```
// Homogeneous Buchberger
// Proceed deg by deg - order the pairs deg by deg
// NR lives the deg  unchanged
// SPoly raise the deg

/*
Example
Use R::=QQ[x,y];
PairLCM([2x^2y-x,3xy^2-x]);
//x^2y^2
*/
Define PairLCM(L)
  Return LCM(LT(L[1]),LT(L[2]));
EndDefine;
/*
Example
Use R::=QQ[x,y,z],Lex;
F:=x-y^3;
G:=y^2-z^4;
DegPlusOrd(F,G);
//True
F:=x-y^3;
G:=y-z^3;
DegPlusOrd(F,G);
//False
*/
Define DegPlusOrd(F,G)
  If Deg(F)<Deg(G) Then
    Return True;
  EndIf;
  Return LT(F)<LT(G);
EndDefine;

Define PairOrdDeg(P1,P2)
  D1:=Deg(PairLCM(P1));
  D2:=Deg(PairLCM(P2));
  If D1<D2 Then
    Return True;
  ElIf D1>D2 Then
    Return False;
  Else
    Return PairLCM(P1)<PairLCM(P2)
  EndIf;
EndDefine;
```

```
Define PairOrdSilly(P1,P2)
  Return PairLCM(P1)>PairLCM(P2)
EndDefine;

/*
Use R::=QQ[t,x,y,z];
L:=[t^3-x^3,t^5-y^5,t^7-z^7];
EqSet(Buchberger(L),ReducedGBasis(Ideal(L)));
*/
Define Buchberger(L)
  Pairs:=Flatten([[[L[I],L[J]]|J In (I+1)..Len(L)]|I In 1..Len(L)],1);
  While Pairs <> [] Do
    Print [Len(Pairs)];
    CurrentPair:=Head(Pairs);
    Pairs:=Tail(Pairs);
    SP := SPoly(CurrentPair[1],CurrentPair[2]);
    SP := NR(SP,L);// Reduce current pair
    If SP <> 0 Then
      Pairs:=Concat(Pairs,[[F,SP]|F In L]);
      SortBy(Pairs,Function("PairOrdSilly"));// Keep Pairs in deg+ord order
      Append(L,SP);
      Print "*";
    Else
      Print "0";
    EndIf;
  EndWhile;
  Return Interreduced(Monic(L));// Return the reduced GB
EndDefine;


////////////////////////////////////////////////////////////////
// Minimal generators



// Idea: exploit the homogeneous structure.
// Mark the original gens, use them last in the degree
// If they survive the reduction, they are minimal


// Example [F,G,H]
Use R::=QQ[t,x,y,z];

F:=x^2y-z^3;
G:=x^2z-y^3;
H:=-x^5z + xy^2z^3 - ty^3 + tx^2z;
```

```
SPoly(F,G);

NR(-x^4z + y^2z^3,[F,G]);

K:=-x^4z + y^2z^3;

NR(H,[F,G,K]);// H IS NOT MINIMAL
//0


// Idea: Proceed degree by degree
//       Generators are special pairs [Gen,0]
//       Consider special paris last in the degree
//       If they are not killed by what came before, they are minimal


Deg 3 Start
            Pairs:=[ [F,0],[G,0],[H,0] ];
                        3     3     6
            GB:=[];

    NR(F,[]); // SPECIAL PAIR
    //x^2y - z^3 // F is is useful AND MINIMAL

    MG:=[F];
    GB:=[F];
    Pairs:=[ [G,0],[H,0] ];

    NR(G,[F]);// SPECIAL PAIR
            //-y^3 + x^2z // G is is useful AND MINIMAL

    MG:=[F,G];
            GB:=[F,G];
    Pairs:=[ [F,G], [H,0] ];


Deg 4 Start
             Pairs:=[ [F,G], [H,0] ];
     //          4      6
             GB:=[F,G];


     NR(SPoly(F,G),[F,G]);// SPoly(F,G) is useful NOT MINIMAL
             K:=-x^4z + y^2z^3;
```

```
     PairLCM([F,K])=x^4yz;

             PairLCM([G,K])=x^4y^3z;


             Pairs:=[ [F,K], [H,0],[G,K]];
             //          6      6    8
     GB:=[F,G,K];

Deg 6
             NR(SPoly(F,K),[F,G,K]); // SPoly(F,K) is useless
             //0
     Pairs:=[ [H,0],[G,K]];

     NR(H,[F,G,K]); // H IS NOT MINIMAL
     //0

// Stop, no more candidates to be minimal
//

PrintLn [F,G], " Are minimal generators of ",Ideal(F,G,H);




/*
Example
Use R::=QQ[x,y];
PairLCM([2x^2y-x,3xy^2-x]);
//x^2y^2
Use R::=QQ[x,y];
PairLCM([2x^2y-x,0]);
//x^2y
*/
Define PairLCM(P)
  If P[2]=0 Then
    Return LT(P[1]);
  EndIf;
  Return LCM(LT(P[1]),LT(P[2]));
EndDefine;


/*
Example
Use R::=QQ[x,y];
```

```
F:=2x^2y-x;
G:=3xy^2-x;
SPolyPair([F,G]);
//2x^2 - 3xy
SPolyPair([F,0]);
//2x^2y - x
*/
Define SPolyPair(P)
  If P[2]=0 Then
    Return P[1];
  EndIf;
  A := LCM(LT(P[1]),LT(P[2]));
  Return LC(P[2])*(A/LT(P[1]))*P[1] - LC(P[1])*(A/LT(P[2]))*P[2];
EndDefine;

Define PairOrdSpecial(P1,P2)
  // Deg First
  D1:=Deg(PairLCM(P1));
  D2:=Deg(PairLCM(P2));
  If D1<D2 Then
    Return True;
  ElIf D1>D2 Then
    Return False;
  EndIf;
  // Then original first
  If P1[2]=0 Then
    If P2[2]<>0 Then
      Return False;
    Else //P2[2]=0, both initial, use term ordering
      Return PairLCM(P1)<PairLCM(P2);
    EndIf;
  EndIf;
  If P2[2]=0 Then // And hence P1[2]<>0
    Return True;
  EndIf;
  //Then term ordering
  Return PairLCM(P1)<PairLCM(P2);
EndDefine;


/*
Example
Use R::=QQ[t,x,y,z];
L:=[t^3-x^3,t^5-y^5,t^7-z^7];
Ideal(L)=Ideal(MinGens(L));
//True
```

```
L:=[t^3-x^3,t^5-y^5,SPoly(t^3-x^3,t^5-y^5),t^7-z^7];
Ideal(L)=Ideal(MinGens(L));
Minimalized(Ideal(L))=Ideal(MinGens(L));

L:=ReducedGBasis(Ideal(L));
MinGens(L);
Ideal(MinGens(L))=Ideal(L);
Len(Minimalized(Ideal(L)))=Len(Ideal(MinGens(L)));
*/
Define MinGens(L)
  NumCandidates:=Len(L);
  G:=[];
  MG:=[];
  Pairs:=[[P,0]|P In L];
  While Pairs <> [] And NumCandidates>0 Do
    Print [Len(Pairs)];
    CurrentPair:=Head(Pairs);
    Pairs:=Tail(Pairs);
    If CurrentPair[2]=0 Then
      NumCandidates:=NumCandidates-1;
    EndIf;
    SP := SPolyPair(CurrentPair);
    SP := NR(SP,G);// Reduce current pair
    If SP <> 0 Then
      Pairs:=Concat(Pairs,[[F,SP]|F In G]);
      SortBy(Pairs,Function("PairOrdSpecial"));// Keep Pairs in deg+ord order
      Append(G,SP);
      Print "*";
      If CurrentPair[2]=0 Then
        PrintLn SP;
        Append(MG,SP);// Or Append(CurrentPair[1],SP); if you prefer
Print " MG Found"
      EndIf;
    Else
      Print "0";
    EndIf;
  EndWhile;
  PrintLn;
  Return MG;// Return the reduced GB
EndDefine;




Use R::=QQ[t,x,y,z],PosTo;
```

```
I:=Ideal(xy^2, t^4, xz^3, x^4);

////// The Res command

RI:=Res(R/I);RI;
0 --> R(-13)
  --> R(-9) + R(-10)^2 + R(-11)
  --> R(-6)^2 + R(-7)^2 + R(-8)^2
  --> R(-3) + R(-4)^3
  --> R

Describe(RI);


Mat([
  [xy^2, x^4, xz^3, t^4]
])
Mat([
  [z^3, x^3, 0, t^4, 0, 0],
  [0, -y^2, z^3, 0, 0, t^4],
  [-y^2, 0, -x^3, 0, t^4, 0],
  [0, 0, 0, -xy^2, -xz^3, -x^4]
])
Mat([
  [x^3, t^4, 0, 0],
  [-z^3, 0, t^4, 0],
  [-y^2, 0, 0, t^4],
  [0, -z^3, -x^3, 0],
  [0, y^2, 0, x^3],
  [0, 0, y^2, -z^3]
])
Mat([
  [t^4],
  [-x^3],
  [z^3],
  [y^2]
])



Transposed(Mat([
  [z^3, x^3, 0, t^4, 0, 0],
  [0, -y^2, z^3, 0, 0, t^4],
  [-y^2, 0, -x^3, 0, t^4, 0],
  [0, 0, 0, -xy^2, -xz^3, -x^4]
```

```
]));


Transposed(Mat([
  [x^3, t^4, 0, 0],
  [-z^3, 0, t^4, 0],
  [-y^2, 0, 0, t^4],
  [0, -z^3, -x^3, 0],
  [0, y^2, 0, x^3],
  [0, 0, y^2, -z^3]
]));



M1:=Transposed(Mat([
  [z^3, x^3, 0, t^4, 0, 0],
  [0, -y^2, z^3, 0, 0, t^4],
  [-y^2, 0, -x^3, 0, t^4, 0],
  [0, 0, 0, -xy^2, -xz^3, -x^4]
]));

M2:=Transposed(Mat([
  [x^3, t^4, 0, 0],
  [-z^3, 0, t^4, 0],
  [-y^2, 0, 0, t^4],
  [0, -z^3, -x^3, 0],
  [0, y^2, 0, x^3],
  [0, 0, y^2, -z^3]
]));

M2*M1;

Untagged(RI);

["R", [[Module([[xy^2], [t^4], [xz^3], [x^4]]), [1, 0]],
       [Module([[xy^2], [t^4], [xz^3], [x^4]]), [1, -3], [3, -4]],
       [Module([Shifts([xy^2, x^4, xz^3, t^4]), [z^3, 0, -y^2, 0], [x^3, -y^2, 0, 0], [0
       [Module([Shifts([xy^2z^3, x^4y^2, x^4z^3, t^4xy^2, t^4xz^3, t^4x^4]), [x^3, -z^3,
       [Module([Shifts([x^4y^2z^3, t^4xy^2z^3, t^4x^4y^2, t^4x^4z^3]), [t^4, -x^3, z^3, y

// Exercise: produce the appropriate shifts for all the modules of a free
// minimal resolution of the ideal I:=Ideal(xy^2, t^4, xz^3, x^4);
// to be homogeneous.

////////////// Eigenvalues computations
```

```
Use R::=QQ[x];

A:=Mat([[3,-2,-1],[4,-2,-2],[4,-2,-6]]);
A-x*Identity(3);

P:=Det(It);
Factor(P);

RealRoots(P);// One real eigenvalue

RealRoots(P, 10^(-5));

DecimalStr(-3171043119/536870912);
DecimalStr(-6342086237/1073741824);

DecimalStr(-3171043119/536870912,20);
DecimalStr(-6342086237/1073741824,20);

Subst(P,[[x,-3171043119/536870912]]);
DecimalStr(It,20);


////////////////////////////////////////////////////////////////////////////
// Castelnuovo regularity



//////// Hilbert function

// HF(R/I)=HF(R/LT(I))


-- This exact sequence is the core of many algorithms for
-- computing the Hilbert function of P/I (I monomial Ideal)
--       0 ----> P/(I:(F)) --(*F)--> P/I --(pr)--> P/(I+(F)) ----> 0

-- Standard grading:
-- Using the notation HNum(J)/HDen(P) for the HilbertPoincare series of P/J
-- where HDen(P) = (1-t)^NumIndets(P)
-- we have that
--   HNum(I) = HNum(I+(F)) + t^deg(F) * HNum(I:(F))
-- If HPS(R/I)=a_0+a_1x^1+..+a_kx^k and Dim(R/I)=d
-- Hilbert Function: Sum(i=0,k) a_iTruncatedBIN(x-i+d-1,d-1)
-- Hilbert Polynomial Sum(i=0,k) a_iBIN(x-i+d-1,d-1)
-- Regularity: k-d+1
```

```
/*
Computes the numerator of HPS
Example
Use R::=ZZ/(32003)[x,y,z],Lex;
L:=[x^3y^2,x^2y^3,x^2y^2z^2];
HPSN(L);
-x^8 + 2x^7 - 2x^5 + 1
*/
Define HPSN(L);
  If Len(L)=1 Then
    Return 1-Indet(1)^Deg(L[1]);
  Else
    Return HPSN(Tail(L))-Indet(1)^Deg(Head(L))*
           HPSN(Gens(Ideal(Tail(L)):Ideal(Head(L))));
  EndIf
EndDefine;

/*
Computes HPS
Example
Use R::=ZZ/(32003)[x,y,z],Lex;
L:=[x^3y^2,x^2y^3,x^2y^2z^2];
HPS(L);
(x^7 - x^6 - x^5 + x^4 + x^3 + x^2 + x + 1)/(x^2 - 2x + 1)
*/
Define HPS(L)
  Return HPSN(L)/(1-Indet(1))^NumIndets();
EndDefine;



Use R::=ZZ/(32003)[x,y,z],Lex;
L:=[x^3y^2,x^2y^3,x^2y^2z^2];
I:=Ideal(L);
HPS(Gens(I));
HS:=HilbertSeries(R/I);HS;


Describe(It);

Untagged(Poincare(R/I));

$hp.Den(HS);
$hp.Num(HS);
```

```
$hp.DenToPoly(HS);
$hp.NumToPoly(HS);

HilbertFn(R/I);
RegularityIndex(HilbertFn(R/I));
HilbertPoly(R/I);

// Check
Use R::=ZZ/(32003)[x[1..5]];
I:=Ideal([Product([x[J]^Rand(0,20)|J In 1..4])|I In 1..6]);
HilbertSeries(R/I);
HPS(Gens(I));
P:=HilbertSeries(R/I);
HPSN(Gens(I))=$hp.NumToPoly(P);

// Timings
Use R::=ZZ/(32003)[x[1..10]];
I:=Ideal([Product([x[J]^Rand(0,20)|J In 1..10])|I In 1..10]);
HilbertSeries(R/I);
Time P:=HilbertSeries(R/I);
Time P1:=HPS(Gens(I));



// Exercise: compute the Hilbert Polynomial of the ideal
//  Use R::=QQ[t,x,y,z];
//  I:=Ideal(t^4-x^3y^2,t^5-x^2y^3,t^6-x^2y^2z^2);

// Multigrading

Use R ::= QQ[t,x,y,z];
I:=Ideal(t^5-x^3y^2,t^5-x^2y^3,t^6-x^2y^2z^2);
LT(I);

WM := Mat([[1,2,1,1]]);
HP.PoincareMultiDeg(R/LT(I),WM);

WM := Mat([[1,7,1,1],[1,-5,1,1]]);
HP.PoincareMultiDeg(R/LT(I),WM);

// Multigraded HPS may be huge
Use R::=ZZ/(32003)[x[1..10]];
I:=Ideal([Product([x[J]^Rand(0,20)|J In 1..10])|I In 1..100]);
WM := Mat([[1,2,1,1,2,3,4,5,6,7]]);
HP.PoincareMultiDeg(R/LT(I),WM);
Size(It);
```

```
WM := Mat([[1,2,1,1,2,3,4,5,6,7],[2,5,1,8,2,1,1,1,1,7]]);
HP.PoincareMultiDeg(R/LT(I),WM);
Size(It);

WM := Mat([[1,2,1,1,2,3,4,5,6,7],
           [2,5,1,8,2,1,1,1,1,7],
   [1,1,0,0,3,1,0,2,2,7]]);
W1:=HP.PoincareMultiDeg(R/LT(I),WM);
Size(It);--667830



////////////// Hilbert Function from Resolution

Use R ::= QQ[t,x,y,z];
I:=Ideal(t^5-x^3y^2,t^5-x^2y^3,t^6-x^2y^2z^2);
LT(I);
P:=Res(R/I);
0 --> R(-12) --> R(-7) + R(-10) + R(-11) --> R(-5)^2 + R(-6) --> R

      -t^12     +t^7   +t^10   +t^11       -2t^5   -t^6         1

$hp.NumToPoly(HilbertSeries(R/I));

// Exercise: write a function that computes the HP series from the resolution



//// Projective Dimension and Depth via Auslander-Buchbsbaum
//// Dim

/*
Use R ::= QQ[x,y,z];
I:=Ideal(x^2,xy,xz,y^3);
Res(R/I);
PD(I);
// 2
PD_ModI(I);
// 3
Depth(I);
// 1
Depth_ModI(I);
//0
Dim(R/I);
```

```
//1
IsCM(I);
// False
*/
Define PD(I)
  W:=Untagged(Res(Ring(Var(RingEnv(I)))/I));
  Return Len(W[2])-2;
EndDefine;
Define PD_ModI(I)
  Return PD(I)+1;
EndDefine;

Define Depth(I)
  Return NumIndets()-PD(I);
EndDefine;
Define Depth_ModI(I)
  Return Depth(I)-1;
EndDefine;

Define IsCM(I)
  Return PD_ModI(I)=Dim(Ring(Var(RingEnv(I)))/I);
EndDefine;




////////////// Castelnuovo-Mumford Regularity
////////////// Betti data



Use R ::= QQ[x,y,z,w];
I := Ideal(xz^2-y^2w, xw-y^2, x^2y+xzw, xy^2, xyz^2);
Res(R/I);
BettiDiagram(I);
Reg(I);

/*
Use R ::= QQ[x,y,z];
I:=Ideal(x^2,xy,xz,y^3);
CMReg(I);
//3
Reg(I);
//3
*/
```

```
Define CMReg(I)
  P:=Res(R/I);
  W:=Untagged(P);
  W:=Tail(W[2]);// The res body
  W:=[Tail(K)|K In W];
  M:=0;
  For I:=1 To Len(W) Do
    LocalMax:=Max([-K[2]|K In W[I]])-I+1;
    M:=Max(M,LocalMax);
  EndFor;
  Return M;
EndDefine;
```