An Interface between Mathematica and Singular

Manuel Kauers and Viktor Levandovskyy

The authors acknowledge support by the grants SFB F1305 (Kauers) and SFB F1301 (Levandovskyy) of the Austrian FWF.

Mission Statement

The purpose of this package is to provide an easy way to access some of the functionality provided by the special purpose computer algebra system Singular from within a Mathematica session or package. It is a common phenomenon that the advantages of general purpose systems and special purpose systems are greatly orthogonal. With this package, we aim at combining some of the advantages of Singular and Mathematica.

Singular (http://www.singular.uni−kl.de) is a system specialized for computations in polynomial rings. It contains sophisticated and flexible implementations of algorithms for commutative algebra and algebraic geometry. Compared to Mathematica, some of its advantages are that it
- has high performance
- has implementations for sophisticated algorithms not contained in Mathematica

On the other hand, Mathematica
- has a simple and easy−to−use language
- has implementations of standard algorithms for a wide range of different topics

This package provides access to the most frequently needed facilities of Singular. These include mainly
- various ways for computing Gröbner bases
- ideal and module arithmetic, "Gröbner basics"
- primary decomposition of polynomial ideals

Many further functionalities of Singular are currently not covered by the interface package, but the interface code should be easily extensible to most of them. The interface package is mainly addressed to users of Mathematica who need more flexible, functionally rich and powerful Gröbner bases−related computations than those which are provided by Mathematica. The package might help users with background in Mathematica to make first steps in getting acquainted with Singular.

Singular can be obtained free of charge from http://www.singular.uni−kl.de. It is distributed under the GNU Public licence (GPL). The interface package can be obtained from the website http://www.risc.uni−linz.ac.at/research/combinat/software/Singular/ and/or upon email request. The GPL applies also to the interface package. This notebook contains a complete description of the functionality of the interface package. (There is no documentation of the package beyond this.) For additional information about Singular itself, we refer to [1, 2].

■ Licence

This software is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. The programs are distributed in the hope that they will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License (http://www.gnu.org/licenses/gpl.html) for more details.

■ Bibliography

[1] G.–M. Greuel, G. Pfister. A Singular Introduction to Commutative Algebra. Springer. 2002.

[2] G.–M. Greuel, G. Pfister and H. Schönemann. Singular 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2006). http://www.singular.uni–kl.de.

```
In[1]:= << Singular.m
```

```
Singular -- Interface to Mathematica   Package by Manuel Kauers
(mkauers@risc.uni-linz.ac.at) and Viktor Levandovskyy (levandov@risc.uni-linz.ac.at)
http://www.risc.uni-linz.ac.at/research/combinat/software/Singular/ — © RISC Linz — V 0.7
(2006-07-28)
```

Quick Tour: Gröbner Basis Computation using Singular

Ideals are represented as lists of polynomial expressions, such as the following.

```
In[2]:= ideal =
          {x^2 * y - t * y^3 * x, x^2 * y^2 - Sqrt[3], Sqrt[3] * x - 2 * Sqrt[3] * t * y^2 + x^3 * y^2};
```

In order to compute a Gröbner basis of an ideal, we may use Singular's command **Groebner,** as follows:

```
In[3]:= SingularGroebner[ideal, {x, y}, MonomialOrder → DegreeReverseLexicographic]
```

$$\text{Out[3]}= \left\{-x + t\, y^2,\ -\sqrt{3}\, t + x^3\right\}$$

The arguments are the same as for Mathematica's **GroebnerBasis** command:

```
In[4]:= GroebnerBasis[ideal, {x, y}, MonomialOrder → DegreeReverseLexicographic]
```

$$\text{Out[4]}= \left\{x - t\, y^2,\ \sqrt{3}\, t - x^3\right\}$$

Unlike Mathematica, Singular can also compute Gröbner bases of modules (more precisely, of submodules of free modules). Such a module is represented by a list of vectors, each vector being represented as a list of polynomials.

```
In[5]:= mod = {{x^2 * y - t * y^3 * x, 1, 0, 0},
              {x^2 * y^2 - Sqrt[3], 0, 1, 0},
              {Sqrt[3] * x - 2 * Sqrt[3] * t * y^2 + x^3 * y^2, 0, 0, 1}};
```

```
General::spell1 :
 Possible spelling error: new symbol name "mod" is similar to existing symbol "Mod". More...
```

```
In[6]:= SingularStd[mod, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[6]= {{6 x - 6 t y^2, 0, -√3 x, √3}, {0, 2 x y, -x + 2 t y^2, -1},
         {0, 2 √3, x^2 y, -x y}, {-6 t + 2 √3 x^3, 0, 2 √3 t - x^3, x^2},
         {6 √3 t - 6 x^3, 0, -6 t + √3 x^3, -√3 x^2}, {-√3 t y + x^3 y, x, t y, 0}}

In[7]:= SingularStd[mod, {x, y}, MonomialOrder → Lexicographic]

Out[7]= {{2 √3 x - 2 √3 t y^2, 0, -x, 1}, {-6 x + 6 t y^2, 0, √3 x, -√3},
         {0, 2 x y, -x + 2 t y^2, -1}, {-√3 + t x y^4, -y, 1, 0},
         {6 - 2 √3 t^2 y^6, 2 √3 y, -2 √3 - t x y^4, t y^4},
         {x^2 y - t x y^3, 1, 0, 0}, {0, 2 √3, x^2 y, -x y}}
```

Operations on Ideals and Modules

■ Gröbner bases and Normal Forms

Three commands are available for computing Gröbner bases: **SingularGroebner**, **SingularStd**, and **SingularSlimgb**. They correspond to the respective commands of Singular.

Normal forms of a polynomial (vector) modulo an ideal (a module) can be obtained using the equivalent commands **SingularReduce** and **SingularNF**.

■ Example 1

```
In[8]:= ideal =
          {x^2 * y - t * y^3 * x, x^2 * y^2 - Sqrt[3], Sqrt[3] * x - 2 * Sqrt[3] * t * y^2 + x^3 * y^2};

In[9]:= SingularGroebner[ideal, {x, y, t}, MonomialOrder → DegreeReverseLexicographic]

Out[9]= {-x + t y^2, -√3 t + x^3, -√3 + x^2 y^2}

In[10]:= SingularStd[ideal, {x, y, t}, MonomialOrder → DegreeReverseLexicographic]

Out[10]= {-x + t y^2, -√3 t + x^3, -√3 + x^2 y^2}

In[11]:= SingularSlimgb[ideal, {x, y, t}, MonomialOrder → DegreeReverseLexicographic]

Out[11]= {-x + t y^2, -√3 t + x^3, -√3 + x^2 y^2}

In[12]:= gbasis = %;

In[13]:= SingularReduce[x^2 + x^3 + 2 * t * x * y - 2 * t^2 * y^3 +
           t * x^2 * y^3 + Sqrt[3] * (-t - x^3 - t * y + t * x^2 * y^2),
          gbasis, {x, y, t}, MonomialOrder → DegreeReverseLexicographic]

Out[13]= x^2
```

■ Example 2

```
In[14]:= mod = {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}}

Out[14]= {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}}
```

```
In[15]:= SingularGroebner[mod, {x, y}, MonomialOrder → Lexicographic]

Out[15]= {{y^4, y}, {y, y^4}, {x, y^2}, {y^2, x}}

In[16]:= SingularStd[mod, {x, y}, MonomialOrder → Lexicographic]

Out[16]= {{y^4, y}, {y, y^4}, {x, y^2}, {y^2, x}}

In[17]:= SingularSlimgb[mod, {x, y}, MonomialOrder → Lexicographic]

Out[17]= {{y^4, y}, {y, y^4}, {x, y^2}, {y^2, x}}

In[18]:= gbasis = %;

In[19]:= vec = (x^2 + y^2 + 1) * {x^2, y} - 5 * {y^2, x} + (x^2 - a) * {x, y^2} - 1 / 8 * {y, x^2}

Out[19]= {x (-a + x^2) - y/8 - 5 y^2 + x^2 (1 + x^2 + y^2), -5 x - x^2/8 + (-a + x^2) y^2 + y (1 + x^2 + y^2)}

In[20]:= SingularNF[vec, gbasis, {x, y}, MonomialOrder → Lexicographic]

Out[20]= {0, 0}
```

■ Elimination

Given an ideal (or a module) *I*, the ideal (or module) of all elements of *I* free of certain specified variables can be computed via the command **SingularEliminate.** It corresponds to Singular's **eliminate** command.

The command expects two lists of variables. The first list contains the variables that are to be eliminated, and the second list contains the remaining variables. Symbols appearing in neither list are treated as elements of the ground field (parameters).

■ Example 1

```
In[21]:= ideal =
           {x^2 * y - t * y^3 * x, x^2 * y^2 - Sqrt[3], Sqrt[3] * x - 2 * Sqrt[3] * t * y^2 + x^3 * y^2};

In[22]:= SingularEliminate[ideal, {x}, {y, t}]

Out[22]= {-Sqrt[3] + t^2 y^6}

In[23]:= SingularEliminate[ideal, {t}, {x, y}]

Out[23]= {-Sqrt[3] + x^2 y^2}

In[24]:= SingularEliminate[ideal, {y}, {x, t}]

Out[24]= {-Sqrt[3] t + x^3}

In[25]:= SingularEliminate[ideal, {y}, {x}]

Out[25]= {-Sqrt[3] t + x^3}
```

■ Example 2

```
In[26]:= mod = {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}}

Out[26]= {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}}
```

```
In[27]:= SingularEliminate[mod, {x}, {y}]

Out[27]= {{y^4, y}, {y, y^4}}

In[28]:= SingularEliminate[mod, {y}, {x}]

Out[28]= {{x^4, x}, {x, x^4}}
```

■ Syzygy Computation

Given a list of polynomials (or vectors), the syzygy module of this vector can be computed via the command **SingularSyz.** It corresponds to Singular's command **syz**.

■ Example 1

```
In[29]:= ideal =
           {x^2*y - t*y^3*x, x^2*y^2 - Sqrt[3], Sqrt[3]*x - 2*Sqrt[3]*t*y^2 + x^3*y^2};

In[30]:= SingularSyz[ideal, {x, y}]

Out[30]= {{2 x y, -x + 2 t y^2, -1}, {6, Sqrt[3] x^2 y, -Sqrt[3] x y}}

In[31]:= % . ideal // Expand

Out[31]= {0, 0}
```

■ Example 2

```
In[32]:= mod = {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}};

In[33]:= SingularSyz[mod, {x, y}]

Out[33]= {{-y^2, x^2, y, -x}, {-x, y, x^2, -y^2}}

In[34]:= % . mod // Expand

Out[34]= {{0, 0}, {0, 0}}
```

■ Cofactor Computation

Given an ideal (or a module) *I* and a subideal (resp. submodule) *J*, the transformation matrix that exhibits the representation of the basis elements of *J* in terms of the basis elements of *I* can be computed via the command **SingularLift.** It corresponds to Singular's command **lift.**

An error is generated if *J* is not a subideal (resp. submodule) of *I*.

■ Example 1

```
In[35]:= ideal =
           {x^2*y - t*y^3*x, x^2*y^2 - Sqrt[3], Sqrt[3]*x - 2*Sqrt[3]*t*y^2 + x^3*y^2};

In[36]:= ideal2 = {(x^2 + y)*First[ideal] + y*Last[ideal],
            (3*x + 2 y^10)*First[ideal] - (x - y) Last[ideal]};
```

```
In[37]:= SingularLift[ideal, ideal2, {x, y}]
```

$$\text{Out[37]}= \left\{\left\{0, -\frac{x^4\, y}{2\sqrt{3}} - \frac{x^2\, y^2}{2\sqrt{3}}, y + \frac{x^3\, y}{2\sqrt{3}} + \frac{x\, y^2}{2\sqrt{3}}\right\},\right.$$
$$\left.\left\{0, -\frac{1}{2}\sqrt{3}\, x^3\, y - \frac{x^2\, y^{11}}{\sqrt{3}}, -x + y + \frac{1}{2}\sqrt{3}\, x^2\, y + \frac{x\, y^{11}}{\sqrt{3}}\right\}\right\}$$

```
In[38]:= % . ideal - ideal2 // Expand

Out[38]= {0, 0}
```

■ Example 2

```
In[39]:= mod = {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}};

In[40]:= mod2 = {(x^2 + y) * mod[[1]] + mod[[2]] - (3 x + 1) * mod[[3]],
             (3 x + y^2) mod[[2]] - (4 x^2 + y^2 - 1) * mod[[4]]};

In[41]:= SingularLift[mod, mod2, {x, y}]

Out[41]= {{x^2 + y, 1, -1 - 3 x, 0}, {x, 3 x - y + y^2, -x^2, 1 - 4 x^2}}

In[42]:= %.mod - mod2 // Expand

Out[42]= {{0, 0}, {0, 0}}
```

■ Ideal Arithmetic

There are commands available for computing sum, product, intersection, and quotients and saturation of ideals and modules.

■ Plus and Times

```
In[43]:= SingularPlus[{x^2 + 1, y^2 + 1}, {x + y}, {x, y}]

Out[43]= {1 + x^2, 1 + y^2, x + y}

In[44]:= SingularTimes[{x^2 + 1, y^2 + 1}, {x + y}, {x, y}]

Out[44]= {(1 + x^2) (x + y), (x + y) (1 + y^2)}

In[45]:= SingularPlus[{{x^2 + 1, a}, {y^2 + 1, b}}, {{x + y, c}}, {x, y}]

Out[45]= {{1 + x^2, a}, {1 + y^2, b}, {x + y, c}}
```

■ Intersection

```
In[46]:= SingularIntersect[{x^2 + 1, y^2 + 1}, {x + y}, {x, y}]

Out[46]= {x^2 - y^2, x + y + x y^2 + y^3}

In[47]:= SingularIntersect[{{x^2 + 1, a}, {y^2 + 1, b}}, {{x + y, c}}, {x, y}]

Out[47]= {{(a - b) x - b x^3 + (a - b) y - b x^2 y + a x y^2 + a y^3, a c - b c - b c x^2 + a c y^2}}
```

■ Ideal Quotient

```
In[48]:= SingularQuotient[{x^9*y, y^2*x}, {x*y}, {x, y}]

Out[48]= {y, x^8}

In[49]:= SingularQuotient[{{x^9*y, x*y}, {y^2*x, x*y}}, {x*y}, {x, y}]

Out[49]= {{y, 1}, {x^8, 1}, {0, x^8 - y}}
```

■ Saturation

```
In[50]:= SingularSat[{x^9*y, y^2*x}, {x*y}, {x, y}]

Out[50]= {{1}, 2}

In[51]:= SingularSat[{{x^9*y, x*y}, {y^2*x, x*y}}, {x*y}, {x, y}]

Out[51]= {{{y, 1}, {x^8, 1}, {0, x^8 - y}}, 1}
```

■ Dimension

The dimension of an ideal (of the annihilator of a module) can be computed using the **SingularDim** command, which corresponds to Singular's **dim**. It only works correctly if the ideal/module is given by a Gröbner basis.

For a zero dimensional ideal (or module) I, the command **SingularVdim** gives the vector space dimension of $K[X]/I$ (respectively $K[X]^m/I$) over K.

■ Example 1

```
In[52]:= ideal =
           {x^2*y - t*y^3*x, x^2*y^2 - Sqrt[3], Sqrt[3]*x - 2*Sqrt[3]*t*y^2 + x^3*y^2};

In[53]:= gbasis = SingularStd[ideal, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[53]= {x - t y^2, -Sqrt[3] t + x^3}

In[54]:= SingularDim[gbasis, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[54]= 0
```

If the input basis was not a Gröbner basis, or it is a Gröbner basis for a different term ordering, the result may be wrong:

```
In[55]:= SingularDim[ideal, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[55]= 1

In[56]:= SingularDim[gbasis, {x, y}, MonomialOrder → Lexicographic]

Out[56]= 1

In[57]:= SingularVdim[gbasis, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[57]= 12
```

```
In[58]:= gbasis = SingularStd[ideal, {x, y, t}, MonomialOrder → DegreeReverseLexicographic]

Out[58]= {-x + t y^2, -Sqrt[3] t + x^3, -Sqrt[3] + x^2 y^2}

In[59]:= SingularDim[gbasis, {x, y, t}, MonomialOrder → DegreeReverseLexicographic]

Out[59]= 1

In[60]:= SingularVdim[gbasis, {x, y, t}, MonomialOrder → DegreeReverseLexicographic]

Out[60]= ∞

In[61]:= SingularDim[{1}, {x, y, z}, MonomialOrder → DegreeReverseLexicographic]

Out[61]= -1
```

■ Example 2

```
In[62]:= mod = {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}}

Out[62]= {{x^2, y}, {y^2, x}, {x, y^2}, {y, x^2}}

In[63]:= gbasis = SingularStd[mod, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[63]= {{y^2, x}, {x, y^2}, {x^2, y}, {y, x^2}}

In[64]:= SingularDim[gbasis, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[64]= 0

In[65]:= SingularVdim[gbasis, {x, y}, MonomialOrder → DegreeReverseLexicographic]

Out[65]= 8
```

■ Primary Decomposition

The primary decomposition of a given ideal can be computed via the command **SingularPrimdecGTZ**, which corresponds to Singular's **primdecGTZ**. It returns a list of pairs of ideals, the first of which is the primary component and the second of which is the associated prime.

If only the associated primes are of interest, the command **SingularMinAssGTZ**, which corresponds to Singular's **minAssGTZ**, can be used. It returns a list of ideals.

There are additional algorithms for computing primary decomposition and associated primes available in Singular. You may wish to use Singular directly if the commands accessible through the interface turn out to be insufficient.

■ Example

```
In[66]:= SingularPrimdecGTZ[
          {y^2 * z^2 - x^2 * y^3 - x * z^3 + x^3 * y * z, y^2 * z - x * z^2}, {x, y, z}]

Out[66]= {{{-y^2 + x z}, {-y^2 + x z}}, {{z^2, y}, {z, y}}, {{z, x^2}, {z, x}}}

In[67]:= SingularMinAssGTZ[
          {y^2 * z^2 - x^2 * y^3 - x * z^3 + x^3 * y * z, y^2 * z - x * z^2}, {x, y, z}]

Out[67]= {{z, x}, {-y^2 + x z}}
```

Advanced Topics

■ Which ring is used?

The ring, in which the computations are carried out, depends on the expressions provided as input, the specified variables, and the values of certain options. Each ring has the form $K(\alpha)(t_1, t_2, ...)[x_1, x_2, ...]$ for a polynomial ring or $K(\alpha)(t_1, t_2, ...)[x_1, \ldots, x_n]_{\langle x_1,\ldots,x_n\rangle}$ for a polynomial ring localized at the maximal ideal. Here, K is either the field of rational numbers of a finite field, α is algebraic over K, and the t_i are transcendental over $K(\alpha)$. (Singular can treat more sophisticated rings, but they are not available through this interface.)

By default, K is the field of rational numbers. By setting the **`Modulus`** option to a positive prime less than 2147483630, a finite field can be selected. The algebraic extension is constructed by extracting all subexpressions from the input which represent algebraic numbers (such as **`Sqrt`** and **`Root`** expressions), and taking as α the primitive element of all these numbers. The computation of a primitive element, which is done in Mathematica, may consume a substantial portion of the overall runtime.

Arbitrary expressions may be used as variables (except polynomials or lists, of course). Every input expression is traversed and subexpressions which do not constitute lists or polynomials are collected. Those which appear in the variable list are taken as the variables of the ring (the x_i from above), the others are considered as transcendental elements over $K(\alpha)$ (the t_i from above).

Whether or not a polynomial ring or a localized polynomial ring will be used depends on the choice of the ordering. See the following section.

■ Specification of Sophisticated Term Orderings

For several commands, the output depends on the choice of a term ordering. The term ordering can be specified using the option **`MonomialOrder`**. The value of this option together with the ordering of the variables determines the term ordering of the underlying ring.

■ Standard Orderings

The following symbols are used to refer to the basic orderings known to the interface. For a precise definition of these orderings we refer to the Singular manual.

Term Orderings for polynomial rings (global orderings):

`Symbol`	**`Singular Name`**
`Lexicographic`	`lp`
`DegreeLexicographic`	`Dp`
`DegreeReverseLexicographic`	`dp`
`WeightedLexicographic`	`Wp`
`WeightedReverseLexicographic`	`wp`

Term Orderings for localized polynomial rings (local orderings):

Symbol	Singular Name
LexicographicSeries	ls
DegreeLexicographicSeries	Ds
DegreeReverseLexicographicSeries	ds
WeightedLexicographicSeries	Ws
WeightedReverseLexicographicSeries	ws

The weighted orderings require an integer vector as argument, the others are used plainly.
The symbolic names may be used interchangeably with the quoted Singular names.

Examples.

```
In[68]:= SingularStd[{a*b, b^2, c*a, d*e, e^2},
          {a, b, c, d, e}, MonomialOrder → DegreeReverseLexicographic]

Out[68]= {e^2, d e, a c, b^2, a b}

In[69]:= SingularStd[{a*b, b^2, c*a, d*e, e^2}, {a, b, c, d, e}, MonomialOrder → "dp"]

Out[69]= {e^2, d e, a c, b^2, a b}

In[70]:= SingularStd[{a*b, b^2, c*a, d*e, e^2},
          {a, b, c, d, e}, MonomialOrder → DegreeLexicographic]

Out[70]= {e^2, d e, b^2, a c, a b}

In[71]:= SingularStd[{a*b, b^2, c*a, d*e, e^2}, {a, b, c, d, e},
          MonomialOrder → WeightedReverseLexicographic[1, 2, 3, 4, 5]]

Out[71]= {a b, a c, b^2, d e, e^2}

In[72]:= SingularStd[{a*b, b^2, c*a, d*e, e^2}, {a, b, c, d, e},
          MonomialOrder → WeightedReverseLexicographicSeries[1, 2, 3, 4, 5]]

Out[72]= {a b, b^2, a c, d e, e^2}
```

■ Matrix Orderings

In addition to the standard orderings, term orderings can be defined using explicit weight matrices. Any regular integer matrix can be specified as a weight matrix. Example:

```
In[73]:= ideal =
           {x^2*y - t*y^3*x, x^2*y^2 - Sqrt[3], Sqrt[3]*x - 2*Sqrt[3]*t*y^2 + x^3*y^2};

In[74]:= SingularStd[ideal, {x, y, t}, MonomialOrder → {{1, 1, 1}, {1, 2, 3}, {1, 4, 9}}]

Out[74]= {-Sqrt[3] t + x^3, -x + t y^2, -Sqrt[3] + x^2 y^2}
```

■ Product Orderings (block orderings)

New orderings can be created out of existing ones by dividing the list of variables into a number of blocks and assigning to each block its own term ordering. In order to use such an ordering, one can give several lists of variables as arguments. Each list then corresponds to a block.

```
In[75]:= ideal =
           {x^2*y - t*y^3*x, x^2*y^2 - Sqrt[3], Sqrt[3]*x - 2*Sqrt[3]*t*y^2 + x^3*y^2};

In[76]:= SingularStd[ideal, {x, y}, {t}, MonomialOrder → DegreeReverseLexicographic]

Out[76]= {-x + t y^2, -Sqrt[3] t + x^3, -Sqrt[3] + x^2 y^2}
```

If only a single symbol is given in the option **MonomialOrder,** then this ordering is applied to all blocks. If different blocks should be sorted differently, a list of orderings has to be given, with one ordering per block.

```
In[77]:= SingularStd[ideal, {x, y}, {t},
          MonomialOrder → {DegreeReverseLexicographic, Lexicographic}]

Out[77]= {-x + t y^2, -Sqrt[3] t + x^3, -Sqrt[3] + x^2 y^2}
```

In this list, any ordering, including matrix and weight orderings, may be specified. The dimensions of matrix and weight orderings has to match the size of the corresponding block.

```
In[78]:= SingularStd[ideal, {x, y}, {t},
          MonomialOrder → {{{1, 2}, {3, 4}}, WeightedLexicographic[19]}]

Out[78]= {-Sqrt[3] t + x^3, -Sqrt[3] + x^2 y^2, -x + t y^2}
```

■ Module Orderings

Singular treats the unit vectors of modules (which indicate the position, or a component, of a monomial in a vector of polynomials) as an independent block of variables. Within this block, the generators can be sorted either in ascending order (i.e., $e_i > e_j$ for $i > j$) or in descending order (i.e. $e_i < e_j$ for $i > j$). Also the relative position of this block to the other "ordinary" variable blocks is up to the user, allowing to use "position–over–term" (POT) orderings (first order with respect to the module ordering, than with the ordering on the variables) or the "term–over–position" (TOP) orderings.

The symbols **ModuleAscending** and **ModuleDescending** represent the block of module generators and request that they be sorted in ascending and descending order, respectively. One of these symbols may be put at any position into the specification of a product ordering. Singular's default will be applied if none of the symbols is used.

Singular's notation **"C"** or **"c"** may be used instead of the symbolic names **ModuleAscending** and **ModuleDescending,** respectively.

Example.

```
In[79]:= ideal =
           {x^2*y - t*y^3*x, x^2*y^2 - Sqrt[3], Sqrt[3]*x - 2*Sqrt[3]*t*y^2 + x^3*y^2};

In[80]:= mod = SingularSyz[ideal, {x, y, t}];

In[81]:= SingularStd[mod, {x, y, t}, MonomialOrder → {Lexicographic, ModuleAscending}]

Out[81]= {{2 x y, -x + 2 t y^2, -1}, {2 Sqrt[3], x^2 y, -x y}}

In[82]:= SingularStd[mod, {x, y, t}, MonomialOrder → {ModuleAscending, Lexicographic}]

Out[82]= {{Sqrt[3] - x^2 y^2, x^2 y - t x y^3, 0}, {-2 x y, x - 2 t y^2, 1}}

In[83]:= SingularStd[mod, {x, y}, {t}, MonomialOrder →
           {WeightedReverseLexicographic[7, 9], ModuleDescending, Lexicographic}]

Out[83]= {{2 x y, -x + 2 t y^2, -1}, {2 Sqrt[3], x^2 y, -x y}, {-4 Sqrt[3] t y + 2 x^3 y, -x^3, -x^2 + 2 t x y^2}}

In[84]:= SingularStd[mod, {x, y}, {t}, MonomialOrder → {"wp"[7, 9], "c", "lp"}]

Out[84]= {{2 x y, -x + 2 t y^2, -1}, {2 Sqrt[3], x^2 y, -x y}, {-4 Sqrt[3] t y + 2 x^3 y, -x^3, -x^2 + 2 t x y^2}}
```

■ Some Technical Issues

■ Where is Singular?

The variable **SingularCommand** carries the command line that is used by the interface for launching Singular.

By default, it just contains the string **"Singular",** but depending on the particular installation, it might be necessary to include also the full directory path that leads to the executable.

```
In[85]:= SingularCommand

Out[85]= Singular
```

If the operating system is recognized as Windows, the **SingularCommand** will be set to **"bash Singular",** because the Windows version of Singular is based on cygwin. If the cygwin executables are not found automatically, you may either modify your systems PATH variable (via Start → Control Panel → Systems → Advanced → Environment_Variables), or modify the **SingularCommand** appropriately. The necessary condition is that executing the **SingularCommand** in a DOS shell (Start → Run, type **cmd**) must start a Singular session.

■ I/O Control

All commands that invoke Singular proceed by first composing an input file containing all the commands that have to be executed, then calling Singular on that input file. Singular will write all its output into a certain output file, then exit and return control to Mathematica. Then Mathematica will read that file and convert its content into Mathematica expressions which are then finally presented to the user.

By default, temporary files are used as input and output file, and these files are deleted when they are no longer used. Instead of using temporary files, specific files names may be given by the user via the **InFileName** and **OutFileName** options. When these options are present, their values will be used as input and/or output file names. These files are not deleted after usage.

```
In[86]:= SingularStd[{x, y}, {x, y}, MonomialOrder → Lexicographic,
           InFileName -> "test.in", OutFileName -> "test.out"]

Out[86]= {y, x}


In[87]:= !! test.in

         ring r = 0, (x1, x2), lp;
         option(redSB); option(redTail);
         ideal a1 = x1, x2;
         ideal result = std(a1);
         write(":a test.out", result);
         quit;

In[88]:= !! test.out

         x2,x1
```

■ Query Version Information.

The command **SingularVersion[]** prints the version of the underlying Singular system.

```
In[89]:= SingularVersion[]

Out[89]= 3001
```

■ Bugs

Please report any bugs that are *not* related to Singular, but to the interface, to Manuel Kauers (mkauers@risc.uni–linz.ac.at).