```
LIB "linalg.lib";
ring r = 3,z,lp;              // F_3[z]

// Realisierung von G:

matrix A[2][2] = 0,1,-1,0;     matrix C[2][2] = -1,1,-1,1;    matrix x[1][2];
matrix B[2][1] = 0,1;          matrix D[2][1] = 0,0;          matrix x1,u,y;

list Z;                       // Liste aller Zustaende, d.h. Z = F_3^2

x =  0, 0; Z[1] = x;           x =  1, 0; Z[4] = x;           x = -1, 0; Z[7] = x;
x =  0, 1; Z[2] = x;           x =  1, 1; Z[5] = x;           x = -1, 1; Z[8] = x;
x =  0,-1; Z[3] = x;           x =  1,-1; Z[6] = x;           x = -1,-1; Z[9] = x;

int i,j,k,wt;
list ZG;
ring rz = (0,ew),w,dp;        // Q(ew)[w]
minpoly = ew^2 + ew + 1;      // mit ew^3 = 1
list Z = imap(r,Z);
matrix L[size(Z)][size(Z)];
setring r;
for (i=1; i<=size(Z); i++)    // durchlaufe mit x jeden Zustand
{
  x = Z[i];
  for (j=1; j<=size(Z); j++)  // durchlaufe mit u jeden Input
  {                           // (hier ist Z auch Menge der Inputs)
    u = Z[j];
    x1 = x*A + u*C;           // berechne Zustand x1 = x_{t+1}
    y  = x*B + u*D;           // berechne Output y
    // beschreibe Zustandsgraphen durch Kanten und Labels
    ZG[size(ZG)+1] = list(x,x1,u,y);
    // erstelle Gewichtsadjazenzmatrix
    k = 1;
    while (k <= size(Z))      // suche Zustand x1 in Z
    {
      if (x1 == Z[k])
      {
        wt = (u[1,1]<>0) + (u[1,2]<>0) + (y[1,1]<>0); // Gewicht von (u,y)
        setring rz;
        L[i,k] = L[i,k] + w^(wt);
        setring r;
        break;
      }
      k++;
    }
  }
}

// Der Zustandsgraph
for (i=1; i<=size(ZG); i++)   // x --> x1 : u|y
{
  print(string(ZG[i][1]) + " --> " + string(ZG[i][2]) + " : "
        + string(ZG[i][3]) + "|" + string(ZG[i][4]));
}
```

```
 ==> 0,0 --> 0,0 : 0,0|0        ==> 1,0 --> 0,1 : 0,0|0        ==> -1,0 --> 0,-1 : 0,0|0
 ==> 0,0 --> -1,1 : 0,1|0       ==> 1,0 --> -1,-1 : 0,1|0      ==> -1,0 --> -1,0 : 0,1|0
 ==> 0,0 --> 1,-1 : 0,-1|0      ==> 1,0 --> 1,0 : 0,-1|0       ==> -1,0 --> 1,1 : 0,-1|0
 ==> 0,0 --> -1,1 : 1,0|0       ==> 1,0 --> -1,-1 : 1,0|0      ==> -1,0 --> -1,0 : 1,0|0
 ==> 0,0 --> 1,-1 : 1,1|0       ==> 1,0 --> 1,0 : 1,1|0        ==> -1,0 --> 1,1 : 1,1|0
 ==> 0,0 --> 0,0 : 1,-1|0       ==> 1,0 --> 0,1 : 1,-1|0       ==> -1,0 --> 0,-1 : 1,-1|0
 ==> 0,0 --> 1,-1 : -1,0|0      ==> 1,0 --> 1,0 : -1,0|0       ==> -1,0 --> 1,1 : -1,0|0
 ==> 0,0 --> 0,0 : -1,1|0       ==> 1,0 --> 0,1 : -1,1|0       ==> -1,0 --> 0,-1 : -1,1|0
 ==> 0,0 --> -1,1 : -1,-1|0     ==> 1,0 --> -1,-1 : -1,-1|0    ==> -1,0 --> -1,0 : -1,-1|0
 ==> 0,1 --> -1,0 : 0,0|1       ==> 1,1 --> -1,1 : 0,0|1       ==> -1,1 --> -1,-1 : 0,0|1
 ==> 0,1 --> 1,1 : 0,1|1        ==> 1,1 --> 1,-1 : 0,1|1       ==> -1,1 --> 1,0 : 0,1|1
 ==> 0,1 --> 0,-1 : 0,-1|1      ==> 1,1 --> 0,0 : 0,-1|1       ==> -1,1 --> 0,1 : 0,-1|1
 ==> 0,1 --> 1,1 : 1,0|1        ==> 1,1 --> 1,-1 : 1,0|1       ==> -1,1 --> 1,0 : 1,0|1
 ==> 0,1 --> 0,-1 : 1,1|1       ==> 1,1 --> 0,0 : 1,1|1        ==> -1,1 --> 0,1 : 1,1|1
 ==> 0,1 --> -1,0 : 1,-1|1      ==> 1,1 --> -1,1 : 1,-1|1      ==> -1,1 --> -1,-1 : 1,-1|1
 ==> 0,1 --> 0,-1 : -1,0|1      ==> 1,1 --> 0,0 : -1,0|1       ==> -1,1 --> 0,1 : -1,0|1
 ==> 0,1 --> -1,0 : -1,1|1      ==> 1,1 --> -1,1 : -1,1|1      ==> -1,1 --> -1,-1 : -1,1|1
 ==> 0,1 --> 1,1 : -1,-1|1      ==> 1,1 --> 1,-1 : -1,-1|1     ==> -1,1 --> 1,0 : -1,-1|1
 ==> 0,-1 --> 1,0 : 0,0|-1      ==> 1,-1 --> 1,1 : 0,0|-1      ==> -1,-1 --> 1,-1 : 0,0|-1
 ==> 0,-1 --> 0,1 : 0,1|-1      ==> 1,-1 --> 0,-1 : 0,1|-1     ==> -1,-1 --> 0,0 : 0,1|-1
 ==> 0,-1 --> -1,-1 : 0,-1|-1   ==> 1,-1 --> -1,0 : 0,-1|-1    ==> -1,-1 --> -1,1 : 0,-1|-1
 ==> 0,-1 --> 0,1 : 1,0|-1      ==> 1,-1 --> 0,-1 : 1,0|-1     ==> -1,-1 --> 0,0 : 1,0|-1
 ==> 0,-1 --> -1,-1 : 1,1|-1    ==> 1,-1 --> -1,0 : 1,1|-1     ==> -1,-1 --> -1,1 : 1,1|-1
 ==> 0,-1 --> 1,0 : 1,-1|-1     ==> 1,-1 --> 1,1 : 1,-1|-1     ==> -1,-1 --> 1,-1 : 1,-1|-1
 ==> 0,-1 --> -1,-1 : -1,0|-1   ==> 1,-1 --> -1,0 : -1,0|-1    ==> -1,-1 --> -1,1 : -1,0|-1
 ==> 0,-1 --> 1,0 : -1,1|-1     ==> 1,-1 --> 1,1 : -1,1|-1     ==> -1,-1 --> 1,-1 : -1,1|-1
 ==> 0,-1 --> 0,1 : -1,-1|-1    ==> 1,-1 --> 0,-1 : -1,-1|-1   ==> -1,-1 --> 0,0 : -1,-1|-1
```

```
// Jetzt zum dualen Code
module G = [z2+1,z-1, 0],            // Erzeugermatrix des Ursprungscodes
           [1,   0,  -1];
print(transpose(syz(transpose(G)))); // Erzeugermatrix des dualen Codes
 ==> -z+1,z2+1,-z+1

setring rz;
matrix Ld[size(Z)][size(Z)];
setring r;
list ZGd;
for (i=1; i<=size(Z); i++)    // durchlaufe mit x jeden Zustand
{
  x = Z[i];
  for (j=-1; j<2; j++)        // durchlaufe mit y jeden Input
  {                           // (hier ist F_3 die Menge der Inputs)
    y = j;
    x1 = x*transpose(A) - y*transpose(B);   // berechne Zustand x1 = x_{t+1}
    u  = x*transpose(C) - y*transpose(D);   // berechne Output u
    // -- beschreibe Zustandsgraphen durch Ecken und Kanten und Labels
    ZGd[size(ZGd)+1] = list(x,x1,u,y);
    // erstelle Gewichtsadjazenzmatrix
    k = 1;
    while (k <= size(Z))      // suche Zustand x1 in Z
    {
      if (x1 == Z[k])
      {
        wt = (y[1,1]<>0) + (u[1,1]<>0) + (u[1,2]<>0); // Gewicht von (y,u)
        setring rz;
        Ld[i,k] = Ld[i,k] + w^(wt);
        setring r;
```

```
        break;
      }
      k++;
    }
  }
}

// Der Zustandsgraph des dualen Codes
for (i=1; i<=size(ZGd); i++)  // x --> x1 : y|u
{
  print(string(ZGd[i][1]) + " --> " + string(ZGd[i][2]) + " : "
        + string(ZGd[i][3]) + "|" + string(ZGd[i][4]));
}
```

```
 ==> 0,0 --> 0,1 : 0,0|-1       ==> 1,0 --> 0,0 : -1,-1|-1     ==> -1,0 --> 0,-1 : 1,1|-1
 ==> 0,0 --> 0,0 : 0,0|0        ==> 1,0 --> 0,-1 : -1,-1|0     ==> -1,0 --> 0,1 : 1,1|0
 ==> 0,0 --> 0,-1 : 0,0|1       ==> 1,0 --> 0,1 : -1,-1|1      ==> -1,0 --> 0,0 : 1,1|1
 ==> 0,1 --> 1,1 : 1,1|-1       ==> 1,1 --> 1,0 : 0,0|-1       ==> -1,1 --> 1,-1 : -1,-1|-1
 ==> 0,1 --> 1,0 : 1,1|0        ==> 1,1 --> 1,-1 : 0,0|0       ==> -1,1 --> 1,1 : -1,-1|0
 ==> 0,1 --> 1,-1 : 1,1|1       ==> 1,1 --> 1,1 : 0,0|1        ==> -1,1 --> 1,0 : -1,-1|1
 ==> 0,-1 --> -1,1 : -1,-1|-1   ==> 1,-1 --> -1,0 : 1,1|-1     ==> -1,-1 --> -1,-1 : 0,0|-1
 ==> 0,-1 --> -1,0 : -1,-1|0    ==> 1,-1 --> -1,-1 : 1,1|0     ==> -1,-1 --> -1,1 : 0,0|0
 ==> 0,-1 --> -1,-1 : -1,-1|1   ==> 1,-1 --> -1,1 : 1,1|1      ==> -1,-1 --> -1,0 : 0,0|1
```

```
// Jetzt zur MacWilliams-Ideantitaet

// 1. Die Gewichtsadjazenzmatrizen L und Ld
setring rz;
print(L);
 ==> 2*w^2+1,0,      0,      0,      0,      w^2+2*w,0,      w^2+2*w,0,
     0,      0,      L[2,3], 0,      L[2,5], 0,      2*w^3+w,0,      0,
     0,      L[3,2], 0,      2*w^3+w,0,      0,      0,      0,      L[3,9],
     0,      2*w^2+1,0,      w^2+2*w,0,      0,      0,      0,      w^2+2*w,
     L[5,1], 0,      0,      0,      0,      L[5,6], 0,      2*w^3+w,0,
     0,      0,      L[6,3], 0,      2*w^3+w,0,      L[6,7], 0,      0,
     0,      0,      2*w^2+1,0,      w^2+2*w,0,      w^2+2*w,0,      0,
     0,      L[8,2], 0,      L[8,4], 0,      0,      0,      0,      2*w^3+w,
     L[9,1], 0,      0,      0,      0,      2*w^3+w,0,      L[9,8], 0
L[9,8];  // alle Eintraege, die nicht oben angezeigt werden, entsprechen diesem
 ===> w^3+2*w^2

print(Ld);
 ==> 1,  w,  w,  0,  0,  0,  0,  0,  0,
     0,  0,  0,  w^2,w^3,w^3,0,  0,  0,
     0,  0,  0,  0,  0,  0,  w^2,w^3,w^3,
     w^3,w^3,w^2,0,  0,  0,  0,  0,  0,
     0,  0,  0,  w,  w,  1,  0,  0,  0,
     0,  0,  0,  0,  0,  0,  w^3,w^3,w^2,
     w^3,w^2,w^3,0,  0,  0,  0,  0,  0,
     0,  0,  0,  w^3,w^2,w^3,0,  0,  0,
     0,  0,  0,  0,  0,  0,  w,  1,  w

// 2. Die Fouriermatrix F
matrix F[size(Z)][size(Z)];
for (i=1; i<=size(Z); i++)
{
  for (j=1; j<=size(Z); j++)
```

```
  {
    F[i,j] = ew^( int( (Z[i]*transpose(Z[j]))[1,1]) );
  }
}
print(F);
 ==> 1,1,      1,      1,      1,      1,      1,      1,      1,
     1,(ew),   (-ew-1),1,      (ew),   (-ew-1),1,      (ew),   (-ew-1),
     1,(-ew-1),(ew),   1,      (-ew-1),(ew),   1,      (-ew-1),(ew),
     1,1,      1,      (ew),   (ew),   (ew),   (-ew-1),(-ew-1),(-ew-1),
     1,(ew),   (-ew-1),(ew),   (-ew-1),1,      (-ew-1),1,      (ew),
     1,(-ew-1),(ew),   (ew),   1,      (-ew-1),(-ew-1),(ew),   1,
     1,1,      1,      (-ew-1),(-ew-1),(-ew-1),(ew),   (ew),   (ew),
     1,(ew),   (-ew-1),(-ew-1),1,      (ew),   (ew),   (-ew-1),1,
     1,(-ew-1),(ew),   (-ew-1),(ew),   1,      (ew),   1,      (-ew-1)

// 3. Das Bild M von L unter p(w) |-> (1+2w)^3 * p( (1-w)/(1+2w) )
matrix M[size(Z)][size(Z)];
poly p,g;
for (i=1; i<=size(Z); i++)
{
  for (j=1; j<=size(Z); j++)
  {
    p = L[i,j];
    g = 0;
    while (p<>0) // Abb. ist kein Hom, daher monomweises Vorgehen
    {
      g = g + (1+2w)^(3-deg(p)) * subst(lead(p),w,1-w);
      p = p - lead(p);
    }
    M[i,j] = g;
  }
}
print(M);
 ==> M[1,1],0,     0,     0,     0,     M[1,6],0,     M[1,8],0,
     0,     0,     M[2,3],0,     M[2,5],0,     M[2,7],0,     0,
     0,     M[3,2],0,     M[3,4],0,     0,     0,     0,     M[3,9],
     0,     M[4,2],0,     M[4,4],0,     0,     0,     0,     M[4,9],
     M[5,1],0,     0,     0,     0,     M[5,6],0,     M[5,8],0,
     0,     0,     M[6,3],0,     M[6,5],0,     M[6,7],0,     0,
     0,     0,     M[7,3],0,     M[7,5],0,     M[7,7],0,     0,
     0,     M[8,2],0,     M[8,4],0,     0,     0,     0,     M[8,9],
     M[9,1],0,     0,     0,     0,     M[9,6],0,     M[9,8],0
// alle Eintraege ungleich 0 entsprechen einem der Folgenden

M[1,1];                 M[1,6];                 M[2,3];                 M[2,7];
 ==> 12*w3+6*w2+6*w+3;   ==> -6*w3-3*w2+6*w+3    ==> 3*w3-3*w2-3*w+3     ==> -6*w3+6*w2-3*w+3


// 4. Die MacWilliams-Identitaet
matrix T = inverse(F)*transpose(M)*F;
print(T-9*Ld);
 ==> 0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0
```