```
>  interface(screenwidth=120);
>  with(LinearAlgebra):
>  # Construction de l'exemple: on veut une reponse de ce type:
>  f:=(i,j)->if(i=j-1) then 1 else 0 fi;
              f := proc(i, j) option operator, arrow; if i = j - 1 then 1 else 0 end if end proc

>  J:=Matrix(8,8,f):J[3,4]:=0:J[6,7]:=0:J;
                  [0    1    0    0    0    0    0    0]
                  [0    0    1    0    0    0    0    0]
                  [0    0    0    0    0    0    0    0]
                  [0    0    0    0    1    0    0    0]
                  [0    0    0    0    0    1    0    0]
                  [0    0    0    0    0    0    0    0]
                  [0    0    0    0    0    0    0    1]
                  [0    0    0    0    0    0    0    0]

>  #On fait un changement de base simple:
>  f:=(i,j)->if(i=j) then 1 else 0 fi;
              f := proc(i, j) option operator, arrow; if i = j then 1 else 0 end if end proc

>  T:=proc(i,j,a)
>  A:=Matrix(8,8,f):A[i,j]:=a;A;
>  end proc;
Warning, 'A' is implicitly declared local to procedure 'T'
              T := proc(i, j, a) local A; A := Matrix(8, 8, f); A[i, j] := a; A end proc

>  # faire Li<-Li+aLj c'est multiplier a gauche par T(i,j,a).
>  # PAr exemple L3<- L3+aL2 c'est multiplier a gauche par:
>  T(3,2,a);
                  [1    0    0    0    0    0    0    0]
                  [0    1    0    0    0    0    0    0]
                  [0    a    1    0    0    0    0    0]
                  [0    0    0    1    0    0    0    0]
                  [0    0    0    0    1    0    0    0]
                  [0    0    0    0    0    1    0    0]
                  [0    0    0    0    0    0    1    0]
                  [0    0    0    0    0    0    0    1]

>  # Remarquer que l'inverse de T(i,j,a) est T(i,j,-a)
>  T(3,2,a)^(-1);
                  [1    0     0    0    0    0    0    0]
                  [0    1     0    0    0    0    0    0]
                  [0    -a    1    0    0    0    0    0]
                  [0    0     0    1    0    0    0    0]
                  [0    0     0    0    1    0    0    0]
                  [0    0     0    0    0    1    0    0]
                  [0    0     0    0    0    0    1    0]
                  [0    0     0    0    0    0    0    1]

>  # Donc conjuguer par T(i,j,a) c'est faire:
>  # Li <- Li+aLj et Cj <- Cj-aCi
>  P:=T(6,7,2).T(4,5,1).T(3,2,2).T(1,2,1):
>  P,P^(-1);
       [1    1    0    0    0    0    0    0]  [1    -1    0    0     0    0     0    0]
       [0    1    0    0    0    0    0    0]  [0     1    0    0     0    0     0    0]
       [0    2    1    0    0    0    0    0]  [0    -2    1    0     0    0     0    0]
       [0    0    0    1    1    0    0    0]  [0     0    0    1    -1    0     0    0]
       [0    0    0    0    1    0    0    0], [0     0    0    0     1    0     0    0]
       [0    0    0    0    0    1    2    0]  [0     0    0    0     0    1    -2    0]
       [0    0    0    0    0    0    1    0]  [0     0    0    0     0    0     1    0]
       [0    0    0    0    0    0    0    1]  [0     0    0    0     0    0     0    1]

>  # Donc faire a l'ordinateur:
>  N:=P.J.P^(-1):
>  # est identique a faire a la main a partir de J:
>  # L1 <- L1+L2 ; C2<-C2 - C1 puis
>  # L3 <- L3+2L2 ; C2 <- C2 -2C3
>  # L4 <- L4+L5 ; C5 <- C5 -C4
>  # L6 <- L6 +2 L7; C7 <- C7 -2 C6
>  ####################################################################
>  # On a maintenant trouve une bel exercice: Trouver la forme de    #
>  # jordan de N et une matrice de passage pour l'obtenir.           #
>  ####################################################################
>  # On calcule N^2 et son noyau.
```

```
>  N,N^2;
       [0    -1    1    0    0    0     0    0]  [0    -2    1    0    0    0     0    0]
       [0    -2    1    0    0    0     0    0]  [0     0    0    0    0    0     0    0]
       [0    -4    2    0    0    0     0    0]  [0     0    0    0    0    0     0    0]
       [0     0    0    0    1    1    -2    0]  [0     0    0    0    0    1    -2    0]
       [0     0    0    0    0    1    -2    0], [0     0    0    0    0    0     0    0]
       [0     0    0    0    0    0     0    2]  [0     0    0    0    0    0     0    0]
       [0     0    0    0    0    0     0    1]  [0     0    0    0    0    0     0    0]
       [0     0    0    0    0    0     0    0]  [0     0    0    0    0    0     0    0]

>  N2:=NullSpace(N^2);
                       [0]  [0]  [0]  [0]  [ 0 ]  [1]
                       [0]  [0]  [0]  [0]  [1/2]  [0]
                       [0]  [0]  [0]  [0]  [ 1 ]  [0]
                       [0]  [0]  [0]  [1]  [ 0 ]  [0]
               N2 := {[0], [0], [1], [0], [ 0 ], [0]}
                       [0]  [2]  [0]  [0]  [ 0 ]  [0]
                       [0]  [1]  [0]  [0]  [ 0 ]  [0]
                       [1]  [0]  [0]  [0]  [ 0 ]  [0]

>  #on choisit a et b  independants modulo ker N^2
>  a:=Vector([0,0,1,0,0,0,0,0]):b:=Vector([0,0,0,0,0,1,0,0]):
>  Rank(Matrix([op(N2),a,b]));
                                 8

>  N1:=NullSpace(N);
                        [1]  [0]  [0]
                        [0]  [0]  [0]
                        [0]  [0]  [0]
                        [0]  [1]  [0]
               N1 := {[0], [0], [0]}
                        [0]  [0]  [2]
                        [0]  [0]  [1]
                        [0]  [0]  [0]

>  #dim ker N^2 -dim ker N= 6-3=2+1 donc N.a,N.b doit etre complete par c
>  # tq N.a,N.b,c indep modulo ker N. Par exemple on prend celui la:
>  c:=Vector([0,0,0,0,0,0,0,1]):
>  #On verifie qu'il convient:
>  Rank(Matrix([op(N1),N.a,N.b,c]));
bytes used=4000100, alloc=3472772, time=0.28
                                 6

>  # dim ker N - dim ker N^0=3 c'est donc engendr{\'e} par
>  # N^2.a,N^2.b,N.c. Il n'y a plus rien a faire, et l'on prend
>  # la base suivante:
>  Q:=Matrix([(N^2).a,N.a,a,(N^2).b,N.b,b,N.c,c]);
                  [1    1    0    0    0    0    0    0]
                  [0    1    0    0    0    0    0    0]
                  [0    2    1    0    0    0    0    0]
                  [0    0    0    1    1    0    0    0]
             Q := [0    0    0    0    1    0    0    0]
                  [0    0    0    0    0    1    2    0]
                  [0    0    0    0    0    0    1    0]
                  [0    0    0    0    0    0    0    1]

>  #On sait maintenant que Q^(-1).N.Q doit donner J. verification:
>  Q^(-1).N.Q;
                  [0    1    0    0    0    0    0    0]
                  [0    0    1    0    0    0    0    0]
                  [0    0    0    0    0    0    0    0]
                  [0    0    0    0    1    0    0    0]
                  [0    0    0    0    0    1    0    0]
                  [0    0    0    0    0    0    0    0]
                  [0    0    0    0    0    0    0    1]
                  [0    0    0    0    0    0    0    0]
```