The software GInv for calculating involutive bases of polynomial ideals

PCA'2024

Yuri Blinkov, Soltan Salpagarov and **Anton Mamonov**

Peoples' Friendship University of Russia, Moscow, Russia
Saratov State University, Saratov, Russia

Apr. 19, 2024

Introduction

The study of problems in mechanics and mathematical physics is often reduced to solving systems of polynomial equations. This problem has no universal numerical solution method in the cases of two and more unknowns.
Finding the Gröbner basis allows to reduce the solution of a system of nonlinear equations with a finite number of solutions to the solution of one equation with one unknown*. The Buchberger's algorithm allows to find the Gröbner basis in a finite number of steps, but in practice it can be used to solve systems only of a small degree and with no more than a dozen unknowns.
GInv is the software that uses a specific approach to finding the bases of polynomial ideals based on the original concept of involutive division.
*Ref: D. Cox, J. Little, and D. O'Shea, // *Ideals, varieties, and algorithms*. Springer, 3 edition, 2007

Gröbner basis

Definition

Gröbner basis, for ideal I of a ring $F[x_1, \cdots, x_n]$ is a finite set $G = \{g_1, \cdots, g_m\}$ of polynomials from F, that generates an ideal I and for any $p \in I$ there is polynomial $g \in G$ such that leading term of p is a multiple of leading term of g

To operate with Gröbner basis, one need to choose the monomial ordering. This order dictates leading terms, monomials and coeficients. There are different ways to define an order, but the three main ones are:

- Lexicographical ordering, (lex).
- Total degree reverse lexicographical ordering (degrevlex).
- Elimination ordering (lexdeg).

Buchberger's algorithm

To find a basis for an ideal I of a polynomial ring R with F being a set of polynomials that generates I we need to:

1. For every pair of polynomials f_i, f_j in F, let g_i be the leading term of f_i in the given monomial ordering, and a_{ij} – the least common multiple of g_i and g_j.

$G = F$.

2. For each pair of polynomials in G let $S_{ij} = \frac{a_{ij}}{g_i} f_i - \frac{a_{ij}}{g_j} f_j$.

3. Reduce S_{ij}, with the multivariate division algorithm relative to the set G until the result is not further reducible. If the result is non-zero, add it to G.

4. Repeat steps 2 and 3 with every possible pair, including those with the new polynomials added by step 3.

Now G is a Gröbner basis for ideal I of a polynomial ring R.

Computational complexity

Let $R[x_1, ..., x_n]$be a ring of multivariate polynomials with coefficients in a field R, and let F be a subset of this ring such that d is the maximum total degree of any polynomial in F. Then for any admissible ordering, the total degree of polynomials in a Gröbner basis for the ideal generated by F is bounded by $2((d^2/2) + d)^{2^{n-1}}$, where n is the number of variables, and d the maximal total degree of the polynomials.
This bound allows, in theory, to use linear algebra over the vector space of the polynomials of degree bounded by this value, for getting an algorithm of complexity $d^{2^{n+o(1)}}$

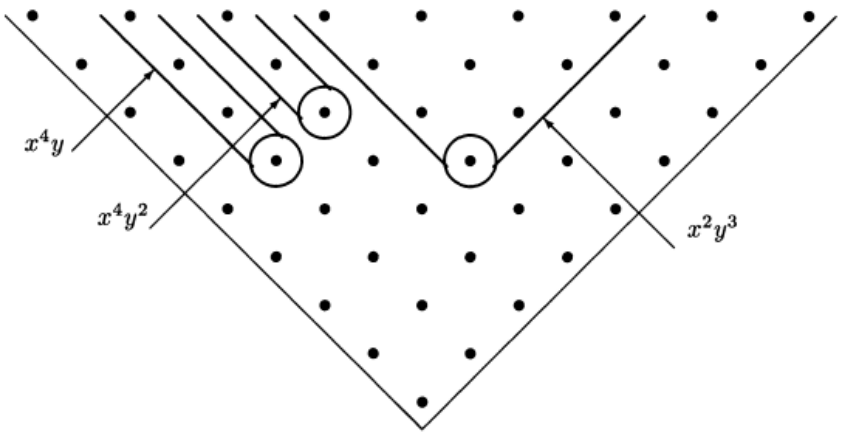
Ref: Thomas W. Dubé // *IThe Structure of Polynomial Ideals and Gröbner Bases*, SIAM Journal on Computing 1990 19:4, 750-773, DOI 10.1137/0219053

Faugère's algorithms

Buchberger algorithm was optimised by Jean-Charles Faugère, in Faugère F4 algorithm. The algorithm uses the same mathematical principles as the Buchberger algorithm, but computes many normal forms in one go by forming a generally sparse matrix and using fast linear algebra to do the reductions in parallel.
Furthermore, the Faugère F5 algorithm first calculates the Gröbner basis of a pair of generator polynomials of the ideal. Then it uses this basis to reduce the size of the initial matrices of generators for the next larger basis.
Both this algorithms are currently implemented in commercial systems, for example, Maple computer algebra system, and in some study versions in SageMath CAS and in SymPy package.

Involutive devision

At the same time, Gerdt, Zharkov and Blinkov [3] proposed a new approach to finding the bases of polynomial ideals based on the original concept of involutive division. Involutive division helps to chose some, which removes the problem of many variants for polinomial reductions. It applies restriction on monomial choice, which can be represented by conic forms:



Conic representation.

In the 2000s, this algorithm was implemented in the form of GInv software and applied to the study of a number of problems in mathematical physics [4].

Program structure

GInv system kernel is developed on C++. This allows a more direct approach for memory usage optimisation. Based on C++ module, a Python interface is created, with Cython as a transmission tool.

It can be deployed on Linux systems, (Debian, Ubuntu, Fedora, e.t.c.), and have a defined requerments list, present on repisitory [5].

Open version is easily obtainable at Git-Hub, `https://github.com/blinkovua/GInv`

Testing module

A testing utility has been developed, in the form of an .ipynb script. The developed utility makes it possible to significantly simplify the GInv testing process by automatically downloading test data in JSON format and running calculations without the need to manually enter parameters.

With already deployed GInv in our environment, we only need o add import command **from ginv import *** inside a code notebook.

For testing GInv, and other CAS, a set of computational tests was obtained. This set contains 135 JSON files, with different classical and original testing problems.

Module output

```
from IPython import display
import json
from pprint import pprint, pformat
from functools import reduce, cmp_to_key
from ginv import *
from zipfile import ZipFile
import re
import time
import pandas as pd
```

```
def receiving_json(test_name, json_data):
    try:
        size = json_data["dimension"]
        print(f"Тест для {test_name}")
        variables_list = json_data["variables"]
        array = [poly_int.to_monom("TOP", "deglex", monom.variable(i, len(variables_list), -1)) for i in range(len(variables_list))]

        for i in range(len(variables_list)):
            globals()[variables_list[i]] = array[i]

        eqs = json_data["equations"]
        new_expressions = [re.sub(r'\^', '**', expression) for expression in eqs]
        print(new_expressions)
        eqs = [eval(expr) for expr in new_expressions]
        ginv(eqs, variables_list, size)

    except Exception as e:
        print(f"Error processing JSON: {e}")
        return
```

JSON reading function.

Module output

```
тест 6:
Тест для chemkin.json
['-4*w*y2+9*y2**2+z2', 'x3**2+y3**2+z3**2-1', 'x4**2+y4**2+z4**2-1', '9*y5**2+9*z5**2-8', '-6*w*x
3*y2+3*x3+3*y2*y3+3*z2*z3-1', '3*x3*x4+3*y3*y4+3*z3*z4-1', 'x4+3*y4*y5+3*z4*z5-1', '-6*w+3*x3+3*x
4+8', '9*y2+9*y3+9*y4+9*y5+8', 'z2+z3+z4+z5', 'w**2-2']
решаем систему...
```

Some tests reading.

```
--------------------
Название задачи: dessin1.json
Длина базиса: 104
Время: 167.77348518371582
Количество переменных: 8
--------------------
Название задачи: hcyclic6.json
Длина базиса: 221
Время: 308.4113597869873
Количество переменных: 7
--------------------
Название задачи: dl.json
Длина базиса: 1
Время: 317.80552864074707
Количество переменных: 8
--------------------
```

```
--------------------
Название задачи: benchmark_i1.json
Длина базиса: 243
Время: 50898.650884628296
Количество переменных: 10
--------------------
Название задачи: i1.json
Длина базиса: 243
Время: 51117.50149726868
Количество переменных: 12
--------------------
Название задачи: hcyclic7.json
Длина базиса: 1182
Время: 56018.998861312866
Количество переменных: 8
--------------------
```

```
--------------------
Название задачи: hairer2.json
Длина базиса: 547
Время: 83530.93075752258
Количество переменных: 13
--------------------
Название задачи: cyclic7.json
Длина базиса: 1442
Время: 91115.35978317261
Количество переменных: 7
--------------------
Название задачи: ilias_k_3.json
Длина базиса: 476
Время: 196373.32940101624
Количество переменных: 8
--------------------
```

Some tests output.

JSON samples: hcyclic6

"variables": ["x1", "x2", "x3", "x4", "x5", "x6", "w"],
"equations": ["x1+x2+x3+x4+x5+x6",
"x1*x2+x1*x6+x2*x3+x3*x4+x4*x5+x5*x6",
"x1*x2*x3+x1*x2*x6+x1*x5*x6+x2*x3*x4+x3*x4*x5+x4*x5*x6",
"x1*x2*x3*x4+x1*x2*x3*x6+x1*x2*x5*x6
+x1*x4*x5*x6+x2*x3*x4*x5+x3*x4*x5*x6",
"x1*x2*x3*x4*x5+x1*x2*x3*x4*x6+x1*x2*x3*x5*x6
+x1*x2*x4*x5*x6+x1*x3*x4*x5*x6+x2*x3*x4*x5*x6",
"-w^6+x1*x2*x3*x4*x5*x6"],
"description": ...
"dimension": 7

JSON samples: hcyclic7

"variables": ["x1", "x2", "x3", "x4", "x5", "x6", "x7", "w"],
"equations": ["x1+x2+x3+x4+x5+x6+x7",
"x1*x2+x1*x7+x2*x3+x3*x4+x4*x5+x5*x6+x6*x7",
"x1*x2*x3+x1*x2*x7+x1*x6*x7+x2*x3*x4
+x3*x4*x5+x4*x5*x6+x5*x6*x7",
"x1*x2*x3*x4+x1*x2*x3*x7+x1*x2*x6*x7+x1*x5*x6*x7
+x2*x3*x4*x5+x3*x4*x5*x6+x4*x5*x6*x7",
"x1*x2*x3*x4*x5+x1*x2*x3*x4*x7+x1*x2*x3*x6*x7+x1*x2*x5*x6*x7
+x1*x4*x5*x6*x7+x2*x3*x4*x5*x6+x3*x4*x5*x6*x7",
"x1*x2*x3*x4*x5*x6+x1*x2*x3*x4*x5*x7+x1*x2*x3*x4*x6*x7
+x1*x2*x3*x5*x6*x7+x1*x2*x4*x5*x6*x7
+x1*x3*x4*x5*x6*x7+x2*x3*x4*x5*x6*x7",
"-w^7+x1*x2*x3*x4*x5*x6*x7"],
"description": ...
"dimension": 8

JSON samples: ilias13

"variables": ["S1", "s1", "d1", "S2", "D2", "s2", "d2"],
"equations": ["4*(5*D2∧2+2*S1*S2-4*S1-S2∧2+4)",
"2*(20*d1*D2+5*d2*D2-16*s1*S1+8*s1*S2+16*s1+2*s2*S1-s2*S2-2*s2+8*S1-4*S2-8)",
"2*(8*d1*S1-4*d1*S2-8*d1+2*d2*S1-d2*S2-2*d2+5*D2*s2-20*D2)",
"4*(-5*D2∧2-2*S1*S2+4*S1+S2∧2-4)",
"2*(4*d1*S2-8*d1+d2*S2-2*d2+8*D2*s1-D2*s2-4*D2)",
"2*(-4*d1*D2-d2*D2+s2*S2-2*s2-4*S2+8)",
"2*(-20*d1*D2-5*d2*D2+16*s1*S1-8*s1*S2-16*s1-2*s2*S1+s2*S2+2*s2-8*S1+4*S2+8)",
"2*(-4*d1*S2+8*d1-d2*S2+2*d2-8*D2*s1+D2*s2+4*D2)",
"-16*d1∧2-8*d1*d2-d2∧2-8*s1*s2+32*s1+s2∧2-16",
"2*(-8*d1*S1+4*d1*S2+8*d1-2*d2*S1+d2*S2+2*d2-5*D2*s2+20*D2)",
"2*(4*d1*D2+d2*D2-s2*S2+2*s2+4*S2-8)",
"16*d1∧2+8*d1*d2+d2∧2+8*s1*s2-32*s1-s2∧2+16",
"4*d1∧3*D2∧2-4*d1∧3*D2*S1+d1∧3*S1∧2+12*d1∧2*D2∧2*s1-
12*d1∧2*D2*s1*S1+3*d1∧2*s1*S1∧2+12*d1*D2∧2*s1∧2-
12*d1*D2*s1∧2*S1+3*d1*s1∧2*S1∧2+4*D2∧2*s1∧3-4*D2*s1∧3*S1+s1∧3*S1∧2-32",
"-4*d1∧3*D2∧2-4*d1∧3*D2*S1-
d1∧3*S1∧2+12*d1∧2*D2∧2*s1+12*d1∧2*D2*s1*S1+3*d1∧2*s1*S1∧2-12*d1*D2∧2*s1∧2-
12*d1*D2*s1∧2*S1-3*d1*s1∧2*S1∧2+4*D2∧2*s1∧3+4*D2*s1∧3*S1+s1∧3*S1∧2-32",
"-d2∧3*D2∧2-2*d2∧3*D2*S2-d2∧3*S2∧2+3*d2∧2*D2∧2*s2+6*d2∧2*D2*s2*S2+3*d2∧2*s2*S2∧2-
3*d2*D2∧2*s2∧2-6*d2*D2*s2∧2*S2-3*d2*s2∧2*S2∧2+D2∧2*s2∧3+2*D2*s2∧3*S2+s2∧3*S2∧2-
32",
"d2∧3*D2∧2-2*d2∧3*D2*S2+d2∧3*S2∧2+3*d2∧2*D2∧2*s2-
6*d2∧2*D2*s2*S2+3*d2∧2*s2*S2∧2+3*d2*D2∧2*s2∧2-
6*d2*D2*s2∧2*S2+3*d2*s2∧2*S2∧2+D2∧2*s2∧3-2*D2*s2∧3*S2+s2∧3*S2∧2-32"],
"description": null,
"dimension": 7

Hardware

The system was tested on a server platform consisting of two 4-core Intel Xeon L5630 processors. Each processor had 4 computing cores with support for Hyper-Threading technology, which allowed running 2 threads on each physical core. Thus, the total number of logical cores (processing threads) was 8. The base clock frequency of each processor core was 2134 MHz. Some results of this testing are displayed on following table 1.

Sample of test results

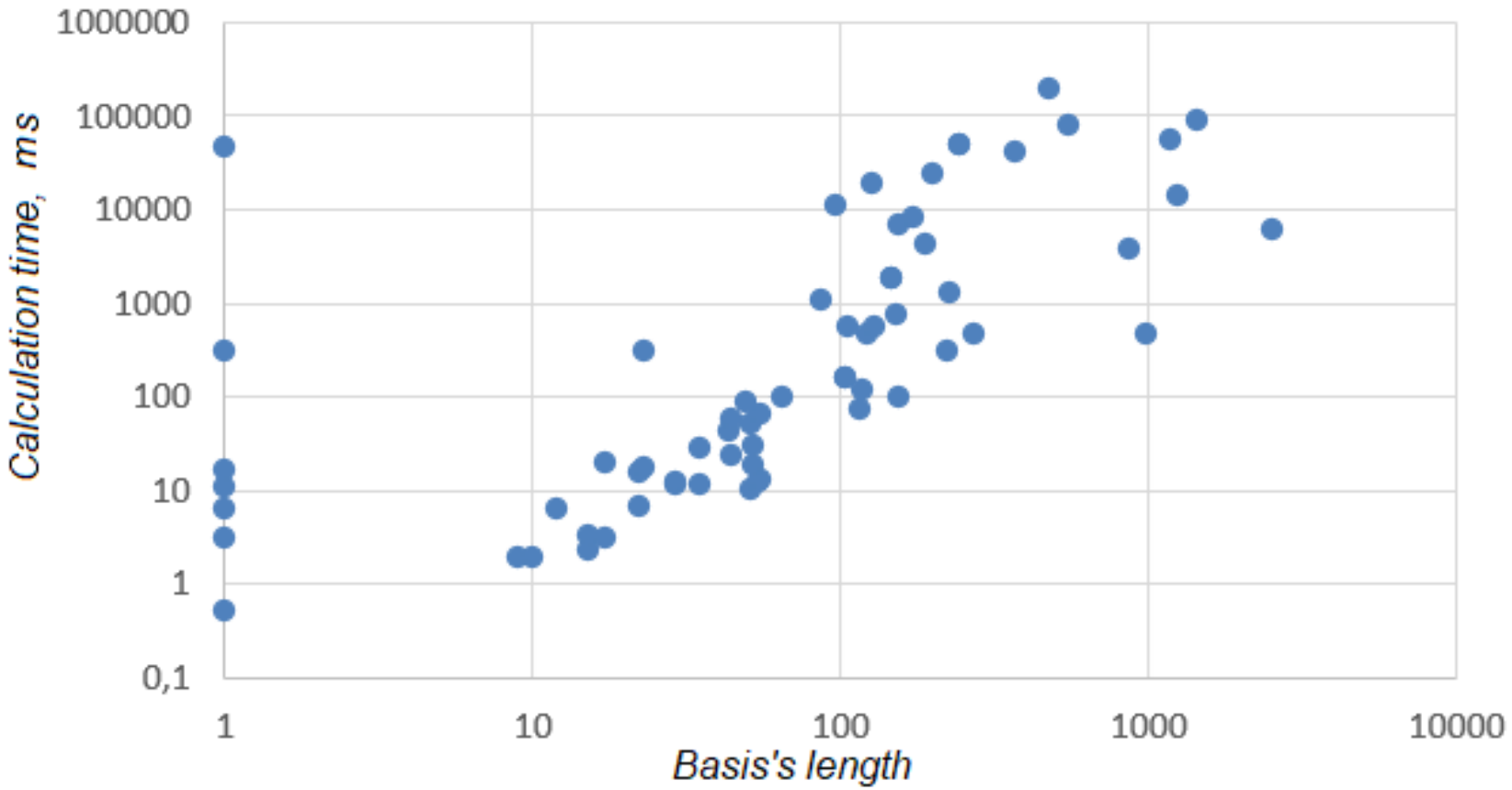


Fig. 1. Calculation time for tests

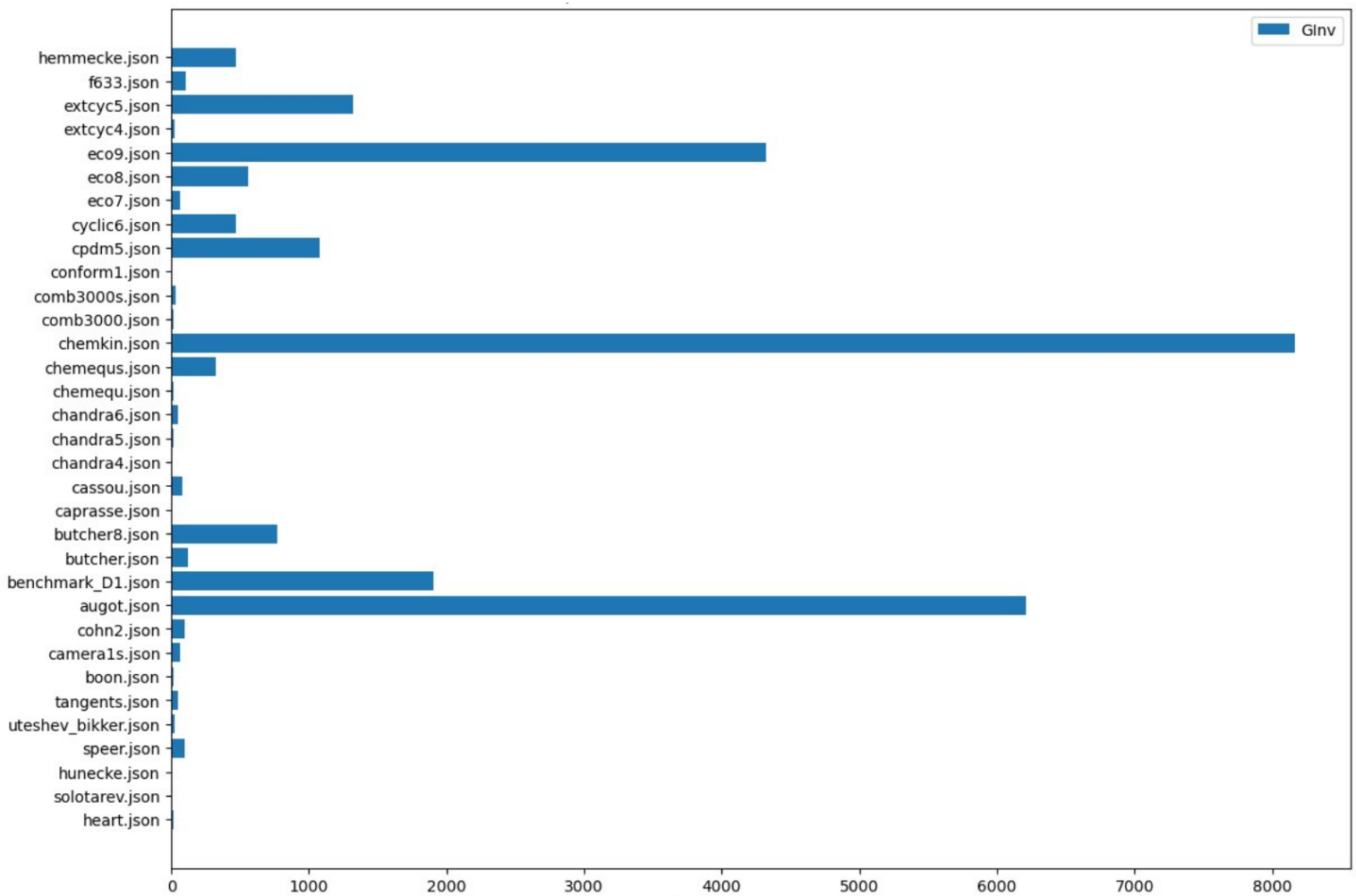
Sample of test results

Test	Dimension	Basis's length	Reduction	Time (ms)
ilias13	7	1	0	0,53
comb3000	10	35	503	11,70
hcyclic6	7	221	18634	308,41
eco9	9	189	159992	4322,05
hcyclic7	8	1182	542213	56018,99

Table: Sample of test results

Output results (basis's length, reduction, calculation time), *mostly* have exponential correlation. Same could not be said for input parameters (dimension, lead degree).

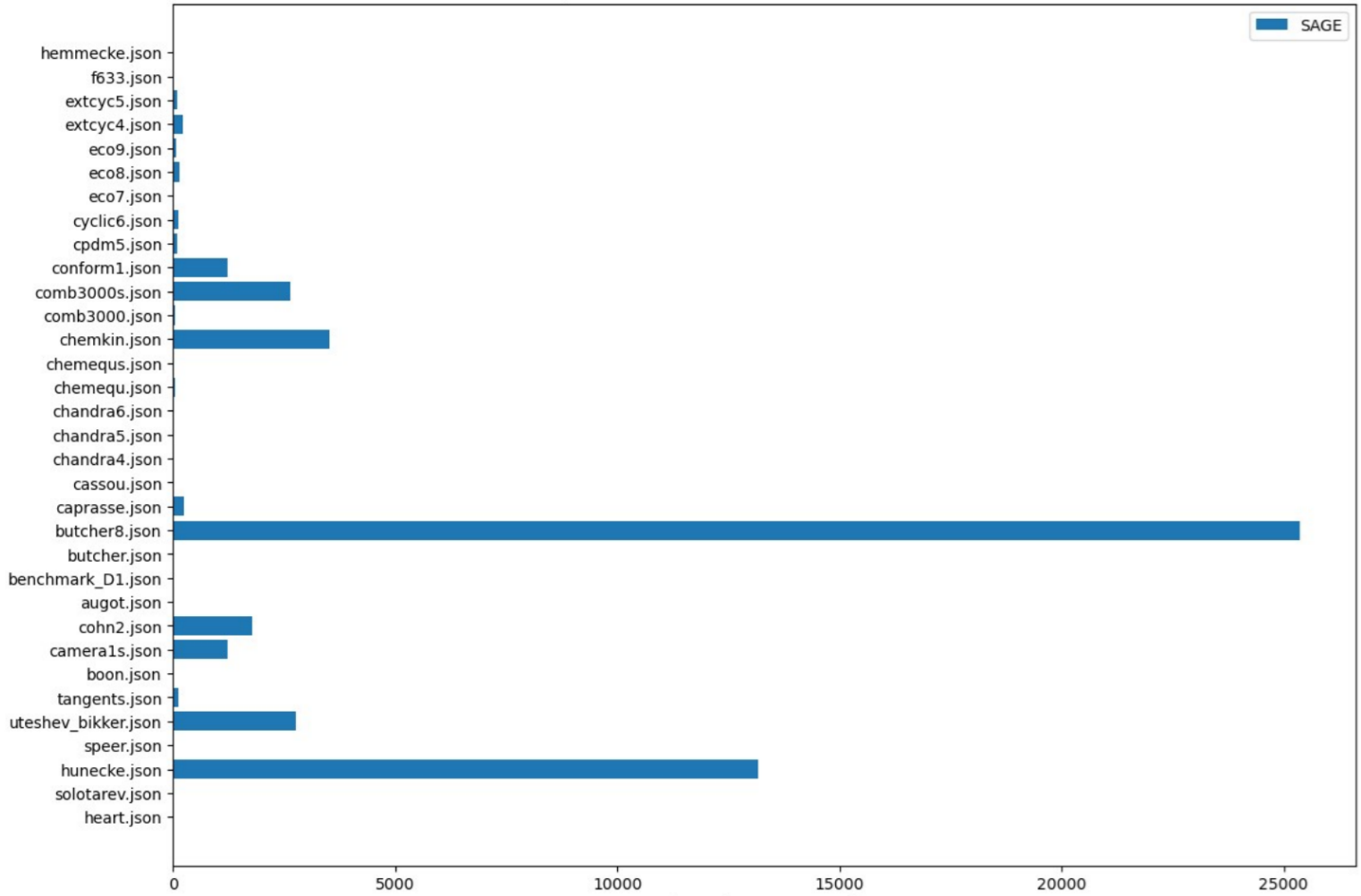
Comparing GInv and Sage



Calculation time for GInv.

Average time: 800 ms.

Comparing GInv and Sage



Calculation time for Sage.

Average time: 1600 ms.

Further perspectives

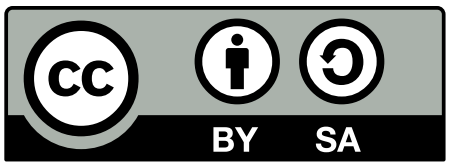
- Expanding test set
- Memory usage monitoring
- Defining usage boundaries
- Comparing against commercial algorithms
- Compiling standalone python package

Repositories

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

URL: `https://github.com/MamonovAnton/ginv_testing`

The End




W. H. Press et al., *Numerical recipes in C : the art of scientific computing*. 2nd edition, ISBN 0-521-43108-5, 1992.


D. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and algorithms*. Springer, 3 edition, 2007


A. Yu. Zharkov and Yu. A. Blinkov, *Laboratoire d'Informatique Fondamentale de Lille*. France, 1993


Yu. A. Blinkov, V. P. Gerdt, *Programming*. 2008.


Yu. A. Blinkov, URL:https://github.com/blinkovua/GInv.


Bayramov R. E. et al, URL:https://github.com/blinkovua/GInv. *Analytical study of cubature formulas on a sphere in computer algebra systems*. Journal of Computational Mathematics and Mathematical Physics, 2023; 63(1): 93-101. DOI: 10.31857/S0044466923010052.


Yu. A. Blinkov, A. A. Mamonov, URL:https://github.com/MamonovAnton/ginv_testing.