GInv: software for calculation of Gröbner involutive basis

Yury A. Blinkov[1, 2] Rustam E. Bayramov[2] Mikhail D. Malykh[2]

[1]Mechanics and Mathematics Department
Saratov State University, Saratov, Russia

[2]Department of Applied Probability and Informatics,
Peoples' Friendship University of Russia, Moscow, Russia

PCA 2022, St.Petersburg, May 02, 2022

Contents

Theory of involutive divisions

Riquier (1910), Janet (1920), Thomas (1937): Involutivity of PDEs.
Zharkov, Blinkov (1993): Pommaret Bases.
Gerdt, Blinkov (1995-1998): Involutive Division $\Longrightarrow$ Involutive Bases.
Apel (1998): Admissible Partial Division $\Longrightarrow$ Involutive Bases.
Gerdt (2001): Pair property of involutive division.
Gerdt, Blinkov, Yanovich (2001): Janet Trees for constructing Janet bases.
Chen, Gao (2002): Involutive Characteristic Sets for PDEs.
Hemmecke (2003): Sliced Involutive Division.
Evans (2004): Noncommutative Involutive Bases.
Semenov, Zyuzikov (2003-2008): Involutive Division via Monomial Ordering.
Gerdt, Blinkov (2005): Janet-like Division.
Chistov, Grigoriev (2007): Complexity of Janet Bases for D-modules.
Seiler (2009): Involutive Bases for Algebras of Solvable Type.
Gerdt (2008-2012); Bächler, Gerdt, Lange-Hegermann, Robertz (2012): based on Janet division Thomas Decomposition of Nonlinear PDEs.
Gerdt, Blinkov (2011): Involutive Division generated by antigraded ordering.
Hashemi, Parnian (2018): D-Noether division.

.....

Research problem: efficient construction of most compact involutive bases.

Implementation

Schwarz (1984): Riquier-Janet theory in Reduce.
Schwarz (1992): Linear differential Janet bases (DJB) in Reduce.
Zharkov, Blinkov (1993); G., Blinkov (1995): Pol. Pommaret bases (PPB) in Reduce.
Kredel (1996): PPB in MAS.
Nischke (1996): Polynomial JB (PJB) and PPB in C++ (PoSSoLib).
Berth (1999): Polynomial and differential involutive bases in Mathematica.
Cid (2000)-Robertz (2002-2011): PJB, DJB and difference JB in Maple.
Blinkov (2000-2007): PJB in Reduce, C++, GInv.
Yanovich (2001-2004): PJB in C, Singular.
Hausdorf, Seiler (2000-2002): DJB and DPB in MuPAD.
Chen, Gao (2002): Involutive extended characteristic sets in Maple.
Hemmecke (2002): Sliced division algorithm in Aldor.
Evans (2005): Noncommutative Involutive Bases in C.
Zhang, Li (2005): Janet bases for linear differential ideals in Maple.
Zinin (2007) - Blinkov (2011): Boolean Janet and Pommaret bases in C++, Reduce, Macaulay.
Langer-Hegermann (2010): Janet Bases for nonlinear and algebraically simple differential systems in Maple.
Albert (2012-2015): PPB and PJB in CoCoA.

.....

The open source software GINV implements the Gröbner bases

method for systems of equations. In the report, a new revised version of GInv will be presented. We use this system for analytical study of cubature formulas on a sphere known as Popov's problem.

The study of systems of nonlinear equations

in modern computer algebra systems is based on the calculation of Gröbner bases of ideals generated by the left-hand sides of the equations of these systems. The implementation of the Buchberger algorithm, which came to this system from the Singular system, is used. Buchberger's algorithm is the oldest, the basic version of Buchberger's algorithm leaves a lot of freedom in carrying out the computational process, thus considerable improvements are obtained by implementing criteria for reducing the number of S-polynomials to be actually considered (e.g., by applying the product criterion or the chain criterion).

GitHub, Inc. is a provider of Internet hosting for software development

and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features.

`https://github.com/blinkovua/GInv`

Example of using GInv

`https://github.com/blinkovua/GInv/blob/master/cython/GInv.ipynb`

Memory usage

in modern programming languages can be divided into three categories according to the time of their «life» in the program:

1. all the time «life», the so-called «static» memory, and variables using this memory are often called static. This also includes global ones.
2. is the execution time of the function, memory is allocated on the stack, and variables are called automatic, but in programming languages the word «local» is used.
3. the time of «life» is determined by the programmer if there is no «garbage collection» (garbage collection) **or** «garbage collection», if it exists in the selected programming language. Memory is allocated in «heap».

«Garbage collection»

was first applied by John McCarthy in 1959 in a programming environment based on the Lisp functional programming language he developed.

Subsequently, it was used in other programming systems and languages, mainly in functional and logical ones. Lists widely used in these languages and complex data structures based on them are constantly being created, built on, expanded, copied during the operation of programs, and it is difficult to correctly determine the moment of deleting an object.

Since the second half of the 1980s, garbage collection technology has been used in both directive (imperative) and object programming languages, and since the second half of the 1990s, an increasing number of created languages and environments focused on application programming include a garbage collection mechanism either as the only one or as one of the available mechanisms dynamic memory management. It is currently used in Oberon, Java, Python, Ruby, C#, D, F#, Go and other languages.

GC is «kitchen», «kitchen» and again «kitchen»

- Garbage collection consumes computing resources to decide which memory to free, even if the programmer may have already known this information.
- The moment when the garbage is actually collected can be unpredictable, which leads to stops (pauses for shifting/freeing memory) scattered throughout the session. Incremental, parallel garbage collectors and real-time garbage collectors solve these problems with various trade-offs.
- The most important thing: GC does not take into account «protected mode», «page organization of memory» and the most «secret weapon» of modern computers cache (super-operational memory).

International Symposium on Memory Management 2020

- Garbage Collection Using a Finite Liveness Domain
- Prefetching in Functional Languages
- Improving Phase Change Memory Performance with Data Content Aware Access
- ThinGC: Complete Isolation With Marginal Overhead
- Verified Sequential MallocFree
- Alligator Collector: A Latency-Optimized Garbage Collector for Functional Programming Languages
- Understanding and Optimizing Persistent Memory Allocation
- Exploiting Inter- and Intra-Memory Asymmetries for Data Mapping in Hybrid Tiered-Memories

GInv implementation

- My proposed approach is fundamentally different from the above and is based on the implementation of OOP in **C++**.
- Restrict the use of pointers to the internal structure of the **C++** class by declaring them in a closed scope. With the right OOP technology, and this property is called **encapsulation**, this should be done anyway, the user should not see how the class is implemented, he should use only its interface.
- Secondly, **C++** has a dedicated **copy constructor**, which allows you to build a copy of the object. It remains to define the **swap** function for the selected class, to replace the object with its copy, and you can organize dynamic memory reallocation to **C++**.
- Unlike small implementations of 11,000 lines of code, the implementation of the OO approach requires about 300 lines, including auxiliary templates **C++**.

Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz, cache size: 8192 KB

Cyclic7 (algorithm TQ)

	time	GC	GC/time %
Allocator	33.49 sec	0.11 sec	0.33%
Allocator (except for Integer)	33.88 sec	0.12 sec	0.35%
malloc/free	57.57 sec	—	—

Popov's problem 21 (algorithm TQ)

	time	GC	GC/time %
Allocator	545.78 sec	4.16 sec	0.76%
Allocator (except for Integer)	482.49 sec	1.17 sec	0.24%
malloc/free	498.68 sec	—	—

eco11 (algorithm TQ)

	time	GC	GC/time %
Allocator	200.85 sec	0.44 sec	0.22%
Allocator (except for Integer)	268.78 sec	0.07 sec	0.03%
malloc/free	341.10 sec	—	—