := \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & a & a^2 & a & a\,(1+a) \\ 0 & 0 & 1+a^2 & a^2 & a^2 \\ 0 & 0 & 0 & a^2 & 1 \end{bmatrix}$$

```
> # Verification de la decomposition BM=LU
> map(mod2, map(simplify,evalm(BM-l&*u)));
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> # Calcul du noyau (a la base, c'est la resolution d'un systeme
  triangulaire)
> K:=map(mod2,map(simplify,convert((op(kernel(u))),Matrix)));
```

$$K := \begin{bmatrix} 1+a+a^2 & a^2+a & 1 & 1+a+a^2 & 1 \end{bmatrix}$$

```
> # Construction des polynomes de Berlekamp-Massey
> g:= K[1,1] + K[1,2]*x+K[1,3]*x^2; h:=K[1,4]+K[1,5]*x;
```

$$g := 1 + a + a^2 + (a^2 + a)\,x + x^2$$

$$h := 1 + a + a^2 + x$$

```
> # Reconstruction du message f !!!
> Normal(-g/h) mod 2;
```

$$1 + x$$

```
>
> # Par la suite on reprends le calcul de la decomposition LU
> # cette fois-ci via l'algo de deplacement
>
> # La 1ere matrice de deplacement (diagonale inversible de taille 4).
> S:=map(mod2, map(simplify,diag(1/x1,1/x2,1/x3,1/x4)));
```

$$S := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a^2 + a & 0 & 0 \\ 0 & 0 & a^2 & 0 \\ 0 & 0 & 0 & a \end{bmatrix}$$

```
> # La 2eme matrice de deplacement (bloc de Jordan de taille 5).
> T:=JordanBlock(0,5);
```

$$T := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> # Verification : on calcule le deplacement de BM par rapport au couple
  (S,T),
> # on verifie que le rang de deplacement est bien 2
> N1:=map(mod2, map(simplify,map(mod2, evalm(S&*BM-BM&*T))));
```

$$N1 := \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ a\,(1 + a) & 0 & 0 & a & 0 \\ a^2 & 0 & 0 & 0 & 0 \\ a & 0 & 0 & a^2 & 0 \end{bmatrix}$$

```
> # On construit des generateurs de N1.
> # Atention! Si on veut faire un algo performant, on n'as pas besoin de
  construire BM.
> # Il suffit de construire les colonnes numero 1, 3, et 4, cela suffit
  pour
> # trouver les colonnes non nules de N1, et donc pour construire les
  generateurs G1, B1 ci-dessus.
> # Dans le cas d'un Berlekamp-Massey general, il suffira de construire 3
  des colonnes de la matrice BM.
> G1:=matrix(4,2,[[N1[1,1],N1[1,4]],[N1[2,1],N1[2,4]],[N1[3,1],N1[3,4]],[N
  1[4,1],N1[4,4]]]);
```

$$G1 := \begin{bmatrix} 1 & 1 \\ a\,(1+a) & a \\ a^2 & 0 \\ a & a^2 \end{bmatrix}$$

```
> B1:=matrix(2,5,[[1,0,0,0,0],[0,0,0,1,0]]);
```

$$B1 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> # Juste une verification (pas necessaire) pour voir si on a les bons
  generateurs.
> evalm(G1&*B1);
```

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ a\,(1+a) & 0 & 0 & a & 0 \\ a^2 & 0 & 0 & 0 & 0 \\ a & 0 & 0 & a^2 & 0 \end{bmatrix}$$

```
> # On construit progressivement la matrice U, ligne par ligne.
> # Pour l'instant, elle n'a que des zeros.
> U:=Matrix(4,5);
```

$$U := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> # On remplit la 1ere ligne avec la 1ere ligne de BM.
> U[1,1]:=BM[1,1]; U[1,2]:=BM[1,2]; U[1,3]:=BM[1,3]; U[1,4]:=BM[1,4];
  U[1,5]:=BM[1,5];
```

$$U_{1,1} := 1$$

$$U_{1,2} := 1$$

$$U_{1,3} := 1$$

$$U_{1,4} := 0$$

$$U_{1,5} := 0$$

```
> # C'est maintenant que l'algorithme commence vraiment.
```

```
> # On initialise A1:=BM
> A1:=evalm(BM):
> # (d'ailleurs il nous suffit avec la 1ere ligne et la 1ere colonne de
  A1)
> # puis on calcule des generateurs pour le complement de Schur A2.
> G2:=map(mod2, map(simplify, evalm(submatrix(G1,2..4,
  1..2)-submatrix(A1,2..4,1..1)&*submatrix(G1, 1..1, 1..2)/A1[1,1])));
```

$$G2 := \begin{bmatrix} 1 + a + a^2 & 1 + a \\ 1 + a^2 & 1 \\ 1 + a & 1 + a^2 \end{bmatrix}$$

```
> B2:=map(mod2, map(simplify, evalm(submatrix(B1,1..2,
  2..5)-submatrix(B1,1..2,1..1)&*submatrix(A1, 1..1, 2..5)/A1[1,1])));
```

$$B2 := \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> # Par la suite on calcule le deplacement du complement de Schur A2.
> # Bien qu'on calcule tous les coefficients, on n'a pas besoin que de la
  1ere ligne et la 1ere colonne,
> # cela devrait se tenir en compte dans une implementation efficace.
> N2:=map(mod2, map(simplify, evalm(G2&*B2)));
```

$$N2 := \begin{bmatrix} 1 + a + a^2 & 1 + a + a^2 & 1 + a & 0 \\ 1 + a^2 & 1 + a^2 & 1 & 0 \\ 1 + a & 1 + a & 1 + a^2 & 0 \end{bmatrix}$$

```
> # On construit la 1ere ligne et la 1ere colonne de A2, a partir de
  celles de N2
> A2:=Matrix(3,4):
> A2[1,1]:=Normal(N2[1,1]*x2) mod 2; A2[1,2]:=Normal((N2[1,2]+A2[1,1])*x2)
  mod 2; A2[1,3]:=Normal((N2[1,3]+A2[1,2])*x2) mod 2;
  A2[1,4]:=Normal((N2[1,4]+A2[1,3])*x2) mod 2;

  A2[2,1]:=Normal(N2[2,1]*x3) mod 2; A2[3,1]:=Normal(N2[3,1]*x4) mod 2;
```

$$A2_{1,1} := a$$

$$A2_{1,2} := a^2$$

$$A2_{1,3} := a$$

$$A2_{1,4} := a^2 + a$$

$$A2_{2,1} := a^2 + a$$

$$A2_{3,1} := a^2$$

```
> # Visualisons ce qu'on a trouve...
```

```
> evalm(A2);
```

$$\begin{bmatrix} a & a^2 & a & a^2 + a \\ a^2 + a & 0 & 0 & 0 \\ a^2 & 0 & 0 & 0 \end{bmatrix}$$

```
> # On retrouve la 2eme ligne de U: c'est la 1ere ligne du complement de
  Schur A2.
```

```
> U[2,2]:=A2[1,1]; U[2,3]:=A2[1,2]; U[2,4]:=A2[1,3]; U[2,5]:=A2[1,4];
```

$$U_{2,2} := a$$

$$U_{2,3} := a^2$$

$$U_{2,4} := a$$

$$U_{2,5} := a^2 + a$$

```
> # Visualisons U jusqu'a present:
```

```
> evalm(U);
```

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & a & a^2 & a & a^2 + a \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> # Et c'est reparti!
```

```
> G3:=map(mod2, map(simplify, evalm(submatrix(G2,2..3,
  1..2)-submatrix(A2,2..3,1..1)&*submatrix(G2, 1..1, 1..2)/A2[1,1])));
```

$$G3 := \begin{bmatrix} 1 + a + a^2 & a^2 \\ a^2 + a & 1 + a \end{bmatrix}$$

```
> B3:=map(mod2, map(simplify, evalm(submatrix(B2,1..2,
  2..4)-submatrix(B2,1..2,1..1)&*submatrix(A2, 1..1, 2..4)/A2[1,1])));
```

$$B3 := \begin{bmatrix} 1+a & 1 & 1+a \\ 0 & 1 & 0 \end{bmatrix}$$

```
> N3:=map(mod2, map(simplify, evalm(G3&*B3)));
```

$$N3 := \begin{bmatrix} a & 1+a & a \\ 1 & 1+a^2 & 1 \end{bmatrix}$$

```
> A3:=Matrix(2,3):
```

```
> A3[1,1]:=Normal(N3[1,1]*x3) mod 2; A3[1,2]:=Normal((N3[1,2]+A3[1,1])*x3)
  mod 2; A3[1,3]:=Normal((N3[1,3]+A3[1,2])*x3) mod 2;
  A3[2,1]:=Normal(N3[2,1]*x4) mod 2;
```

$$A3_{1,1} := 1 + a^2$$

$$A3_{1,2} := a^2$$

$$A3_{1,3} := a^2$$

$$A3_{2,1} := 1 + a^2$$

```
> evalm(A3);
```

$$\begin{bmatrix} 1+a^2 & a^2 & a^2 \\ 1+a^2 & 0 & 0 \end{bmatrix}$$

```
> U[3,3]:=A3[1,1]; U[3,4]:=A3[1,2]; U[3,5]:=A3[1,3];
```

$$U_{3,3} := 1 + a^2$$

$$U_{3,4} := a^2$$

$$U_{3,5} := a^2$$

```
> G4:=map(mod2, map(simplify, evalm(submatrix(G3,2..2,
  1..2)-submatrix(A3,2..2,1..1)&*submatrix(G3, 1..1, 1..2)/A3[1,1])));
```

$$G4 := [1 \quad 1 + a + a^2]$$

```
> B4:=map(mod2, map(simplify, evalm(submatrix(B3,1..2,
  2..3)-submatrix(B3,1..2,1..1)&*submatrix(A3, 1..1, 2..3)/A3[1,1])));
```

$$B4 := \begin{bmatrix} a^2 & a\,(1+a) \\ 1 & 0 \end{bmatrix}$$

```
> N4:=map(mod2, map(simplify, evalm(G4&*B4)));
```

$$N4 := [1 + a \quad a\,(1 + a)]$$

```
> A4:=Matrix(1,2):
```

```
> A4[1,1]:=Normal(N4[1,1]*x4) mod 2; A4[1,2]:=Normal((N4[1,2]+A4[1,1])*x4)
  mod 2;
```

$$A4_{1,1} := a^2$$

$$A4_{1,2} := 1$$

```
> evalm(A4);
```

$$[a^2 \quad 1]$$

```
> U[4,4]:=A4[1,1]; U[4,5]:=A4[1,2];
```

$$U_{4,4} := a^2$$

$$U_{4,5} := 1$$

```
> # Et finalement, le resultat !
```

```
> # On verifie que cela coincide bien avec le calcul fait precedemment
  avec l'algo de Gauss.
```

```
> evalm(U);
```

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & a & a^2 & a & a^2 + a \\ 0 & 0 & 1 + a^2 & a^2 & a^2 \\ 0 & 0 & 0 & a^2 & 1 \end{bmatrix}$$

```
>
```