A New Algorithm for Computing Certified Numerical Approximations of the Roots of a Zero-dimensional System

Stef Graillat
UPMC/CNRS LIP6, PEQUAN team
104 avenue du Président Kennedy
75016 Paris (France)
Stef.Graillat@lip6.fr

Philippe Trébuchet
UPMC/CNRS LIP6, APR team
104 avenue du Président Kennedy
75016 Paris (France)
Philippe.Trebuchet@lip6.fr

ABSTRACT

This paper provides a new method for computing numerical approximations of the roots of a zero-dimensional system. It works on general systems, even those with multiple roots, and avoids any arbitrary choice of linear combination of the multiplication operators. It works by computing eigenvectors (or a basis of the full invariant subspaces). The sparsity/structure of the multiplication operators by one variable can also be taken into account.

Categories and Subject Descriptors

G.0 [**Mathematics of Computing**]: General; G.1 [**Numerical Analysis**]: General—*Numerical algorithms, Interval arithmetic, Multiple precision arithmetic*; G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*Systems of equations, Error analysis*

General Terms

Algorithms

Keywords

Multivariate polynomial, normal form, quotient algebra, root-finding, symbolic-numeric computation

1. INTRODUCTION

Let $\mathbb{K}$ a field (generally $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$) and $f_1, \ldots, f_s \in \mathbb{K}[\mathbf{x}]$ be multivariate polynomials whose common zero set is composed of finitely many points. The algebraic polynomial system solving process mainly relies on two time-consuming steps :

- The first one is to compute a representation of the quotient algebra $A = \mathbb{K}[\mathbf{x}]/(f_1, \ldots, f_s)$.
- The second one is to derive some information about the roots from the previous representation.

ISSAC'09, July 28–31, 2009, Seoul, Republic of Korea.

The paper will only focus on the second step. More precisely, the purpose of this paper is to compute a certified numerical approximation of the solutions. To obtain this information from the representation of A, only four strategies are currently available :

- Following [14], one can compute a Rational Univariate Representation (RUR), which is a symbolic representation of the roots. As a symbolic representation, its computation involves computations with big coefficients, and furthermore, it involves the computation of the full multiplication table in the quotient algebra, which is costly. Afterwards it remains to compute effectively the required numerical approximation, which is done, for instance, via the application of Uspensky's algorithm.
- An algorithm based on homotopic continuation method (see [19]) can be very efficient.The problem is that there is no certification even for the number of roots.
- Following [5], one can compute a basis which makes simultaneously triangular all the multiplication operators by one variable in A. Modulo a heuristic of clustering, it only remains to read off the diagonals the coordinates of the roots. For computing the basis, one can use a *generic* linear combination of the multiplication operators by one variables. Unfortunately there is no way to check that a given linear form is generic or not. Moreover, there is no way to check that the clustering heuristic performed well or not. Finally no concern is given in this paper [5] for certifying the precision eventually obtained. Another drawback of this method is that the linear combination of the multiplication operators produces in general a dense matrix, even though the multiplication operators by one variable are sparse separately.
- Using the eigenvalues/eigenvectors of the multiplication operators separately to recover the information about the roots. In [11] two algorithms are suggested, either computing the eigenvalues of $M_{x_{i+1}}$ (the matrix multiplication by x_{i+1}) restricted to the eigenspaces of M_{x_i}, or using the property that the common eigenvectors to all the multiplication operators are the evaluation operators at the roots.

To sum up, the first method transforms the problem of solving a polynomial system into the problem of accurately computing the roots of an univariate polynomials [14]. The

third and fourth methods transforms the problem of solving a polynomial system into the pure linear algebra problem of finding eigenvalues and eigenvectors [5, 6, 9, 10, 11].

In this paper we provide an acceleration of the fourth method and extend it to get a numerical approximation of the roots from the multiplication operators. Our routine performs without any generic choice and more importantly gives a certified result.

2. PRELIMINARIES

Given B be a set of vectors, we will hereafter denote by $\langle B \rangle$ the vector space spanned by the vectors of B.

In the rest of the paper we will assume as given :

- a field $\mathbb{K}$ ($\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$),
- a list $f_1, \ldots, f_s$ of multivariate polynomials of $\mathbb{K}[\mathbf{x}]$ generating a zero-dimensional ideal (with $\mathbf{x} = x_1, \ldots, x_n$),
- a monomial basis B of $A = \mathbb{K}[\mathbf{x}]/(f_1, \ldots, f_s)$,
- a function that allows us to compute the multiplication operator $M_i : \langle B \rangle \to \langle B \rangle$
  $p \to x_i p$.

As the whole paper is devoted to the study of the quotient algebra $A = \mathbb{K}[\mathbf{x}]/(f_1, \ldots, f_s)$, and for convenience, we will identify a vector of $\langle B \rangle$ with a polynomial of A.

The operators we will be dealing with are operators that belong to the dual space $\langle \hat{B} \rangle$. Let m be a monomial of the monomial basis of B, and let v be a vector of the vector space spanned by B, i.e. $\langle B \rangle$, we will call *the coordinate of m in v*, the component v_i of v such that $v - v_i m$ is a polynomial whose support does not contain the monomial m.

First of all let us recall a well known result proved in [11] and that can be found in [7, Thm 4.23,p.87].

THEOREM 2.1. *The common eigenvectors to all the transposed multiplication operators are the evaluation at the roots.*

From this one easily derive the following result.

THEOREM 2.2. *The common eigenvectors to all the transposed multiplication operators by one variable, i.e. x_i, are the evaluation at the roots.*

3. A FIRST ALGORITHM IN THE EXACT CASE

Most of the material presented in this section was already presented in [11], but for the sake of completeness we recall the main points.

Hereafter, we will be concerned with the dual operators of the operators from A to A (recall that $A = \mathbb{K}[\mathbf{x}]/(f_1, \ldots, f_s)$) that are associated with the multiplication by one variable. There are n such dual operators.

Let us now state the algorithm, where it is assumed that there is no numerical error done and where we assume that we are always able to decide whether or not a certain vector is an eigenvector.

ALGORITHM 1 (UNDERNUM).

INPUT : *$lm = (M_1, \ldots, M_n)$ the dual multiplication operators*
i an integer index
lv a list a vectors expressed on the canonical basis
OUTPUT : *A numerical approximation of a common eigenvector to all the M_i.*

- *$Sol = []$*
- *Compute M, the matrix of the restriction of M_i on the vector space spanned by the vectors of lv.*
- *For each eigenvalue v of M do*
  - *if the eigenspace associated to v has dimension 1 then*
    - *Let e be the eigenvector of the eigenspace.*
    - *Let M_{lv} be the matrix whose columns are the vectors of lv.*
    - *$e' = M_{lv} e$*
    - *$Sol = Sol \cup [e']$*
  - *else*
    - *Let le denote the list of eigenvectors associated to v.*
    - *$Sol = Sol \cup$ `Undernum`$(lm, i+1, le)$*
- *Return Sol*

Given Algorithm `Undernum`, we can now present Algorithm `Symbonum` which computes the roots of a polynomial system thanks to the eigenvalues computed by `Undernum`.

ALGORITHM 2 (SYMBONUM).

INPUT : *$M_1, \ldots, M_n$ the n transposed multiplication operators*
OUTPUT : *A numerical approximation of the roots of the system $f_1, \ldots, f_s$.*

- *$Res = []$*
- *Let C be the list of the vectors of the canonical basis.*
- *$Tmp\_sol =$ `Undernum`$([M_1, \ldots, M_n], i, C)$*
- *For each v in Tmp_sol do*
  - *tmpres=[]*
  - *for i from 1 to n do*
    - *$tmpres = tmpres \cup DotProd(Row(1, M_i), v/v[1])$*
  - *$Res = Res \cup tmpres$*
- *return Res*

In the previous algorithm, *DotProd* computes the dot product of two vectors. The function $Column(i, M)$ returns the i-th column of the matrix M.

THEOREM 3.1. *During all the calls to* `Undernum` *the transposed multiplication operator M_i leaves the space spanned by lv globally invariant.*

Proof. The first time `Undernum` is called, the space spanned by the vectors of lv is the whole space so M_1 leaves it globally invariant. During the first level of recursion call to `Undernum` the vectors of lv denote a basis of an eigenspace of M_1, so as M_1 and M_2 commute, M_2 leaves these spaces globally invariant.

For treating the second level of recursion, let E_2 be the vector space spanned by the vectors of lv at this level, and E_1 the vector space on which the restriction of M_2 has been computed at the first level of recursion. E_1 is left globally invariant by M_3 and as M_3 and M_2 commute, this eigenspace of the restriction of M_2 on E_1 is left globally invariant by M_3. Hence the matrix of the restriction of M_3 to E_2 can be computed.

The same argument applies for treating the higher level of recursion. □

THEOREM 3.2. *The algorithm* `Undernum` *stops after a finite number of operations and produces a numerical approximation of the common eigenvectors to all the multiplication operators by one variable.*

Proof. Let us consider here the space E spanned by the vectors of the list lv for a certain recursive call. From the previous theorem, E is left globally invariant by all the M_i. Hence it is meaningful to compute the matrix of the restriction of these operators on E.

Remark now that the eigenvectors found after n levels of recursion are necessarily common to all the multiplication operators by one variable. Remark also that an eigenvector v whose associated eigenvalue is of geometric multiplicity one, i.e. an eigenvector that is stored in our *Sol* variable, is a common eigenvector to all the multiplication operators by one variable : if the associated eigenvalue has geometric multiplicity one, then the eigenspace is of dimension one. However this space is left globally invariant by the other multiplication operators, that is to say v is also an eigenvector to the other multiplication operators. In other words, v is a common eigenvector to all the multiplication operators.

Hence, after at most n recursion calls, all the common eigenvectors to all the multiplication operators will be found. In other words after at most n recursions, we have computed the solutions of the system $f_1, \ldots, f_s$. □

First of all remark that in many applications the first variable is a separating element (the first coordinate of all the distinct roots are different), hence in the previous algorithm only one computation of eigenvectors is undertaken which is very efficient. Though this latter remark reveals one reason to consider this algorithm, this method has several drawbacks :

- there is no certification of what is computed,
- numerically speaking the eigenvectors are not well defined,
- the algorithm requires the computation of all the multiplication operators by all the variables.

4. NUMERICAL METHODS

In this section, we present an optimization of the algorithm presented in the previous section.

4.1 Numerical observations

Recently, certified algorithms [13, 15] have been designed for computing eigenvalues. Precisely, these methods perform using controlled arithmetic and compute an inclusion box of the eigenvalues/eigenvector. However, in case of eigenvalues of geometric multiplicity greater than one, it is impossible to get bounds on the computed eigenvectors. The following theorem from [16, Lemma 2.1], essentially states this.

THEOREM 4.1. *For $A \in M_n(\mathbb{K})$, let an eigenvalue $\lambda \in \mathrm{Spec}(A)$ (*Spec *denotes the spectrum) be given with algebraic multiplicity m and let $y \neq 0$, be a vector of $\mathbb{K}^n$ such that $Ay = \lambda y$, i.e. y is an eigenvector associated to λ. Then for all $\epsilon > 0$ there exists $\tilde{A}$ such that $||A - \tilde{A}||_\infty \leq \epsilon$ and the following properties hold :*

- $\lambda \in \mathrm{Spec}(\tilde{A})$.
- *λ is of algebraic multiplicity m.*
- *λ is of geometric multiplicity one.*
- $\tilde{A}y = \lambda y$.

This theorem essentially states that a floating point computation of the eigenvectors of a matrix, essentially computes bases of its full-invariant subspaces, and not only of its eigenspace. It also states that using floating point operations, one cannot recover the common eigenvectors to all the multiplication operators.

Fortunately, the full invariant subspaces of one multiplication operator are stable by the other ones and contains, of course, the eigenvectors we are looking for. Furthermore, one could also notice that they also provide informations about the algebraic multiplicity about the roots, even when the considered eigenvalue is defective. Dealing with defective eigenvalues and still recovering the algebraic multiplicity is the subject of a further work.

4.2 Clusterization

In what follows, we will be faced with the problem of grouping together eigenvalues and vectors associated to these eigenvalues. From the numerical point of view, only non defective eigenvalues exist, rounding errors make defective eigenvalues appear as non defective but with associated eigenvectors that are quite close. Hence in what follows we will denote by eigenvectors the vectors provided by the QR-multishift routine.

Numerical roundoff errors also make eigenvalues of algebraic multiplicity greater than one appear as many close eigenvalues.

What we want to do is grouping together these eigenvalues that are quite close and their associated eigenvectors. Moreover in the case of defective eigenvalues, clusterizing also means extracting a numerical basis of such a group of vectors. The danger if this latter operation is not performed is to loose all the meaningful information in the remaining operations.

For performing the grouping operations we use a well known clustering technique based on LAPACK error estimators and which can also be found in [5].

Consider an eigenvalue α, and v and u be its associated left eigenvector and right eigenvector : $Au = \alpha u$ and $v^t A = \alpha v$. The reciprocal condition number of α is the quantity

$$\mathrm{rcond}_\alpha = \frac{|v.u|}{\|v\| \cdot \|u\|},$$

where $u.v$ denotes the dot product of the vector u and v and $\|\cdot\|$ represents the Euclidean norm of the vectors.

The reciprocal condition number measures the sensitivity of the computed eigenvalue toward small perturbations of the input matrix; it may be seen as some sort of first derivative of the function that takes a matrix and returns its eigenvalues.

We consider that two eigenvalues α_i, α_j are the same if and only if their difference is smaller than the quantity

$$\max(\frac{prec \times \|V_{\alpha_i}\|}{\text{rcond}_{\alpha_i}}, \frac{prec \times \|V_{\alpha_j}\|}{\text{rcond}_{\alpha_j}}).$$

Using the informal characterization of the upper paragraph, one can easily see that this formula for error estimate is just the precision to which the matrix is known times the norm of the matrix (i.e. the precision to which the coefficients are known inside the computation times the sensitivity of the computation with respect to perturbations).

What we must say here is that these estimators are only estimators and can be arbitrarily wrong. We must also mention that this wrong behavior hardly ever happen in practice.

From what precedes one can see that the method for making clusters is only a heuristic and that there is nothing certified at that point!

We can also notice that contrary to the usual exposition, we did not here used ϵ, the machine precision, but *prec* which is the precision to which the matrix is known. Indeed consider what happens during one of the recursive call to `Undernum` : we express a small matrix based on approximate vectors, hence the exact eigenvalues of the approximate matrix could be known to be separated while they should be the same if the initial vector would have been known exactly.

4.3 Modified Algorithm

In the following proposition we say that a common eigenvector v to all the multiplication operators by one variable is normalized if its first coordinate v is 1.

PROPOSITION 4.2. *Suppose that $x_1 \in B$, and let e be an eigenvalue of $M^t_{x_1}$ of algebraic multiplicity k, then the normalized common eigenvectors to all the transposed multiplication operators that lie in the full invariant subspace associated to e have their coordinate in x_1 equal to e.*

Proof. The proof can be found in [7, p.83]. □

The previous proposition gives some information about the structure of the common eigenvectors we are looking for. More precisely, the following theorem gives precise information about the structure of the eigenvectors we are looking for.

THEOREM 4.3. *Let $\alpha_1, \ldots, \alpha_k$ be eigenvalues of respectively $M^t_{x_1}, \ldots, M^t_{x_k}$. Consider the vector space*

$$E = \text{Eig}(M^t_{x_1}, \alpha_1) \cap \text{Eig}(M^t_{x_2}, \alpha_2) \cap \cdots \cap \text{Eig}(M^t_{x_k}, \alpha_k)$$

where $\text{Eig}(M^t_{x_i}, \alpha_i)$ denotes the full invariant subspace of the matrix $M^t_{x_i}$ associated to the eigenvalue α_i. Let m be a monomial of the monomial basis B such that $m = x_1^{d_1} \cdots x_k^{d_k}$ Then the common eigenvectors to all the transposed multiplication operators that belong to E are such that : the coordinate of m in these vectors is $\alpha_1^{d_1} \cdots \alpha_k^{d_k}$.

Proof. The considered vectors are the operators of evaluations at the roots. Let $\zeta = (\zeta_1, \ldots, \zeta_n)$ be a root of $f_1, \ldots, f_s$. By Proposition 4.2 the evaluation operator associated to ζ belongs to the vector space E if and only if $\zeta_1 = \alpha_1, \ldots, \zeta_k = \alpha_k$. The conclusion follows easily. □

Finally, remark that now one can use the self validating methods for eigenelements, as exposed in [15] in the method to get an enclosure of the roots.

In the following algorithm, A, a representation of the quotient algebra will be defined by fields which are :

- B are monomial basis of A as $\mathbb{K}$-vector space.
- $normalform$ a function that returns the normal form of any polynomial p in $\langle B \rangle$.

ALGORITHM 3 (`Undernum`).

INPUT : *A a representation of the quotient algebra*
i an integer index
prec a scalar
$lv = (lval, lvec)$ a couple `(list of scalar,vectors)`
OUTPUT : *A list of couple* `(list of enclosure of scalar, vector)`

- $Sol = []$
- *Compute M, the matrix of the restriction of M_i on the vector space spanned by the vectors of $lvec$.*
- *Compute numerically the eigenvectors of M, and compute an enclosure for all of them and their associated eigenvalues.*
- *Apply the clusterization with precision prec.*
- *For each (clusterized) eigenvalue v of M do*
  - *let d be the number of monomials m of $A.B$ such that $m = x_1^{\alpha_1} \ldots x_i^{\alpha_i}$.*
  - *let mult be the algebraic multiplicity of v*
  - *if mult is less than d*
    - *if the vectors associated with v span a vector space of dimension mult*
      - *Let $S = \{i_1, \ldots, i_d\}$ be the list of the indexes of the monomials of $A.B$ corresponding to monomial of the form $x_1^{\alpha_1} \ldots x_i^{\alpha_i}$.*
      - *Let $W = []$.*
      - *for k in S do*
        $W = W :: A.B[k](lval)$
      - *Get the remaining coordinates of W expressing it as a linear combination of the vectors associated to v.*
      - *$Sol = Sol \cup ([], W)$*
  - *else*
    - *if $i == n$ then*
      - *$Sol = Sol \cup (lval \cup v, [])$*
    - *else*
      - *Let le denote the list of vectors associated to v.*
      - *Compute $le' = M_{lv} le$*
      - *Using M_i, le' and v, compute an enclosure for both v and le'.*

· *let l_val be the list of eigenvalues stored in the first member of lv.*

· *Let estim_prec be the estimated precision on the vectors given by : $estim\_prec = \frac{EPSnorm(M)}{SEP(i)}$, where $SEP(i)$ is smallest singular value of the spectral projector associated to v and M (Cf. [2, 18, 8])*

· $Sol = Sol \cup$ `Undernum`$(A, i{+}1, estim\_prec, (l_val :: v, le'))$

- *Return Sol*

ALGORITHM 4 (SYMBONUM).

INPUT : *A a representation of the quotient algebra* OUTPUT *: A certified numerical approximation of the roots of the system $f_1, \ldots, f_s$.*

- $Res = []$
- *Let C be the list of the vectors of the canonical basis.*
- $Tmp\_sol =$ `Undernum`$(A, 1, \eta, ([], C))$
- *Compute the* eigenvectors *of M_1 numerically and apply the clusterization technique on it.*
- *Use the computational method given in [15] to obtain enclosure of each cluster if it fail return* `FAILURE`*.*
- *For each $(lval, lv)$ in Tmp_sol do*
  - *if $lv != []$ then*
    - *Determine which cluster belong the unique vector of lv.*
    - *Compute the linear combinationăof the certified basis vectors that is* equal *to the vector of lv*
    - *Deduce the precision on the lines corresponding to the variables*
  - *else*
    - *Compute de vector W, such that its coordinates satisfies $W[i] = eval(A.B[i], lval)$, i.e. the i-th line of W contains the evaluation of the i-th monomial of $A.B$ at the values contained in $lval$.*
    - *Determine which cluster belong the unique vector of lv.*
    - *Compute the linear combination of the certified basis vectors that is* equal *to the vector of lv*
    - *Deduce the precision on the lines corresponding to the variables*
  - *return Res*

From what precedes, this algorithm stops and its result is either a certified numerical approximation of the roots of the system or `FAILURE`. In that latter case, one has to restart the algorithm with a higher precision and eventually get the desired results.

4.4 An example

Let us see how this algorithm works on an example.

Let us consider the system X^2, Y^2. This example is not radical and is known to make the simple algorithm of [7] to fail regardless of the multiplication operator considered, i.e. no multiplication operator allows to find $(1, 0, 0, 0)$ as an eigenvector.

X^2, Y^2 is a Gröbner basis of (X^2, Y^2) the ideal generated by X^2 and Y^2.

Let us start Algorithm 4, and compute the *eigenvectors* of M_X, we find:

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} \epsilon \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \epsilon \\ 1 \end{pmatrix},$$

where $\epsilon \approx 10^{-406}$.

Hence the dimension of the full invariant subspace associated to the eigenvalue 0 is 4 and the numerical rank of this family is 2, meaning that 0 is defective. So only two vectors are kept here.

There are exactly two monomials in $B = \{1, x, y, xy\}$ that can be expressed using only the variable x, namely 1 and x. The evaluation operator we are looking for is thus such that it has a 1 on its first line and a 0 on its second. Looking at these linear constraints we find out there are not enough information to decide. So a new call to `Undernum` is made.

This last call finds only $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ as eigenvector and finds that 0 is also defective for the restricted operator. y being the last variable, the call to `Undernum` ends and the certification operations start.

Finally a certification up to 10^{-16} in IEEE double precision is found.

5. IMPLEMENTATION

The previous algorithms have been implemented, first in Maple, and then in C++. Maple performances have been disappointing, this is the reason why a C++ implementation has been done.

The implementation is divided into three main components:

- The routine to compute the normal form of the quotient algebra, based on generalized normal forms, see e.g. [12] for the details on how it works.
- The routine for performing the numerical root computing.
- The routine to certify the clusters of the first matrix chosen, see [15] for details.

For dense numerical linear algebra we used a modified version of LAPACK [2], still under active development making it possible to deal with arbitrary coefficients (not only float or double precision coefficients but also GMP [1]). A similar BLAS (Basic Linear Algebra Subprograms), called `blas<T>`, has also been written.

Eigenvectors computation has been performed with the routine `zgeevx<T>` of our library, which is essentially a QR-multishift method with aggressive deflation (see [3]), and

adapted of what is already present in LAPACK for the function `zgeevx`.

The resulting code is roughly 2500 lines of C++ for the main routines, and 100000 of C++ for the numerical linear algebra routines.

We used the Boost interval library (`www.boost.org`) for our implementation of interval arithmetic. Though not being the fastest nor the most complete one, it is generic with respect to the underlying coefficient type which was our main concern.

The actual code used here, or part of it, although not officially released yet is freely available on simple demand. In fact the code is still being actively worked on, and subject to many changes/improvements.

5.1 Raw timings

We first show reported timings (see Table 1) on sample benchmark problems which show the very high efficiency of our method and next detail the computation actually performed. The timings reported here are only concerning the symbolic/numeric method, not including the computation of the representation of the quotient algebra.

The computations have been performed on a Laptop running on Linux 2.6.26, with an Intel Core2 Duo 8400, with 4Go of DDR2-800 RAM. Extended precision computation are performed with mantissa with at least 200 bits.

Table 1 represents the timing of Algorithm 4 for classical matrices arising in a wide range of applications such as robotics, magnetism etc. The dimension of the matrices varies from 10 to 1000. For each matrix, we give the precision with which the computation has been performed (double (64 bits), long double (80 bits) or mpf (>200 bits)) as well as the number of eigenvalue, the measured computed time and the least accuracy of the computed eigenvalues.

Name	arith	Nb. sol.	Time (s)	Prec
cyclic5	double	70	2	1e-15
cyclic5	long double	70	3	1e-16
cyclic5	mpf_class	70	103.84	<1e-50
katsura6	double	64	0.3	1e-10
katsura6	long double	64	0.91	1e-16
katsura6	mpf_class	64	33.91	1e-40
katsura7	double	128	3.8	1e-10
katsura7	long double	128	7.3	1e-16
katsura7	mpf_class	128	515	1e-43
katsura8	double	256	96	1e-4
katsura8	long double	256	127	1e-10
katsura8	mpf_class	256	$> 1h$	1e-10
fabrice24	double	40	0.07	1e-8
fabrice24	long double	40	0.14	1e-11
fabrice24	mpf_class	40	9	1e-41

Table 1: Timing and precision for Algorithm `Symbonum`

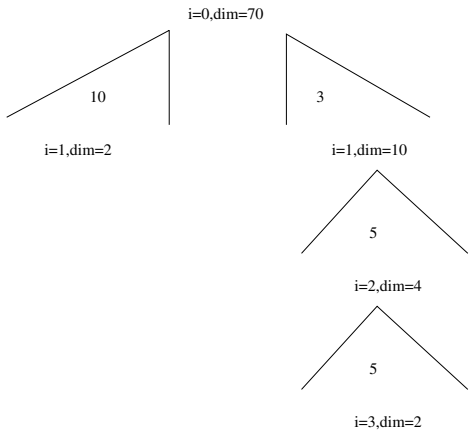
The behaviour at the extended precision computations shows that extended precision computations are rarely needed and that they are much more slower than their native counterpart. The reasons for that the is very simple : `mpf-class` arithmetic is not adapted for these computations. Indeed let us examine what happens when performing a simple operation, lets say $a+=b$.

This operation is translated into a call to the function `__gmp_binary_plus` which involves a call to an allocator, and the actual operation. The call to the allocator is the main drawback since it induces two breaks in the computation flow and spoils the cache reuse. One could imagine a template arithmetic, *statically* parameterized by the desired precision that would avoid these problems. Preliminary tests with *double-double* and *quad-double* arithmetic tend to validate this idea.

Finally in order to get really efficient software, one could imagine running in parallel several increasingly precise versions of the computational routine, the first, even if it is the least precise, that gives a correct answer is enough. Indeed the computational routine given in [15] is just the guarantee that the Newton iteration converges. Hence even if the required precision is not obtained directly, it suffices to apply the Newton procedure to refine it.

5.2 Fine Grain analysis

We present here the details of the computations performed on `cyclic5`. This system is a radical ideal but has the particularity that many different roots share the same coordinate. We report here how the computation behaves. The tree shows the number of recursive calls to `Undernum` and the dimension of the eigenspace considered. Remark that some root may be found since the first step of computation and disappears from the subsecant calls.



What we can see here is that with three level of recursion the numerical computation stops. This improves the algorithm of [11] which would have performed two more calls.

The analysis how the computational time is spent on this example, shows however that most of the time, more or less 66%, is spent in the first eigenvectors computation. Apart this one, the time spent in any other small computation is significantly smaller.

Finally, we noticed that the certification of the computation is most of the time much more time-consuming than the numerical computation itself. However the certification is well adapted for parallel execution.

6. CONCLUSION AND ONGOING WORK

In this article we presented a first version of an efficient symbolic numeric technique for computing an accurate and certified representation of the root of a multivariate polynomial system. Due to a tremendous lack of time, some points have not been investigated so far :

- Instead of choosing the first variable, it is much more interesting, to avoid unneeded recursive calls to consider the multiplication operator by the variable such that the maximum number of monomial are expressed in terms of this variable and the preceding ones (in the computation).
- We are also investigating the idea to include the numerical certification inside the computation routine, and use its indications to propagate exact bounds e.g. for the clusterization.
- No provision is made in the current implementation to use other normal form engine than the one of [12]. It could be interesting to plug the present method in some other solvers.
- Due to bad numerical behavior we were not able to make eigenvalues computation work with stochastic arithmetic [20, 4] which could prove invaluable form clustering this problem is being worked on actively.

However, some preliminary work allows us to expect major improvements in the computational timing, and allows us to provide additional information about the clusters of root computed.

- First of all, recursive calls can clearly be done in parallel, namely, recursive calls correspond to the walk of the call-graph for the routine `Undernum`. Such a description is very well suited for being implemented as OpenMP task. We already have such a prototype implementation and already register substantial performance improvement.
- From a theoretical point of view, one can notice that algebraic multiplicity is not given as an output of the algorithm presented in this paper. The reason for this is that computation are performed only with eigenspaces and not with the full invariant subspaces. These latter objects are precisely encoding the algebraic multiplicity. Remark now that, computing numerically eigenelements of matrices performs the computation of an approximation of a basis of the full invariant subspaces. Hence it remains to store the dimension of this spaces and propagate them through the computation to recover the algebraic multiplicity of the cluster.

To conclude we can summarise our algorithm as two steps :

- A first numerical computation that is currently not certified.
- A second step that is a verification of the numerical computations.

The main improvement of the hereabove algorithm is the use of duality and of the structure of the evaluation operators to avoid some recursive calls.

7. REFERENCES

[1] *GMP, the GNU Multi-Precision library.* Available at URL = `http://www.swox.com/gmp/`.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling an A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide.* SIAM, Philadelphia, 1992. `http://www.netlib.org/lapack/`.

[3] Zhaojun Bai and James Demmel. On a block implementation of hessenberg multishift qr iteration. *Int. J. High Speed Comput.*, 1(1):97–112, 1989.

[4] Jean-Marie Chesneaux, Stéphane Guilain, and Jean Vignes. La bibliothèque CADNA : présentation et utilisation. Manual, Laboratoire d'Informatique de Paris 6, Université P. et M. Curie, Paris, France, November 1996. Available at URL = `http://www-pequan.lip6.fr/cadna/`, (in French).

[5] R.M. Corless, P.M. Gianni, and B.M. Trager. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In W.W. Küchlin, editor, *Proc. ISSAC*, pages 133–140, 1997.

[6] Robert M. Corless. Gröbner bases and matrix eigenproblems. *SIGSAM Bull.*, 30(4):26–32, 1996.

[7] Mohamed Elkadi and Bernard Mourrain. *Introduction à la résolution des systèmes polynomiaux*, volume 59 of *Mathématiques & Applications (Berlin) [Mathematics & Applications]*. Springer, Berlin, 2007.

[8] Nicholas J. Higham. *Accuracy and stability of numerical algorithms.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

[9] Dinesh Manocha and Shankar Krishnan. Solving algebraic systems using matrix computations. *SIGSAM Bull.*, 30(4):4–21, 1996.

[10] H. Michael Möller and Hans J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numer. Math.*, 70(3):311–329, 1995.

[11] H. Michael Möller and Ralf Tenberg. Multivariate polynomial system solving using intersections of eigenspaces. *J. Symb. Comput.*, 32(5):513–531, 2001.

[12] B. Mourrain and Ph. Trébuchet. Generalized normal forms and polynomial system solving. In M. Krauers, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, pages 253–260. New-York, ACM Press., 2000.

[13] Shin'ichi Oishi. Fast enclosure of matrix eigenvalues and singular values via rounding mode controlled computation. *Linear Algebra Appl.*, 324(1-3):133–146, 2001. Special issue on linear algebra in self-validating methods.

[14] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering*, 9:433–461, 1999.

[15] S. Rump. Computational error bounds for multiple or nearly multiple eigenvalues. *Linear Algebra and its Applications*, 324:209–226, 2001.

[16] Siegfried M. Rump and Jens-Peter M. Zemke. On eigenvector bounds. *BIT*, 43(4):823–837, 2003.

[17] Hans J. Stetter. Matrix eigenproblems are at the heart of polynomial systems solving. *SIGSAM Bull.*, 30(4):22–25, 1996.

[18] G. W. Stewart. *Matrix algorithms. Vol. II.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Eigensystems.

[19] Jan Verschelde. Algorithm 795: Phcpack: a general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999.

[20] Jean Vignes. A stochastic arithmetic for reliable scientific computation. *Math. and Comp. in Sim.*, 35:233–261, 1993.