COMPUTATIONAL DETAILS ON THE DISPROOF OF MODULARITY

RALF GERKMANN, MAO SHENG†, AND KANG ZUO

Abstract. The purpose of these notes is to provide the details of the Jacobian ring computations carried out in [1], based on the computer algebra system Magma [2].

1. Finding an appropriate matrix

Our first target is to construct the Jacobian ring associated to a Calabi-Yau $\hat{X}$ as described in section 2 of [1]. This variety $\hat{X}$ is defined by a matrix $A \in \mathbb{C}^{4 \times 8}$ subject to the condition that all $4 \times 4$-minors are non-zero. In practice, we work over the field $\mathbb{Q}$ or $\mathbb{F}_p$, $p$ prime, instead of $\mathbb{C}$. The matrices contained in the hyperelliptic locus were of special importance for us: These are of the form $A := (a_{ij})$ with $a_{ij} = \lambda^j_i$, where $(\lambda_1, ..., \lambda_8)$ denotes a tuple of 8 distinct numbers in $\mathbb{C}$.

The Magma program below produces such an admissible matrix. There are three options:

(i) Work with a user-defined matrix. In this case one sets randmat:=false and hyperell:=false. The line $A:=\text{RMatrixSpace}(K,4,8)!0; \ (\text{marked with } \text{"modify here"})$ has to be replaced by a statement that defines the desired matrix, e.g.

\[
A:=\text{Matrix}(K,[
[ 1, 1, 1, 1, 1, 1, 1, 1],
[ 1, 2, 3, 4, 5, 6, 7, 8],
[ 1, 4, 9, 16, 25, 36, 49, 64],
[ 1, 8, 27, 64,125,216,343,512]]);
\]

In this case the non-degeneracy condition is not checked; it has to be ensured by the user.

(ii) Use a random matrix. Set randmat:=true and hyperell:=false. The program creates a random matrix in $K^{4 \times 8}$ which satisfies the non-degeneracy condition. The parameter randrange specifies the maximal absolute value of the matrix coefficients that are created.

(iii) Work with a user-defined matrix contained in the hyperelliptic locus. In this case put hyperell:=true, the value of randmat is irrelevant. In the

† The second author was supported by the Chinese Science Foundation.
line marked with “modify here” the definition $\mathbf{la} := $ can be replaced by any tuple of 8 distinct elements in $K$.

For our implementation we used Magma version 2.11-2. It was carried out under the operating system Linux on an ordinary personal computer. The following program produces the desired matrix.

```plaintext
randmat := true;
randrange := 10;
hyperell := true;

K := Rationals();
A := RMatrixSpace(K,4,8)!0; // <== modify here

genpos:=not(randmat) and not(hyperell);
while not(genpos) do
    N:=randrange;
    if hyperell then
        la:=[1,2,3,4,5,6,7,8]; // <== modify here
        A:=[];
        for i:=0 to 3 do
            A[i+1]:=[
                for j:=1 to 8 do
                    A[i+1][j]:=la[j]^i;
                end for;
            end for;
    else
        A:=[
            for i:=1 to 4 do
                A[i]:=[
                    for j:=1 to 8 do
                        A[i][j]:=Random(2*N-2)-N+1;
                    end for;
                end for;
        end if;
        A:=Matrix(K,A);
    end if;
    seq:=[1,2,3,4,5,6,7,8];
    set:=SequenceToSet(seq);
    subsets:=Subsets(set,4);
    res:=true;
    for su in subsets do
        seq:=SetToSequence(su);
        B:=[
            for i:=1 to 4 do
                B[i]:=[
                    for j:=1 to 4 do
                        B[i][j]:=A[j][seq[i]];
                    end for;
                end for;
        end for;
```
In the first part of the program a random or hyperelliptic matrix is created according to the user specification. The second part is to check the non-degeneracy condition. This is done in the straightforward way: A loop runs through all 4-element subsets $S \subseteq \{1, ..., 8\}$, creates the submatrix that consists of the columns of $A$ given by the numbers in $S$, and computes the determinant of this submatrix.

2. Construction of the Jacobian ideal

In order to carry out explicit computations in the cohomology of $\tilde{X}$ one replaces the intersection of four quadrics given by the matrix $A$ by a certain toric hypersurface with defining equation

$$F = y_1 f_1 + y_2 f_2 + y_3 f_3 + y_4 f_4 \quad \text{with} \quad f_1, ..., f_4 \in K[x_0, ..., x_7].$$

as outlined in section 4 of [1]. The next step is to define the polynomial ring and the Jacobian ideal associated to this hypersurface. The polynomial ring is created by

```plaintext
PR := PolynomialRing(K, 12, "glex");
xn := [];
for i := 0 to 7 do
    xn[i + 1] := "x" cat IntegerToString(i);
end for;
yn := [];
for i := 1 to 4 do
    yn[i] := "y" cat IntegerToString(i);
end for;
AssignNames(~PR, xn cat yn);
x := function(i)
    return PR.(1 + i);
end function;
y := function(i)
    return PR.(8 + i);
end function;
```

The polynomial ring $PR$ has 12 variables $x_0, ..., x_7, y_1, ..., y_4$ which are accessed by $x(i)$ and $y(j)$ and displayed as $x_i, y_j$, respectively. Now the defining equation of the toric hypersurface can be computed from the coefficient matrix $A$.

```plaintext
f := [];
for i := 1 to 4 do
    f[i] := PR!0;
end for;
```
for \( j := 0 \) to 7 do
   \( f[i] := f[i] + A[i][j+1] \times x(j)^2; \)
end for;
end for;
F := PR!0;
for \( i := 1 \) to 4 do
   F := F + y(i) \times f[i];
end for;
The resulting equation is now stored in the variable \( F \). Finally, we define the Jacobian ideal associated to this equation.

\[
I := [];
for \( i := 1 \) to 12 do
   I := Append(I, Derivative(F, i));
end for;
I := Ideal(I);
I := GroebnerBasis(I);
I := IdealWithFixedBasis(I);

3. Deducing a cohomology basis

In this section we describe how to determine a basis for the cohomology of \( \tilde{X} \) as described in section 4 of [1]. In section 2 we defined an action on the ring \( \text{PR} \) by a certain finite group \( N_1 \). The cohomology of \( \tilde{X} \) is given by the \( N_1 \)-fixed part of this ring modulo the Jacobian ideal. It is a graded \( K \)-algebra

\[
R = \bigoplus_{p=0}^{3} R_p
\]

where the component \( R_p \) is generated by the monomials with total degree \( 2p \) in the variables \( x_0, ..., x_7 \) and total degree \( p \) in \( y_1, ..., y_4 \). It follows that a basis of \( R \) can be computed by the following procedure:

(i) Enumerate all monomials of \( x \)-degree \( 2p \) and \( y \)-degree \( p \).
(ii) Discard all monomials which do not agree with their normal form with respect to the Jacobian ideal. This yields a basis of \( R_p \).
(iii) Discard all monomials which are not fixed by the action of \( N_1 \). The remaining monomials constitute a basis of \( R_p \).

For (i) we define two additional polynomial rings with 8 and 4 indeterminates, respectively.

\[
\text{XR} := \text{PolynomialRing}(K, 8);
\text{YR} := \text{PolynomialRing}(K, 4);
\]

We also provide functions which map the indeterminates of \( \text{XR} \) to \( x_0, ..., x_7 \) and the indeterminates \( \text{YR} \) to \( y_1, ..., y_4 \), respectively. Using these functions, we are now
able to enumerate all monomials of $x$-degree $i$ and $y$-degree $j$ as follows: First, we use the Magma function `MonomialsOfDegree` in order to produce all monomials of degree $i$ in $X_R$ and degree $j$ in $Y_R$. The functions below cast these monomials into elements of $PR$, and the desired monomial set is obtained by multiplying any element of the first set with any element of the second.

```magma
xmon2mon:=function(xmon)
  res:=PR!1;
  for i:=0 to 7 do
    exp:=Degree(xmon,i+1);
    res:=res*(PR.(i+1))^[exp];
  end for;
  return res;
end function;

ymon2mon:=function(ymon)
  res:=PR!1;
  for i:=1 to 4 do
    exp:=Degree(ymon,i);
    res:=res*(PR.(i+8))^[exp];
  end for;
  return res;
end function;
```

Step (ii) is based on the following simple function.

```magma
isbasiselt:=function(mon)
  nf:=NormalForm(mon,I);
  return (nf eq mon);
end function;
```

Finally, step (iii) is carried out by the function below. It makes use of the fact that the group $N_1$ is generated by $\sigma_a$, where $a$ runs through the set

\[
\{e_1 + e_2, e_2 + e_3, ..., e_7 + e_8, e_8 + e_1\}
\]

(see section 4 of [1] for the notation).

```magma
isHinvar:=function(mon)
  cond:=true;
  for i:=0 to 6 do
    deg1:=Degree(mon,i+1);
    deg2:=Degree(mon,i+2);
    cond:=cond and IsEven(deg1+deg2);
  end for;
  deg1:=Degree(mon,1);
  deg2:=Degree(mon,8);
  cond:=cond and IsEven(deg1+deg2);
  return cond;
end function;
```
Based on these functions, the following loop now determines a basis for every component the graded ring. The result is the nested array \texttt{basis} in which \texttt{basis}[p+1] contains the basis elements of \(R_p\), given as elements of the ring \(PR\).

\begin{verbatim}
basis:=[];
for i:=0 to 2 do
    basis[i+1]:=[];
    xdeg:=2*i;ydeg:=i;
    xmons:=MonomialsOfDegree(XR,xdeg);
    ymons:=MonomialsOfDegree(YR,ydeg);
    for xmon in xmons do
        for ymon in ymons do
            mon:=xmon2mon(xmon)*ymon2mon(ymon);
            cond:=isHinvar(mon);
            cond:=cond and isbasiselt(mon);
            if cond then
                basis[i+1]:=Append(basis[i+1],mon);
            end if;
        end for;
    end for;
end for;
basis[4]:=[x(7)^6*y(4)^3];
\end{verbatim}

In order to save time, we define the component \texttt{basis}[p+1] directly instead of using the procedure. The reason is that the space of polynomials of bidegree \((6,3)\) is already quite large, so the computation would take several seconds.

4. THE FIRST CHARACTERISTIC SUBVARIETY

We briefly recall the definition of the characteristic subvariety \(R_1\) from section 3 in [1]. Let \(e_1, ..., e_9\) denote a basis of \(R_1\) and \(e'_1, ..., e'_9\) a basis of \(R_2\). Taking the multiplication map to its dual yields a linear map

\[ \mu^*: R_2^* \rightarrow S^2(R_1^*). \]

Fix a bijection \(\varphi: \{(i,j) \in \mathbb{N}^2 \mid 1 \leq i \leq j \leq 9\} \rightarrow \{1, ..., 45\} \). Furthermore, let \(A \in K^{9 \times 45}\) denote a representation matrix of \(\mu^*\) with respect to \((e'_1)^*, ..., (e'_9)^*\) and \(b_{\varphi(i,j)} := e'_i e'_j\). Then \(R_1\) is the subvariety in \(\mathbb{P}^8\) given by the equations

\[ f_\ell := \sum_{i=1}^{9} \sum_{j=i}^{9} a_{\ell,\varphi(i,j)} z_i z_j \in K[z_1, ..., z_9]. \]

Thus in order to compute the variety, we have to carry out the following steps:

(i) provide functions for the bijection \(\varphi\) and its inverse \(\varphi^{-1}\)
(ii) compute a representation matrix of the multiplication map \(\mu: S^2(R_1) \rightarrow R_2\) with respect to the basis \(\tilde{b}_{\varphi(i,j)} := e_i e_j\) and \(e'_1, ..., e'_9\)
(iii) obtain a representation matrix with respect to \((e'_1)^*, ..., (e'_9)^*\) and \(\tilde{b}_{\varphi(i,j)}^* = (e_i e_j)^*\) by transposition
(iv) deduce matrix \(A\) by base change from \((e_i e_j)^*\) to \(e'_i e'_j\)
(v) write down the defining equations of $R_1$

The bijections $\varphi$ and $\varphi^{-1}$ are provided by the following two functions.

```plaintext
r:=PolynomialRing(GF(2),9);
symm2order:=MonomialsOfDegree(r,2);
symm2order:=SetToSequence(symmetric2order);
Symm2Spc:=VectorSpace(K,#symm2order);

ijpos:=function(i,j)
    if i > j then
        i_:=j; j_:=i;
    else
        i_:=i; j_:=j;
    end if;
    p:=9 - i_ + 1;
    offs:=45 - 1/2*p*(p+1);
    offs:=offs + (j_ - i_) + 1;
    offs:=Integers()!offs;
    return offs;
end function;

pos2ij:=function(pos)
    mon:=symm2order[pos];
    res:=[];
    for i:=1 to 9 do
        if Degree(mon,i) eq 2 then
            return [i,i];
        end if;
        if Degree(mon,i) eq 1 then
            res:=Append(res,i);
        end if;
    end for;
    return res;
end function;
```

In order to carry out (ii) we need a function which computes, for an arbitrary polynomial $g$ in the polynomial ring $PR$, a coordinate vector of length $1+9+9+1 = 20$ of the cohomology class represented by $g$ with respect to basis.

```plaintext
CSpc:=VectorSpace(K,20);
poly2vec:=function(poly)
    res:=CSpc!0;
    nf:=NormalForm(poly,1);
    offs:=0;
    for p:=0 to 3 do
        for i:=1 to #basis[p+1] do
            coe:=MonomialCoefficient(nf,basis[p+1][i]);
            res[offs+i]:=coe;
            offs:=offs+#basis[p+1];
        end for;
        offs:=offs+#basis[p+1];
    end for;
```
The representation matrix is now obtained by computing the products of all pairs of elements in $\text{basis}[2]$. The 45 rows of the matrix are given by the $R_2$-parts of the coordinate vectors of these products.

```plaintext
M:=[];
for l:=1 to 45 do
    M[l]:=[];
    ij:=pos2ij(l); i:=ij[1]; j:=ij[2];
    mon1:=basis[2][i]; mon2:=basis[2][j];
    vec:=poly2vec(mon1*mon2);
    for k:=1 to 9 do
        M[l][k]:=vec[10+k];
    end for;
end for;
M:=Matrix(K,M);
```

The implementation of the steps (iii) and (iv) is straightforward, provided one takes into account that $(e_i e_i)^* = e_i^* e_i^*$ and $(e_i e_j)^* = 2 e_i^* e_j^*$. The final result is stored in the matrix named $C$.

```plaintext
N:=[];
for l:=1 to 45 do
    N[l]:=[];
    ij:=pos2ij(l);
    S:=SequenceToSet(ij);
    if #S eq 2 then
        la:=2;
    else
        la:=1;
    end if;
    for k:=1 to 9 do
        N[l][k]:=la*M[l][k];
    end for;
end for;
N:=Matrix(K,N);
C:=Transpose(N);
```

Finally, the matrix $C$ is used to carry out (v). The resulting projective scheme is stored in the variable $\text{charvar}$.

```plaintext
CSR:=PolynomialRing(K,9);
chareqs:=[];
for l:=1 to 9 do
    f:=CSR!0;
    for k:=1 to 45 do
        ij:=pos2ij(k); i:=ij[1]; j:=ij[2];
        f:=f+M[l][k]*f;
    end for;
    chareqs[l]:=[f];
end for;
```

```plaintext
return res;
end function;
```
mon:=CSR.i*CSR.j;
f:=f+C[l][k]*mon;
end for;
chareqs[l]:=f;
end for;
P8:=ProjectiveSpace(CSR);
charvar:=Scheme(P8,chareqs);

The dimension and arithmetic genus are obtained by the function calls

Dimension(charvar) and ArithmeticGenus(charvar).

5. The second characteristic subvariety

The computation of the characteristic subvariety \( R_2 \) works essentially in the same way as in the previous section. As before, let \( e_1, ..., e_9 \) denote a basis of \( R_1 \). By \( e'_1 \) we denote a vector which spans the 1-dimensional space \( R_3 \). The dual of the multiplication map yields

\[ \mu^*: R_3^* \rightarrow S^3(R_1^*). \]

In order to write down a representation matrix, we fix a bijection

\[ : \{(i, j, k) \in \mathbb{N}^3 \mid 1 \leq i \leq j \leq k \leq 9\} \rightarrow \{1, ..., 210\}. \]

Let \( A \in K^{1 \times 165} \) a representation matrix of \( \mu^* \) with respect to \( e'_1 \) and \( b_{\varphi(i,j,k)} := e'_i e'_j e'_k \). Then \( R_2 \) is by definition a hypersurface in \( \mathbb{P}^8 \) given by the quation

\[ g := \sum_{i=1}^{9} \sum_{j=i}^{9} \sum_{k=j}^{9} a_{1,\varphi(i,j,k)} z_i z_j z_k \in K[z_1, ..., z_9]. \]

Thus we can proceed as follows.

(i) provide functions for the bijection \( \varphi \) and its inverse \( \varphi^{-1} \)
(ii) compute a representation matrix of the multiplication map \( \mu: S^3(R_1) \rightarrow R_3 \) with respect to the basis \( b_{\varphi(i,j,k)} := e_i e_j e_k \) and \( e'_i \)
(iii) obtain a representation matrix with respect to \( (e'_i)^* \) and \( \tilde{b}_{\varphi(i,j,k)} = (e_i e_j e_k)^* \) by transposition
(iv) deduce matrix \( A \) by base change from \( (e_i e_j e_k)^* \) to \( e'_i e'_j e'_k^* \)
(v) write down the defining equation of \( R_2 \)

Step (i) is achieved by

\[ r := \text{PolynomialRing}(GF(2), 9); \]
\[ \text{symm3order} := \text{MonomialsOfDegree}(r, 3); \]
\[ \text{symm3order} := \text{SetToSequence}(\text{symm3order}); \]
\[ \text{Symm3Spc} := \text{VectorSpace}(K, \#\text{symm3order}); \]

\[ \text{ijkpos} := \text{function}(i, j, k) \]
\[ S := \text{Sort}([i, j, k]); \]
\[ i_\ -= S[1]; j_\ -= S[2]; k_\ -= S[3]; \]
\[ \text{offs} := 0; \]
\[ \text{for } l := 1 \text{ to } i_- \text{ do} \]
p:=10 - l;
offs:=offs + 1/2*p*(p+1);
end for;
p0:=9-i_+1;
j0:=j_-i_+1;
k0:=k_-i_+1;
offs2:=0;
p0_:=p0 - j0 + 1;
offs2:=1/2*p0*(p0+1) - 1/2*p0_*(p0_+1);
offs2:=offs2 + (k0 - j0) + 1;
offs:=offs+offs2;
return offs;
end function;

pos2ijk:=function(pos)
mon:=symm3order[pos];
res:=[ ];
for i:=1 to 9 do
if Degree(mon,i) eq 3 then
return [i,i,i];
end if;
if Degree(mon,i) eq 2 then
res:=res cat [i,i];
end if;
if Degree(mon,i) eq 1 then
res:=Append(res,i);
end if;
end for;
return res;
end function;

For step (ii) we can use the function poly2vec from the previous section. The representation matrix of the multiplication map \( \mu \) is then computed by

\[
M:=[];
\]
for l:=1 to 165 do
M[1]:=[ ];
i,j,k:=pos2ijk(l); i:=ijk[1]; j:=ijk[2]; k:=ijk[3];
mon1:=basis[2][i]; mon2:=basis[2][j];
mon3:=basis[2][k];
vec:=poly2vec(mon1*mon2*mon3);
M[1][1]:=vec[20];
end for;
M:=Matrix(K,M);

For step (iii) and (iv), notice that \((e_i^*)^* = (e_i^*)^3\), \((e_i^2e_j)^* = 3(e_i^*)^2e_j^*\) and \((e_ie_je_k)^* = 6e_i^*e_j^*e_k^*\) for \(i, j, k\) pairwise distinct.

N:=[];
for l:=1 to 165 do
Now the characteristic variety is obtained by

\[
\text{CSR}:=\text{PolynomialRing}(K,9);
\text{charreq}:=\text{CSR!}0;
\text{for } l:=1 \text{ to } 165 \text{ do}
\quad \text{ijk:=pos2ijk}(l);
\quad i:=\text{ijk}[1]; j:=\text{ijk}[2]; k:=\text{ijk}[3];
\quad \text{mon}:=\text{CSR.i*CSR.j*CSR.k};
\quad \text{charreq}:=\text{charreq}+\text{C}[1][l]*\text{mon};
\quad \text{end for};
\text{P8:=ProjectiveSpace(CSR)};
\text{charvar:=Scheme(P8,charreq)};
\]

6. Computation of the Higgs field

For the implementation of the *Plethysm method* (see section 3 in [1]) it is neccessary to compute a representation matrices \( A_w \in K^{20 \times 20} \) of the maps

\[
\mu_w : R \longrightarrow R, \quad x \mapsto wx \quad \text{for } w \in R_1
\]

with respect to the basis computed in section 3. The idea is simple: Multiply every basis element with some fixed \( w \in R_1 \), use the function \text{poly2vec} from section 3 in order to transform the products into row vectors and form a matrix out of these rows. This task is carried out by the following code. The result is an array called \text{thetamats} whose elements are the representation matrices corresponding to the 9 basis vectors of \( R_1 \).

\[
\text{thetamats}:=[];
\text{for } j:=1 \text{ to } 9 \text{ do}
\quad \text{the}:=\text{basis}[2][j];
\quad \text{thetamat}:=[];
\quad \text{for } p:=0 \text{ to } 3 \text{ do}
\quad \quad \text{for } i:=1 \text{ to } \#\text{basis}[p+1] \text{ do}
\quad \quad \quad \text{poly}:=\text{the*basis}[p+1][i];
\]
vec:=poly2vec(poly);
vec:=ElementToSequence(vec);
thetamat:=Append(thetamat,vec);
end for;
end for;
thetamat:=Matrix(K,thetamat);
thetamats:=Append(thetamats,thetamat);
delete thetamat;
end for;

7. The induced Higgs field on symmetric 2-space

Each map $\mu_w : R \to R$ induces an endomorphism on $S^2(R)$. The aim of this section is to represent the elements of $S^2(R)$ by vectors in $K^{210}$, and to compute the action of $\mu_w$ with respect to this representation. For the first part, we only have to fix a bijection

$$\varphi : \{(i,j) \in \mathbb{N}^2 \mid 1 \leq i \leq j \leq 20\} \to \{1, 2, \ldots, 210\}.$$

If $b_1, \ldots, b_{20}$ denotes a basis of $R$, then the isomorphism $S^2(R) \to K^{210}$ is given by $b_ib_j \mapsto e_{\varphi(i,j)}$. The following two functions provide such a bijection.

```plaintext
r:=PolynomialRing(GF(2),20);
symm2order:=MonomialsOfDegree(r,2);
symm2order:=SetToSequence(symm2order);
Symm2Spc:=VectorSpace(K,#symm2order);

ijpos:=function(i,j)
if i gt j then
  i_:=j; j_:=i;
else
  i_:=i; j_:=j;
end if;
p:=20 - i_ + 1;
offs:=210 - 1/2*p*(p+1);
offs:=offs + (j_-i_) + 1;
offs:=Integers()!offs;
return offs;
end function;
pos2ij:=function(pos)
  mon:=symm2order[pos];
  res:=[I];
  for i:=1 to 20 do
    if Degree(mon,i) eq 2 then
      return [i,i];
    end if;
    if Degree(mon,i) eq 1 then
      res:=Append(res,i);
    end if;
  end for;
  return res;
end function;
```
Given an basis vector $w \in R_1$ and some $e_k \in K^{210} \cong S^2(R)$, our next task is to compute the image $\mu_w(e_k)$. By the above isomorphism, $e_k$ corresponds to an element of the form $b_i b_j$, and the induced map is given by
\[
\mu(b_i b_j) = b_i \mu(b_j) + \mu(b_i) b_j.
\]
It means that we have to proceed as follows:

(i) Use the matrices in $\text{thetamats}$ in order to compute the images $\mu(b_i)$ and $\mu(b_j)$ as elements in $K^{20}$.

(ii) Compute the product $b_i \mu(b_j)$: if $\mu(b_j) = \sum_{k=1}^{20} \alpha_k b_k$, then
\[
b_i \mu(b_j) = \sum_{k=1}^{20} \alpha_k b_i b_k.
\]

Use the bijection $\varphi$ in order to map this element to $K^{210}$.

(iii) Do the same for $\mu(b_i) b_j$, and return the sum.

This is carried out by the following Magma code.

```magma
symm2imthe:=function(th,pos)
    ij:=pos2ij(pos); i:=ij[1]; j:=ij[2];
    imi:=CSpc!0; imi[i]:=1;
    imj:=CSpc!0; imj[j]:=1;
    imj:=imj*thetamats[th];
    res1:=Symm2Spc!0;
    for k:=1 to 20 do
        if not(imj[k] eq 0) then
            res1[ijpos(i,k)]:=imi[i]*imj[k];
        end if;
    end for;
    imi:=CSpc!0; imi[i]:=1;
    imi:=imi*thetamats[th];
    imj:=CSpc!0; imj[j]:=1;
    res2:=Symm2Spc!0;
    for k:=1 to 20 do
        if not(imi[k] eq 0) then
            res2[ijpos(k,j)]:=imi[k]*imj[j];
        end if;
    end for;
    return res1+res2;
end function;
```

By linearity we can now compute the image of arbitrary elements in $K^{210}$. 

```magma
symm2imthe_:=function(th,vec)
    res:=Symm2Spc!0;
    seq:=ElementToSequence(vec);
    for i:=1 to #seq do
    end for;
    return res;
end function;
```
For the application of the plethysm method, it is essential to carry out the following task: Given a subspace \( U \leq S^2(R) \), compute the sum
\[
\mu_{w_1}(U) \oplus \mu_{w_2}(U) \oplus \cdots \oplus \mu_{w_9}(U)
\]
of the images of all Higgs field maps. (Here \( w_1, ..., w_9 \) denotes a basis of the graded subspace \( R^1 \).) We use the built-in linear algebra functions of MAGMA in order to provide this.

\[
\text{symm2image} := \text{function}(U) \\
\quad B := \text{Basis}(U); \\
\quad \text{ims} := [ ]; \\
\quad \text{for } \text{bvec in } B \text{ do} \\
\quad \quad \text{for } \text{th} := 1 \text{ to } 9 \text{ do} \\
\quad \quad \quad \text{ims} := \text{Append}(\text{ims}, \text{symm2imthe}(\text{th}, \text{bvec})); \\
\quad \quad \text{end for}; \\
\quad \text{end for}; \\
\quad \text{return sub<Symm2Spc|ims>;} \\
\end{function};
\]

8. The graded subspaces of symmetric 2-space

The grading \( R = \oplus_{p=0}^3 R_p \) on the Jacobian ring induces a natural grading
\[
S^2(R) = \sum_{p=0}^6 S^2(R)_p
\]
on symmetric 2-space. For the plethysm method, we have to compute the iterated images of the graded subspaces \( S^2(R)_p \) under the Higgs field elements. In particular, for \( 0 \leq p \leq 6 \) we have to determine the image of \( S^2(R)_p \) under the isomorphism \( S^2(R)_p \cong K^{210} \). We provide a function which returns for each \( p \) a set \( \{i_1, ..., i_p\} \) such that the image of \( S^2(R)_p \) in \( K^{210} \) is spanned by the basis vectors \( e_{i_1}, ..., e_{i_p} \).

It seems to be the easiest strategy to compute the index set for each \( p \) “by hand”. Let \( b_1, ..., b_{20} \) denote a basis of \( R \). We assume that the indices are chosen such that \( R_0 \) is spanned by \( b_1 \), \( R_1 \) is spanned by \( b_2, ..., b_{10} \), \( R_2 \) by the elements \( b_{11}, ..., b_{19} \) and finally \( R_3 \) by \( b_{20} \). We first consider \( p = 0 \). In this case \( S^2(R)_0 \) is generated by the image of \( R_0 \otimes R_0 \), which means that the image of \( S^2(R)_0 \) in \( K^{210} \) is spanned by \( e_{\varphi(1,1)} \). The space \( S^2(R)_1 \) is generated by \( R_0 \otimes R_1 \), so the image is spanned by \( b_1b_i = e_{\varphi(1,i)} \) with \( 2 \leq i \leq 9 \). For \( p = 2 \) we have to consider the image of \( (R_0 \otimes R_2) \oplus (R_1 \otimes R_1) \) etc. If we work through all degrees up to 6, we end up with the following function.
symm2degind:=function(p)
    case p:
        when 0:
            return [ijpos(1,1)];
        when 1:
            res:=[ ];
            for i:=1 to 9 do
                res:=Append(res,ijpos(1,1+i));
            end for;
            return res;
        when 2:
            res:=[ ];
            for i:=1 to 9 do
                res:=Append(res,ijpos(1,10+i));
            end for;
            for i:=1 to 9 do
                for j:=i to 9 do
                    res:=Append(res,ijpos(1+i,1+j));
                end for;
            end for;
            return res;
        when 3:
            res:=[ijpos(1,20)];
            for i:=1 to 9 do
                for j:=1 to 9 do
                    res:=Append(res,ijpos(1+i,10+j));
                end for;
            end for;
            return res;
        when 4:
            res:=[ ];
            for i:=1 to 9 do
                res:=Append(res,ijpos(1+i,20));
            end for;
            for i:=1 to 9 do
                for j:=i to 9 do
                    res:=Append(res,ijpos(10+i,10+j));
                end for;
            end for;
            return res;
        when 5:
            res:=[ ];
            for i:=1 to 9 do
                res:=Append(res,ijpos(10+i,20));
            end for;
            return res;
        when 6:
            res:=[ijpos(20,20)];
            return res;
    end case;
    return -1;
end function;
The indices provided by \texttt{symm2degind} now admit the computation of the $p$-th graded subspace of $S^2(R)$.

\begin{verbatim}
symm2degspc:=function(p)
basis:=[];
inds:=symm2degind(p);
for i:=1 to #inds do
  vec:=Symm2Spc!0;
  vec[inds[i]]:=1;
  basis:=Append(basis,vec);
end for;
return sub<Symm2Spc|basis>;
end function;
\end{verbatim}

The above function now have to be invoked in the following way in order to make the plethysm method work.

\begin{verbatim}
U51:=symm2degspc(1);
U42:=symm2image(U51);
U33:=symm2image(U42);
Dimension(U33);
\end{verbatim}

The result is 78, whereas the modularity assumption would have implied a dimension $\leq 65$.

\section*{References}

[1] R. Gerkmann, M. Sheng and K. Zuo. \textit{Disproof of modularity of the moduli space of CY 3-folds coming from eight planes of $\mathbb{P}^3$ in general positions}

[2] J. Canon et al. \textit{The computer algebra system Magma}. Download, documentation and further information available at magma.maths.usyd.edu.au/magma.

Universität Mainz, Fachbereich 17, Mathematik, 55099 Mainz, Germany
\textit{E-mail address}: gerkmann@mathematik.uni-mainz.de

East China Normal University, Dep. of Mathematics, 200062 Shanghai, P.R. China
\textit{E-mail address}: msheng@math.ecnu.edu.cn

Universität Mainz, Fachbereich 17, Mathematik, 55099 Mainz, Germany
\textit{E-mail address}: kzuo@mathematik.uni-mainz.de