Computer Algebra System SINGULAR 2-2-0 Goes Noncommutative

http://www.singular.uni-kl.de

V. Levandovskyy
Fachbereich Mathematik
TU Kaiserslautern, Germany
email: levandov@mathematik.uni-kl.de
http://www.singular.uni-kl.de/plural

1 Introduction

SINGULAR 2-2-0 is a further development of both actual 2-0-X release series and 2-1-X pre-release series. The new SINGULAR 2-2-0 offers a wide range of possibilities for modern researcher in the field of both commutative and noncommutative polynomial computation.

SINGULAR is a free service to the scientific community. Moreover, it has been proved to be a good development platform, featuring flexibility of different levels of an user/system interaction.

The 2004 *Richard D. Jenks Memorial Prize for Excellence in Software Engineering for Computer Algebra* was awarded to the SINGULAR team at the yearly premier International Symposium on Symbolic and Algebraic Computation (ISSAC) in Santander (Spain).

2 Key Strengths of SINGULAR

- distributed under GPL (GNU Public License)
- available for most of hardware and software platforms
- the biggest choice of ground fields on the market
- the factorization over most of fields is provided
- one of the fastest implementations
  - ▷ of Gröbner basis algorithm (also noncommutative)
  - ▷ of standard basis algorithm
- primary decomposition for ideals and modules
- resolution of singularities
- **noncommutative subsystem PLURAL**
- additional functionality: more than 60 libraries available
- exhaustive documentation over 1000 pages long
- extensive support via e-mail and online SINGULAR Forum

3 Which Rings Can Be Handled with PLURAL?

Let $\mathbb{K}$ be a field and $R = \mathbb{K}[x_1, \ldots, x_n]$ be a commutative ring.
Suppose there are elements $c_{ij} \in \mathbb{K} \setminus \{0\}$ and $d_{ij} \in R$, $\forall\, 1 \le i < j \le n$.

Consider an algebra $A = \mathbb{K}\langle x_1, \ldots, x_n \mid \forall\, i < j\ x_j x_i = c_{ij} x_i x_j + d_{ij}\rangle$.
It is called **a G–algebra** (in n variables) if the following conditions hold:

1) $\exists \prec$, a well–ordering on R such that $\forall\, i < j \quad \mathrm{lm}(d_{ij}) \prec x_i x_j$,

2) *Nondegeneracy conditions* are fulfilled, that is $\forall\, 1 \le i < j < k \le n$

$$c_{ik}c_{jk} \cdot d_{ij}x_k - x_k d_{ij} + c_{jk} \cdot x_j d_{ik} - c_{ij} \cdot d_{ik} x_j + d_{jk} x_i - c_{ij} c_{ik} \cdot x_i d_{jk} = 0.$$

Theorem. Let A be a G–algebra in n variables. Then

- A has a PBW basis $\{x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \mid \alpha_i \in \mathbb{N} \cup \{0\}\}$,
- A is left and right Noetherian,
- A is an integral domain with $\mathrm{gl.\,dim}\, A \le n$.

A GR**–algebra** is a factor of G–algebra by a proper two–sided ideal. There are several comfortable ways to construct GR–algebras.

1. **Generic Matrices**: creating a G–algebra according to the definition.
   Input: a ring R, $n \times n$ matrices $C = (c_{ij})$ and $D = (d_{ij})$.
   Shortcuts: if either all c_{ij} or all d_{ij} are equal, you can input just one value for all of them.
2. **Tensor Products**: starting from two G–algebras A and B, one can easily create a G–algebra $A \otimes_{\mathbb{K}} B$.
3. **Library Procedures**: many important algebras are predefined in libraries. One can initialize these algebras over fields of different characteristics. For quantum algebras, specializing the quantum parameter at the given root of unity is possible.

• Weyl and Heisenberg algebras (different realizations)	`nctools.lib`
• $U(\mathfrak{sl}_n)$, $U(\mathfrak{gl}_n)$, $U(\mathfrak{g}_2)$	`ncalg.lib`
• $U_q(\mathfrak{sl}_2)$, $U_q(\mathfrak{sl}_3)$, $U'_q(\mathfrak{so}_3)$	`ncalg.lib`
• $\mathcal{O}_q(\mathbb{A}^n)$, $\mathcal{O}_q(M_n(\mathbb{K}))$	`qmatrix.lib`
• exterior algebras and general fin.–dim. algebras	`nctools.lib`

4 New and Revised Functionalities in PLURAL

Revised and newly–implemented algorithms

- ▷ left Gröbner basis of a module w.r.t. any well–ordering
- ▷ completion of a two–sided generating set to the left Gröbner basis
- ▷ intersection with subalgebras (elimination of variables)
- ▷ intersection of a finite set of modules
- ▷ saturation and quotient of modules by ideals
- ▷ kernel of a module homomorphism
- ▷ free resolutions ("normal" and "minimized" algorithms)
  - ▶ syzygy modules of a given module
  - ▶ Betti numbers for graded modules over graded algebras
- ▷ center of an algebra (O. Motsak, `center.lib`)
  - ▶ centralizer of a finite set of polynomials
  - ▶ works over any field; in any proper factor–algebra
  - ▶ for example, the center of $U(\mathfrak{g}_2)$ is computable on usual PC
- ▷ algebraic dependence of pairwise commutative elements

Newly–developed and implemented algorithms in PLURAL

- ♢ central character decomposition of a module (`ncdecomp.lib`)
  - ♦ central quotient and central saturation of modules by ideals
  - ♦ central character of module
- ♢ opposite and enveloping algebras, "opposing" given object
- ♢ preimage of an ideal/module under a morphism
  - ♦ correctness of a given map of G–algebras
  - ♦ character of module w.r.t. a given (commutative) subalgebra
- ♢ annihilator of a holonomic (for example, Harish–Chandra) module
  - ♦ annihilator of an element from a module
  - ♦ annihilator of a finitely dimensional module

5 Collaborations and Perspectives

Contributed functionality

- V. Gerdt and D. Yanovich (Dubna, Russia)
  - ∘ Involutive (Janet) bases for well–orderings (kernel implementation)
- J. Gómez-Torrecillaz, F. Lobillo and C. Rabelo (Granada, Spain)
  - ∘ Gelfand–Kirillov dimension (`gkdim.lib`)
  - ∘ Quantum matrices and quantum minors (`qmatrix.lib`)
- G. Pfister and W. Decker (Germany)
  - ∘ Tate resolution and cohomology of sheaves (`sheafcoh.lib`)

Currently under development

- ♢ New efficient approach to bimodules; basic operations for right modules
- ♢ Homological algebra for finitely presented modules
  - ♦ for a left module L and a centralizing bimodule N, $\mathrm{Hom}(L,N)$, $\mathrm{Ext}(L,N)$ and $\mathrm{Tor}(N,L)$ are computable modules
  - ♦ Hochschild cohomology $H^n(A,M)$ for (A,A)–bimodule M

6 Contributors Are Welcome!

- Do you want to implement complicated algorithms in an efficient way?
- Have you ever thought of doing this based on SINGULAR?
- Contact us and let us develop together!

SINGULAR PLURAL