```
def coef_non_nul_en_tete(D,P,Q):
    # prend une matrice non nulle (un triplet, en fait...)
    # renvoie une matrice equivalente avec un coef non nul en [0,0]
    n=D.nrows();
    p=D.ncols();
    i=0;
    j=0;
    while i<n and D[i,j]==0:
        j=0;
        while j<p and D[i,j]==0:
            j=j+1;
            if j==p: i=i+1;
    D.swap_rows(0,i); P.swap_rows(0,i);
    D.swap_columns(0,j); Q.swap_columns(0,j);
    return (D,P,Q)
```

```
def position_premier_coef_non_multiple(D):
    # prend une matrice de format (n,p) avec coef D[0,0] non nul
    # renvoie un couple d'entiers (i,j) tq :
    # (i,j)= + petit (pour l'ordre lex) couple d'entiers tq D[i,j] non multiple de D[0,0] s'il y en a,
    # i=n si tous les coefs de D sont multiples de D[0,0]
    n=D.nrows();
    p=D.ncols();
    a=D[0,0];
    i=0;
    j=0;
    while D[i,j]%a==0:
        j=j+1;
        if j==p:
            i=i+1; j=0;
        if i==n: break;
    return (i,j)
```

```
def pgcd_en_tete(D,P,Q):
    # prend une matrice non nulle
    # renvoie une matrice dont le pgcd des coefficients est place en [0,0]
    n=D.nrows();
    (D,P,Q)=coef_non_nul_en_tete(D,P,Q);
    (i,j)=position_premier_coef_non_multiple(D);
    while i<n:
        if i>0:
            D=D.add_multiple_of_row(0,i,1); P.add_multiple_of_row(0,i,1);
        q=D[0,j]//D[0,0];
        D.add_mutiple_of_column(j,0,-q);
        Q.add_mutiple_of_column(j,0,-q);
        D.swap_columns(0,j);
        Q.swap_columns(0,j);
        (i,j)=position_premier_coef_non_multiple(D);
    return (D,P,Q)
```

```
def annule_premiere_ligne(D,Q):
    # prend une matrice non nulle dont le pgcd des coefficients est place en [0,0]
    # et annule sa premiere ligne
    p=D.ncols();
    if p>1:
        for j in [1..p-1]:
            q=D[0,j]//D[0,0];
            if q<>0: D.add_multiple_of_column(j,0,-q); Q.add_multiple_of_column(j,0,-q);
    return (D,Q)
```

```
def annule_premiere_colonne(D,P):
    # prend une matrice non nulle dont le pgcd des coefficients est place en [0,0]
    # et annule sa premiere colonne
    n=D.nrows();
    if n>1:
        for i in [1..n-1]:
            q=D[i,0]//D[0,0];
            if q<>0: D.add_multiple_of_row(i,0,-q); P.add_multiple_of_row(i,0,-q);
    return (D,P)
```

```
def sur_matrices(D,P,Q,a):
    # prend une matrice de format (n-1,p-1) (un quintuplet...) et un coef a
    # renvoie les matrices bloc-diagonales de format (n,p) obtenues en ajoutant
    # un bloc (1,1) en NO (=en haut à gauche) égal à (a) pour D et (1) pour les 4 autres
    D=block_diagonal_matrix(matrix(1,1,a),D,subdivide=False);
    P=block_diagonal_matrix(matrix(1,1,1),P,subdivide=False);
    Q=block_diagonal_matrix(matrix(1,1,1),Q,subdivide=False);
    return (D,P,Q)
```

```
def FI(D,P,Q):
    # prend 3 matrices
    # renvoie sa forme de Smith D et les matrices de passage P,Q
    n=D.nrows();
    p=D.ncols();
    if D==0:
        return (D,P,Q);
    else:
        (D,P,Q)=pgcd_en_tete(D,P,Q);
        (D,Q)=annule_premiere_ligne(D,Q);
        (D,P)=annule_premiere_colonne(D,P);
        a=D[0,0];
        P1=matrix(n-1,n-1,1);
        Q1=matrix(p-1,p-1,1);
        D1=D.submatrix(1,1,n-1,p-1);
        (D1,P1,Q1)=FI(D1,P1,Q1);
        (D,P2,Q2)=sur_matrices(D1,P1,Q1,a);
        P=P2*P; Q=Q*Q2;
        return (D,P,Q)
```

```
def facteurs_invariants(M):
    # prend une matrice M
    # renvoie D sous forme de Smith et P,Q qui comptabilisent
    # les opérations lignes-colonnes effectuées pendant le processus
    n=M.nrows();
    p=M.ncols();
    D=copy(M);
    P=matrix(n,n,1);
    Q=matrix(p,p,1);
    (D,P,Q)=FI(D,P,Q);
    return (D,P,Q), P*M*Q
```

```
B=matrix(4,4,range(16)); B; facteurs_invariants(B)

[ 0  1  2  3]
[ 4  5  6  7]
[ 8  9 10 11]
[12 13 14 15]

[1 0 0 0]    [ 1  0  0  0]     [ 0  1  1  2]      [1 0 0 0]
[0 4 0 0]    [-5  1  0  0]     [ 1  0 -2 -3]      [0 4 0 0]
[0 0 0 0]    [ 1 -2  1  0]     [ 0  0  1  0]      [0 0 0 0]
[0 0 0 0]    [ 2 -3  0  1]     [ 0  0  0  1]      [0 0 0 0]
```