Japan J. Indust. Appl. Math., 26 (2009), 517–530 Area ⟨2⟩

Computer Algebra for Guaranteed Accuracy. How Does It Help?

Masaaki KANNO* and Hirokazu ANAI†

* *Institute of Science and Technology Academic Assembly, Niigata University 8050 Ikarashi 2-no-cho, Nishi-ku, Niigata 950-2181, Japan E-mail: M.Kanno.99@cantab.net*

† *Fujitsu Laboratories Ltd./Kyushu University 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki 211-8588, Japan E-mail: anai@jp.fujitsu.com*

Received August 20, 2008
Revised November 7, 2008

Today simulation technologies (based on numerical computation) are definitely vital in many fields of science and engineering. The accuracy of numerical simulations grows in importance as simulation technologies develop and prevail, and many researches have been carried out for establishing numerically stable algorithms. In recent years, combining computer algebra and other guaranteed accuracy approaches draws much attention as one of promising directions for developing guaranteed accuracy algorithms for a wider class of problems. This paper illustrates several typical usages of symbolic and algebraic methods for guaranteed accuracy computation, highlighting some of the recent applications in control problems.

Key words: algebraic methods, control systems design, guaranteed accuracy computation, validated numerical methods, interval methods

1. Introduction

Since the emergence of modern computers, their computation power has been exploited by scientists and engineers in pursuit of new discoveries and technological advances. Drastic increase of computer power over the course of recent decades allows us to solve a large-scale problem of high complexity which could not be dealt with some time ago. Computers are now indispensable tools in every field of science and engineering. Indeed, *Monozukuri* (innovative design and manufacturing in which Japanese industry has its strength) in recent years would not hold without support from computers; simulation, optimization, and analysis of experimentation results based on high-precision computation in computers are crucial ingredients for successful *Monozukuri* in the light of the recent trend accelerating towards the *model-based development*.

In order to assist computerization in science and engineering, extensive research on the establishment of numerically stable algorithms has been carried out since the infancy of the computer era. One can now appreciate the glorious history of a field in computer science called *numerical analysis*. Numerical properties and computational complexity of numerous fundamental algorithms such as those for numerical linear algebra are now well understood [1, 2]. However, there always remains some subtle issues:

- Does every fundamental mathematical problem admit an efficient and numerically stable solution algorithm?
- Does an algorithm that is composed of a series of well-behaved elementary algorithms always have a desirable property? (Notice that most algorithms for practical problems are derived by combining basic algorithms from toolboxes.)

Indeed, a thirst for more reliable numerical tools for some control problems is reported [3]. These issues have become more critical in recent *Monozukuri* due to toughened requirements, which impose on us solution of ill-conditioned problems from time to time. Thus, the development of numerically reliable algorithms even for such problems has become of great practical interest.

Parallel to the ordinary numerical analysis, the idea of having computers carry out rigorous error analysis emerged [4, 5]. That is, increased computer power is utilized for not only solution of larger problems but also the validation/verification of computed results. Research in this direction is termed "guaranteed accuracy computation" or "validated numerical methods" [6], and numerous achievements have been made [7, 8], ranging from the development of programming languages supporting directed rounding to that of guaranteed accuracy algorithms for solution of a set of linear equations, matrix inversion, computation of polynomial zeros. Most of such developments are based on floating-point arithmetic with directed rounding [9, 10], interval methods [9, 10, 11, 12], and, most importantly, the Krawczyk–Moore method using the Brouwer fixed-point theorem [13, 14]. Now, there are abundant successful applications of guaranteed accuracy computation for scientific and engineering problems [15].

Recent years have also seen the increased interest in combining computer algebra and other guaranteed accuracy approaches for developing guaranteed accuracy algorithms for a wider class of problems, see [16] for example. It is the belief of the authors that symbolic computation/computer algebra can assist in achieving guaranteed accuracy for a wide range of problems of practical significance. In this paper, it is discussed what capacities of computer algebra are useful for realizing guaranteed accuracy algorithms and how guaranteed accuracy can be achieved with the help of computer algebra. This paper, in particular, considers the applicability and practicality of computer algebra for achieving guaranteed accuracy in various contexts for control problems.

The rest of the paper is organized as follows. The relationship between computer algebra and validated numerical methods are explained in Section 2. In Section 3, we mention the applicability of some well-known algebraic methods to stability analysis in control. In Section 4, we briefly discuss the role of computer algebra in a "multi-step" algorithm for achieving guaranteed accuracy computation. Section 5 is devoted to illustrate the capability of computer algebra to transform the original problem into a different form which is tractable for a guaranteed accuracy algorithm. In Section 6, we show how parametric treatment of the problem, which is another advantage of computer algebra, helps us devise guaranteed accuracy algorithms by taking "polynomial spectral factorization" as an example. Section 7 concludes the paper.

2. Computer algebra and validated numerical methods

Computer algebra [17], also called *algebraic computation* or *symbolic manipulation,* is a field of scientific computation which develops, analyses, implements and uses algebraic algorithms. Typical features of algebraic computation are computation with arbitrary precision numbers (i.e., exact and no rounding), computation with symbols and variables, and manipulation of formulae. Algebraic algorithms in computer algebra are usually executed based on exact arithmetic over the field of rationals or an algebraic extension field, which leads to rigour, and moreover allow us *parametric* treatment of problems. There are further advantages of computer algebra in scientific computing, e.g., the capability of obtaining exact global optima for non-convex optimization problems. One may say that computer algebra methods and validated numerical methods share the common aim of solving various mathematical problems rigorously with the aid of computers, although computer algebra approaches attempt to establish effective algorithms based on exact computation, and validated numerical methods move in the opposite direction and refine efficient numerical algorithms for achieving guaranteed accuracy.

As a matter of fact, computer algebra and validated numerical methods can complement each other [18, 19]. Whilst a set of linear equations, for instance, allows the exact solution to be obtained in a computer algebra system, most problems of practical significance do not admit exact/closed-form solutions, and computer algebra does not remove the need for guaranteed accuracy computation algorithms completely. The capability of computer algebra to execute algebraic methods exactly is exploited instead and, powered by such tools, some guaranteed accuracy algorithms were proposed for computing polynomial real/complex roots by isolation and refinement [20, 21]. (Isolation is the process of finding disjoint regions such that each region contains exactly one root (ignoring multiplicity) and every root is contained in some region, while refinement is the process of making isolating regions as small as desired.) In order to accomplish more effective and practical implementation of algebraic algorithms, a research direction towards hybrid methods combining computer algebra with numerical verification methods have been pursued. The idea behind such hybrid methods is to construct a method that still provides guaranteed results in spite of a drastic improvement of computation speed of the corresponding algebraic algorithm by combining the speed of numerical computation with the exactness of symbolic methods. In fact, for non-convex constraint solving/optimization methods, several hybrid algorithms which produce exact global optima have been developed [22, 23, 24]. Typically, validated numeric computation is utilized at crucial parts in hybrid methods; indeed, the hybrid methods utilize floating-point arithmetic and/or interval arithmetic in intermediate computation to avoid computation over algebraic extension field. Viewing such symbolic-numeric methods from another side, one can see that such approaches provide new validated numerical methods for non-convex optimization problems which are difficult tasks for ordinary numerical methods, with the help of symbolic computation.

A further advantage of employing such hybrid methods is discussed in Section 4.

3. Algebraic methods in control

Before the emergence and prevalence of modern computers, algebraic methods had been essential tools in control along with analytic approaches, the reason being that they admit hand calculation. A well-known algebraic method is the Sturm test [25], which allows one to count exactly the number of real roots of a given polynomial in an interval. A tool useful in control is developed based on the Sturm test. This tool is called the Routh–Hurwitz test [26], which examines whether all the roots of a polynomial are located in the open left half plane (LHP), that is, investigates the stability of the polynomial. The approach first constructs a matrix whose elements consist of coefficients of the polynomial under investigation and 0, and then checks the signs of the leading principal minors of the matrix. There is no need to compute/approximate roots of a polynomial, and the approach is amenable to hand calculation.

In recent years, such approaches regained their relevance in the context of modern robust control [27, 28], where real (sometimes complex) parameters are explicitly considered. These tools show conditions with respect to parameters under which the stability of a system is maintained. It should be mentioned here that a parametric version of the Sturm test, called the Sturm–Habicht sequence [29], is also developed, which yields an efficient quantifier elimination (QE) tool for the sign definite condition of a polynomial [30, 31]. It goes without saying that such approaches are compatible with computer algebra.

Here, a numerical example is provided to illustrate the applicability of such an algebraic method. Consider the following polynomial in s:

$$f(s;k) := (s+1)\cdot(s^2+s+1)+k(s+3) = s^3+2s^2+(k+2)s+3k+1,$$

where k is a real parameter. When $k=0$, the roots of $f(s;0)$ are $s=-1,-\frac{1}{2}\pm\mathrm{i}\frac{\sqrt{3}}{2}$, where i is the imaginary unit. Namely, $f(s;0)$ is stable. Now, if k is increased, some of the roots move into the right half plane, i.e., $f(s;k)$ becomes unstable. Thus, the task is to identify at which value of k, f becomes unstable, in other words, f has roots on the imaginary axis. Construct the so-called Hurwitz matrix [26]:

$$H = \begin{bmatrix} 2 & 3k+1 & 0 \\ 1 & k+2 & 0 \\ 0 & 2 & 3k+1 \end{bmatrix}.$$

Its 3 leading principal minors are:

$$[2, 3-k, -3k^2+8k+3\ (= (3k+1)(3-k))]. \tag{1}$$

The necessary and sufficient condition for $f(s;k)$ to be stable is that the signs of the leading principal minors are all positive. (Notice that, when $k=0$, (1) becomes $[2,3,3]$, i.e., all the leading principal minors are positive. This agrees with the observation made earlier for $f(s;0)$.) Analysis of (1) immediately reveals that, when $k=3$, the last two minors become 0, and that, as k further increase, they become negative, and $k=3$ is just on the boundary of the stable and unstable regions. This result and similar analysis for negative k show that $f(s;k)$ is stable for $k \in \left(-\frac{1}{3}, 3\right)$. Indeed, we can see that

$$f\left(s;-\frac{1}{3}\right) = -\frac{1}{3}s(3s^2+6s+5),$$
$$f(s;3) = (s+2)(s^2+5),$$

and we can confirm that, when $k=-\frac{1}{3}$ or $k=3$, $f(s;k)$ has roots precisely on the imaginary axis and that, as k decreases from $-\frac{1}{3}$ or increases from 3, the roots on the imaginary axis move into the right half plane.

4. Symbolic manipulation towards hybrid approaches

One of difficulties in achieving good numerical accuracy or guaranteed accuracy computation for a practical problem lies in the fact that such a problem is more often than not solved by a "multi-step" algorithm, i.e., an algorithm consisting of a number of sub-algorithms which themselves solve concrete problems. For instance, the solution to a problem is obtained by solving a particular equation using an algorithm and then computing by another algorithm the solution of another problem whose input data rely on the first solution. Indeed, it is common that control synthesis problems require such an approach. It can happen that a normalization, say, achieves better numerical accuracy in the first part of the entire solution process, but that the normalization is not suited for the remaining part, which thus yields poor numerical accuracy as a whole. Also, if the first part is solved by means of a guaranteed accuracy algorithm, then input data to the subsequent part will usually be intervals (guaranteed to contain the true values), which is not a typical scenario in guaranteed accuracy computation. A conceivable way to proceed may be to work with such intervals by employing interval arithmetic, but a pitfall in such an approach is that the resulting answer tends to suffer from the wrapping effect [5, 32, 33] and will be given as intervals whose widths are too large to be meaningful in practice.

This section discusses that computer algebra can connect sub-steps in a multi-step algorithm and help achieve guaranteed accuracy computation. The advantage is accomplished by the capability of handling symbols. By expressing the solutions of the sub-steps as symbols, it is possible to relate the input data (to the original problem) and the final result for a number of practical problems. That is, introduction of intermediate variables allows the final result to be directly related to the input data. By exploiting the obtained relationship and by way of the Krawczyk–Moore method [14], intervals of large width can be shrunk to desired width.

The approach has another advantage in that it allows an efficient "hybrid" guaranteed accuracy algorithm [4, 19] to be devised. Namely, an ordinary numerical method is combined with a computer algebra-based approach to produce a numerical method that has guarantee on the correctness of the result. Since a set of equations the solution has to satisfy is explicitly obtained, one can use it to examine the fidelity of the solution obtained from an ordinary numerical method. In case the numerical method fails to deliver a solution that is sufficiently close to the true solution for yielding a guaranteed accuracy solution, then the guaranteed accuracy algorithm is to be invoked from the beginning. By doing so, guaranteed accuracy computation is mainly used to check and improve the result from an efficient ordinary numerical method and the average computation time is reduced, while accuracy of the computed result is always guaranteed.

Such an approach may be employed for computing all the roots of a given polynomial. First notice that all the roots and the coefficients of the polynomial are related in an algebraic manner. The approach begins with approximating the roots of the polynomial by means of an ordinary numerical polynomial root finder, and error bounds are to be estimated in some way; they are used to form initial intervals. With those intervals, the Krawczyk–Moore condition [14] is firstly examined, and, if the condition is satisfied, it is guaranteed that all the roots are contained in the initial intervals. In order to get tighter guaranteed bounds, the Krawczyk–Moore method [14] is repeatedly applied until the required accuracy is achieved. In case the numerical method did not yield a sufficiently accurate result, an infallible guaranteed accuracy method such as [21] is then employed. In this way, one can always compute polynomial roots with guaranteed accuracy, and the computation time required is kept low on average.

5. Problem modification

Another strength of computer algebra is its capability to convert the original problem into a different form which is more amenable to a guaranteed accuracy algorithm. The point is that the conversion is exact, and, if one can find a solution with guaranteed accuracy for the problem in converted form, then the solution also serves as a guaranteed accuracy solution for the original problem one wants to solve. A typical example is a problem of computing the solution to a system of polynomial equations with guaranteed accuracy. By Gröbner basis computation with respect to the pure lexicographical order, we can convert a system of polynomial equations that has only finitely many complex solutions into a triangular form. We can then solve the obtained system with ease by recursively finding roots of univariate polynomials. For example, consider the following set of multivariate polynomials

$$\mathscr{I} := \{x - y - z, x + y - z^2, x^2 + y^2 - 1\},$$

which indeed has a finite number of zeros. Then the Gröbner basis for the ideal $\langle \mathscr{I} \rangle$ generated by $\mathscr{I}$ with respect to the pure lexicographical order $x \succ y \succ z$ is given as follows:

$$\mathscr{J} := \{z^4 + z^2 - 2, 2y - z^2 + z, 2x - z^2 - z\}.$$

Note that $\mathscr{J}$ has the same set of common zeros as the original set of polynomials, $\mathscr{I}$, but that it is in upper triangular form. This obviously implies that one can find zeros of $\mathscr{J}$ more easily than those of the original $\mathscr{I}$. For further details of Gröbner bases and associated ideas such as the ideal and the pure lexicographic order, readers are referred to standard textbooks such as [34, 35].

Also effective is an algebraic approach called quantifier elimination (QE), where a given set of first order formulae with quantifiers (for instance, $\exists$, $\forall$) is reduced to another *equivalent* set of formulae without quantified variables. The resulting formulae are in general polynomial equations/inequalities. Therefore, QE methods may help obtain a problem that can be easily tackled. As is stated in Section 2, guaranteed accuracy algorithms for real/complex root computation are established. It is therefore desirable if one can convert the original problem into a polynomial root finding problem. The computation of the $\mathscr{L}_\infty$-norm of a linear dynamical system is an example that can be dealt with in such a way [36], which is briefly reviewed in the sequel.

Write as $\mathscr{RL}_\infty$ the Lebesgue space of matrix valued functions G, whose elements are rational functions with real coefficients, that are bounded on the imaginary axis, with norm defined by

$$\|G\|_\infty := \sup_{\omega \in \mathbb{R}} \overline{\sigma}\{G(\mathrm{i}\omega)\},$$

where $\overline{\sigma}\{\,\cdot\,\}$ denotes the largest singular value of a matrix. A typical numerical approach for computing the $\mathscr{L}_\infty$-norm $\|G\|_\infty$ of a system $G(s) \in \mathscr{RL}_\infty$ finds upper/lower bounds for $\|G\|_\infty$ by examining the existence/non-existence of imaginary axis eigenvalues in the associated Hamiltonian matrix [37]. In the context of floating-point arithmetic, this computation may be prone to numerical difficulties since it essentially tries to compute nearly multiple eigenvalues near the imaginary axis. The following result gives a guaranteed accuracy algorithm for computing the $\mathscr{L}_\infty$-norm of a dynamical system.

THEOREM 1 ([36]). *Suppose that $G(s) \in \mathscr{RL}_\infty$, and let $\Phi_\gamma(s) := \gamma^2 I - G^{\mathrm{T}}(-s)G(s)$. Substitute x for s^2 in $\det \Phi_\gamma(s)$ and write as $g_\gamma(x)$, i.e., $g_\gamma(s^2) = \det \Phi_\gamma(s)$. Furthermore, write $g_\gamma(x) = \frac{n_\gamma(x)}{d_\gamma(x)}$ where $n_\gamma(x)$ and $d_\gamma(x)$ are polynomials in x and γ which are coprime over $\mathbb{R}[x,\gamma]$. Compute $h_\gamma^s(x)$ as*

$$h_\gamma^s(x) = \frac{n_\gamma(x)}{\mathrm{GCD}\big(n_\gamma(x), \frac{\partial}{\partial x} n_\gamma(x)\big)}.$$

Then, $\|G\|_\infty$ is one of the following quantities:
(i) $\bar{\sigma}\{G(0)\}$,
(ii) $\bar{\sigma}\{G(\mathrm{i}\infty)\}$,
(iii) *a real root of the discriminant of $h^s_\gamma(x)$ with respect to x (which will be a polynomial in γ (γ^2, in fact)).*

Moreover, each of the above quantities is a (real) root of a real univariate polynomial, and the true $\|G\|_\infty$ can be determined by means of the Sturm test on $h^s_\gamma(x)$.

Thus, the $\mathscr{L}_\infty$-norm computation is reduced to the computation of polynomial roots. It is emphasized that guaranteed accuracy computation for polynomial roots is equivalent to guaranteed accuracy computation of the $\mathscr{L}_\infty$-norm since the conversion is exact. Namely, guaranteed accuracy computation for $\|G\|_\infty$ can be achieved by the power of computer algebra to modify a problem into a more desirable form, i.e., eliminating s in $\Phi_\gamma(s)$, or x in $h^s_\gamma(x)$ in this case. It is noted that, in this problem, the structural property of the problem is fully exploited to give an effective QE approach. For general QE problems, various computational approaches have been developed, e.g., [38]. Typically, the more general problem an algorithm aims at, the more expensive its computation cost is. There is a trade-off between the applicability and the practicality of the algorithm. Extensive efforts have been made to achieve more efficient general algorithms, but making as much use of problem properties as possible is equally important. The Sturm–Habicht sequence mentioned in Section 2 is in fact one such example and is suited to perform QE for deriving conditions of the positivity/negativity of a polynomial with parameters in its coefficients.

6. Parametric problem

Problems with parameters are abundant in practical engineering applications, and systematic approaches to such problems are highly desired. Approaches exploiting interval methods have been developed for various control problems, where ranges of parameters are regarded as intervals and computation is executed for those intervals [33]. While such approaches can compute, for instance, guaranteed upper/lower bounds for the solution, it will be beneficial if one could deal with parameters in a direct fashion.

It goes without saying that one of strengths of computer algebra is the flexible symbolic computation capability. This allows one to handle parameters as they are and to get a solution in the presence of parameters. Analysis of the solution in terms of parameters would give much deeper insight into the behaviour of the solution against parameter variations than analysing a set of numerical solutions for several specific values of parameters. Another advantage is that parametric optimization can be performed, where the optimal cost is expressed explicitly in terms of parameters (rather than as the result of numerical optimization). This is in particular effective in such a scenario in control as the following: one wants to design both the plant and the controller simultaneously to achieve the optimal overall performance, but the optimal controller is in fact a function of the plant in an abstract sense. In such a situation, the problem can be cast as, for example,

$$\min_{\text{Plant}} \min_{\text{Controller}} J,$$

where J is a given cost function. If the first optimization ("$\min_{\text{Controller}}$" part) admits parametric optimization, then the second optimization ("$\min_{\text{Plant}}$" part) will be amenable to various sorts of optimization approaches, resulting in a high chance of achieving the global optimum. If this is not the case, one would then have to attempt some heuristic optimization approach.

When the first part requires solution of a set of linear equations, then what is needed is computer algebra that gives exact solutions containing parameters, and approaches developed to visualize the parametric solution set exactly, e.g., [39], are applicable. A more important problem relevant to application areas is a non-linear one. Given a set of algebraic equations with parameters, it is in general difficult to find the solution set. The Gröbner basis mentioned in Section 5 can be useful to modify the problem into a simpler form suited to further computation even in the presence of parameters. Indeed, clever exploitation of the problem structure can allow one to investigate a parametric non-linear problem efficiently.

Here, a solution approach to a particular non-linear problem is reviewed that can be employed for the parametric case. The problem under study is called polynomial spectral factorization, an important mathematical tool in signal processing and control. Consider the following even polynomial of degree $2n$ in s:

$$f(s) = a_{2n}s^{2n} + a_{2n-2}s^{2n-2} + \cdots + a_2s^2 + a_0. \tag{2}$$

In the following, it is assumed that $a_{2k} \in \mathbb{R}$ for $k = 0, 1, \ldots, n$ for the brevity of the exposition, but the approach can be generalized to the parametric case where a_{2k} is some polynomial in parameters [40]. Assume without loss of generality that $a_{2n} > 0$. It can be deduced that the roots of $f(s)$ are located symmetrically with respect to the imaginary axis. It is supposed that $f(s)$ has no roots on the imaginary axis. Polynomial spectral factorization is a decomposition of $f(s)$ into two real polynomials, one that captures all the left half plane roots and its "mirror image":

$$a_{2n}f(s) = (-1)^n g(s)g(-s), \tag{3}$$

where

$$g(s) = b_ns^n + b_{n-1}s^{n-1} + \cdots + b_1s + b_0 \in \mathbb{R}[s], \quad b_n = a_{2n}, \tag{4}$$

has roots in the open *left* half plane only. The polynomial $g(s)$ is called the *spectral factor* of $f(s)$. There always exists such $g(s)$, and it is unique and in $\mathbb{R}[s]$.

There are various *numerical* algorithms devised for computing $g(s)$ for a given $f(s)$ [41]. Here, an *algebraic* algorithm is presented. Note that an *algebraic* method can deal with the parametric case while a *numerical* method can do little in such a case. The following fact is the basis of the efficient approach developed.

THEOREM 2 ([40, 42]). *Given $f(s)$ and $g(s)$ as in (2) and (4), respectively, consider b_i, $i = 0, \ldots, n-1$, as variables. A system of algebraic equations in terms of b_i's is obtained by comparing the coefficients of (3). Then, the set $\mathscr{G}$ of the polynomials obtained from the polynomial parts of the equations, with the coefficient of b_i^2 set to 1, forms the reduced Gröbner basis of the ideal $\langle \mathscr{G} \rangle$ generated by itself in $\mathbb{R}[\mathbf{B}]$ with respect to the graded reverse lexicographic order $b_{n-1} \succ \cdots \succ b_0$, where $\mathbf{B}$ denotes $\{b_0, \ldots, b_{n-1}\}$.*

The ideal $\langle \mathscr{G} \rangle$ of $\mathbb{R}[\mathbf{B}]$ is called the *ideal of spectral factorization.* It is straightforward to show that the ideal of spectral factorization is 0-dimensional and that the number of its zeros with multiplicities counted is 2^n. Furthermore, it can be shown [40] that b_{n-1} is generically a separating element [43]. These facts indicate that $\langle \mathscr{G} \rangle$ has a special Gröbner basis.

THEOREM 3 ([42]). *The ideal of spectral factorization generically has a Gröbner basis of so-called shape basis with respect to any elimination ordering $\{b_0, \ldots, b_{n-2}\} \succ\succ b_{n-1}$:*

$$\mathscr{F} := \{\hat{S}_f(b_{n-1}), b_{n-2} - \hat{h}_{n-2}(b_{n-1}), \ldots, b_0 - \hat{h}_0(b_{n-1})\},$$

where $\hat{S}_f$ is a polynomial of degree exactly 2^n and $\hat{h}_i$'s are polynomials of degree strictly less than 2^n.

The implication of this result is that polynomial spectral factorization is in essence reduced to the computation of a root β of a univariate polynomial $\hat{S}_f(b_{n-1})$; the rest of the computation is mere evaluation of polynomials $\hat{h}_i$'s at $b_{n-1} = \beta$. Also important is that, of 2^n roots of $\hat{S}_f(b_{n-1})$, the largest real root $\tilde{\beta}$ yields the spectral factor $g(s)$ [40, 42] and that no extra effort is required to choose from a set of possible solutions. In regard to computation efficacy, what is needed is basis conversion from one Gröbner basis to another Gröbner basis. Efficient algorithms, e.g., [43, 44], are available and one does not have to resort to a generally expensive Gröbner basis computation method based on, e.g., Buchberger's algorithm. Lastly, this approach can be extended to the parametric case [40] due to the fact that all the manipulation required is algebraic operations.

An example is employed to illustrate the result outlined above. The following even polynomial in s is decomposed:

$$f(s) := s^6 - \alpha_2^2 s^4 + (1 - 6\alpha_2)s^2 - \alpha_1^2 - 9,$$

where α_1, α_2 are real parameters. The degree of $f(s)$ is 6, and thus the spectral factor is of degree 3 and is written as

$$g(s) := s^3 + b_2 s^2 + b_1 s + b_0.$$

Comparing the coefficients of (3), the following set of algebraic equations is obtained:

$$\begin{cases} b_2^2 - 2b_1 - \alpha_2^2 = 0, \\ b_1^2 - 2b_2b_0 + 6\alpha_2 - 1 = 0, \\ b_0^2 - \alpha_1^2 - 9 = 0. \end{cases} \tag{5}$$

It is noted that the polynomial parts (i.e., the left hand sides) of the above equations form a Gröbner basis of the ideals generated by themselves with respect to the graded reverse lexicographic order $b_2 \succ b_1 \succ b_0$.

Converting them into the Gröbner basis with respect to, say, the pure lexicographic order $b_0 \succ b_1 \succ b_2$, the shape basis is obtained. The set of polynomials is equivalent to

$$\begin{cases} b_2^8 - 4\alpha_2^2 b_2^6 + (6\alpha_2^4 + 48\alpha_2 - 8)b_2^4 \\ \quad + (-4\alpha_2^6 - 96\alpha_2^3 - 64\alpha_1^2 + 16\alpha_2^2 - 576)b_2^2 \\ \quad + \alpha_2^8 + 48\alpha_2^5 - 8\alpha_2^4 + 576\alpha_2^2 - 192\alpha_2 + 16 = 0, \\ 2b_1 - b_2^2 + \alpha_2^2 = 0, \\ 8(\alpha_2^4 + 24\alpha_2 - 4)b_0 + b_2^7 - 4\alpha_2^2 b_2^5 + (5\alpha_2^4 + 24\alpha_2 - 4)b_2^3 \\ \quad + (-2\alpha_2^6 - 48\alpha_2^3 - 64\alpha_1^2 + 8\alpha_2^2 - 576)b_2 = 0. \end{cases} \tag{6}$$

Notice that the left hand side of the first equation of (6) is a polynomial in b_2 only (i.e., without b_1, b_0) with parameters α_1, α_2 in its coefficients. The left hand sides of the remaining 2 equations are linear in b_1 and b_0, respectively. Namely, the first equation relates b_2 and the parameters whereas the second (resp., third) equation gives a description of b_1 (resp., b_0) in terms of b_2 and the parameters. Since the first equation contains b_2 only and other b_i's are related in a linear manner, it is much easier to analyse the relationship between b_i's and the parameters compared to carrying out analysis based on, e.g., (5).

Remark. In fact, in the above example, when $\alpha_2^4 + 24\alpha_2 - 4 = 0$, a shape basis of different form is obtained; see the third polynomial equation in (6). In that case, it can be confirmed that the first equation in (6) has a multiple root, and that, after removing it, we can get a suitable shape basis. Namely, the "singular" case can be dealt with; see [40] for further detail on how to treat such a case.

This method is called the *Sum of Roots* (SoR) approach since the focus is on b_{n-1}, and $-b_{n-1}/b_n$ is the "sum of roots" of spectral factor $g(s)$. The virtue of this approach is that not only the approach is computationally effective but also the SoR has some essential meaning in control theory [45, 46]. Polynomial spectral factorization reviewed in this section is the *continuous-time* one, to be precise. It is shown [47] that a similar algebraic approach can be devised for *discrete-time* polynomial spectral factorization, where the polynomial to be decomposed has roots symmetric about the unit circle and the spectral factor is to contain all the roots inside the unit circle.

Cost functions and optimal controllers for many control problems can be explicitly expressed in terms of the coefficients of the polynomial spectral factor [40, 45, 46]. Therefore, once parametric polynomial spectral factorization is done, quantities to be pursued are expressed in terms of the SoR and parameters, typically as rational functions. Given the ranges of values that the parameters can take, one would want to perform optimization of a cost function over parameters. It is shown [40] that such an optimization problem can be cast as a QE problem and the (true) exact global optimum may be obtained (as a root of a polynomial). In this way, computer algebra can help compute with guaranteed accuracy the global optimum for intrinsically difficult problems.

The point in this approach presented above is that the cost function that is usually obtained through a multi-step algorithm is related with parameters in the initial input data by means of effective use of an algebraic method (namely, the Gröbner basis technique) and is further optimized systematically (using the QE approach), resulting in a guaranteed accuracy solution. Typically, such a problem is tackled with a rather brute force method, computing the optimal cost for a particular set of parameter values and heuristically finding a better solution repeatedly. Algebraic methods help establish a systematic treatment for guaranteed accuracy computation.

7. Concluding remarks

This paper has discussed how computer algebra can be utilized for achieving guaranteed accuracy in the solution of some control problems. Hybrid approaches seem promising in that the power of symbolic manipulation can widen the range of problems we can tackle and moreover the efficiency of ordinary validated numerical methods may significantly improve the computation time when compared to pure algebraic approaches. The authors believe that further research on the development of effective hybrid methods will lead to the establishment of general schemes for designing hybrid algorithms for practical problems in science and engineering. It is hoped that the paper stimulates readers into this important direction and that new exciting ideas will emerge to resolve numerical issues arising in solving realistic problems of practical significance.

Acknowledgements. The authors are grateful to Prof. Shinji Hara of the University of Tokyo and Prof. Kazuhiro Yokoyama of Rikkyo University for useful and fruitful discussions. The authors would also like to thank the anonymous referees for helpful suggestions which have resulted in the improvement of this paper.

References

[1] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd edition. The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[2] N.J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd edition. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[3] N.J. Higham, M. Konstantinov, V. Mehrmann and P. Petkov, The sensitivity of computational control problems. IEEE Control Systems Magazine, **24** (2004), 28–43.

[4] W. Krandick and S. Rump, Foreword of the guest editors. Special issue on validated numerical methods and computer algebra, Journal of Symbolic Computation, **24** (1997), 625–626.

[5] S.M. Rump, Self-validating methods. Linear Algebra and Its Applications, **324** (2001), 3–13.

[6] W. Krandick and S. Rump (eds.), Special issue on validated numerical methods and computer algebra. Journal of Symbolic Computation, Vol. 24, No. 6, Academic Press, 1997.

[7] U.W. Kulisch and W.L. Miranker (eds.), A New Approach to Scientific Computation. Notes and Reports in Computer Science and Applied Mathematics, Vol. 7, Academic Press, New York, 1983.

[8] J. Rohn, S.M. Rump and T. Yamamoto (eds.), Special issue on linear algebra in self-validating methods. Linear Algebra and Its Applications, Vol. 324, No. 1–3, Elsevier, 2001.

[9] R.E. Moore, Interval Analysis. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[10] A. Neumaier, Interval Methods for Systems of Equations. Cambridge University Press, Cambridge, 1990.

[11] T. Sunaga, Theory of interval algebra and its application to numerical analysis. RAAG Memoirs of the Unifying Study of Basic Problems in Engineering and Physical Sciences by Means of Geometry, Vol. 2, K. Kondo (ed.), Gakujutsu Bunken Fukyu-kai, Tokyo, Japan, 1958, 29–46 (547–564).

[12] G. Alefeld and J. Herzberger, Introduction to Interval Computations. Academic Press, New York, NY, 1983.

[13] R. Krawczyk, Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. Computing, **4** (1969), 187–201.

[14] R.E. Moore, A test for existence of solutions to nonlinear systems. SIAM Journal on Numerical Analysis, **14** (1977), 611–615.

[15] R.L. Muhanna and R.L. Mullen (eds.), Special issue on reliable engineering computing. Reliable Computing, Vol. 12, No. 6–Vol. 13, No. 2, Springer, Netherlands, 2006–2007.

[16] B. Buchberger, C. Jansson, S. Oishi, M. Plum and S.M. Rump, 05391 executive summary—Numerical and algebraic algorithms and computer-assisted proofs. Algebraic and Numerical Algorithms and Computer-Assisted Proofs, B. Buchberger, S. Oishi, M. Plum and S.M. Rump (eds.), Dagstuhl Seminar Proceedings, No. 05391, Dagstuhl, Germany, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[17] J. von zur Gathen and J. Gerhard, Modern Computer Algebra, 2nd edition. Cambridge University Press, Cambridge, 2003.

[18] S.M. Rump, Algebraic computation, numerical computation and verified inclusions. Trends in Computer Algebra, R. Janssen (ed.), Lecture Notes in Computer Science, Vol. 296, Springer-Verlag, New York, NY, 1988, 177–197.

[19] S.M. Rump, Computer-assisted proofs and self-validating methods. Accuracy and Reliability in Scientific Computing, B. Einarsson (ed.), Software, Environments, Tools, Vol. 18, Chapter 10, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005, 195–240.

[20] A.G. Akritas, Elements of Computer Algebra with Applications. John Wiley & Sons, New York, NY, 1989.

[21] G.E. Collins and W. Krandick, An efficient algorithm for infallible polynomial complex root isolation. Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC '92, P.S. Wang (ed.), ACM Press, New York, NY, 1992, 189–194.

[22] H. Hong, An efficient method for analyzing the topology of plane real algebraic curves. Mathematics and Computers in Simulation, **42** (1996), 571–582.

[23] S. Ratschan, Approximate quantified constraint solving by cylindrical box decomposition. Reliable Computing, **8** (2002), 21–42.

[24] A.W. Strzebonski, Cylindrical algebraic decomposition using validated numerics. Journal of Symbolic Computation, **41** (2006), 1021–1038.

[25] F.R. Gantmacher, The Theory of Matrices, Vol. 2. Chelsea Publishing Company, New York, NY, 1960.
[26] K. Ogata, Modern Control Engineering, 4th edition. Prentice Hall, Upper Saddle River, NJ, 2002.
[27] J. Ackermann, Robust Control: Systems with Uncertain Physical Parameters. Springer-Verlag, London, 1993.
[28] S.P. Bhattacharyya, H. Chapellat and L.H. Keel, Robust Control: The Parametric Approach. Prentice-Hall, Upper Saddle River, NJ, 1995.
[29] L. González-Vega, T. Recio, H. Lombardi and M.-F. Roy, Sturm–Habicht sequences determinants and real roots of univariate polynomials. Quantifier Elimination and Cylindrical Algebraic Decomposition, B.F. Caviness and J.R. Johnson (eds.), Texts and Monographs in Symbolic Computation, Springer, Wien, New York, 1998, 300–316.
[30] H. Anai, H. Yanami, S. Hara and K. Sakabe, Fixed-structure robust controller synthesis based on symbolic-numeric computation: design algorithms with a CACSD toolbox. Proceedings of the 2004 IEEE International Conference on Control Applications, Taipei, Taiwan, 2004, 1540–1545.
[31] H. Anai and S. Hara, A parameter space approach to fixed-order robust controller synthesis by quantifier elimination. International Journal of Control, **79** (2006), 1321–1330.
[32] R.E. Moore, Methods and Applications of Interval Analysis. SIAM Studies in Applied Mathematics, Vol. 2, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.
[33] L. Jaulin, M. Kieffer, O. Didrit and É. Walter, Applied Interval Analysis. Springer-Verlag, London, 2001.
[34] D. Cox, J. Little and D. O'Shea, Ideals, Varieties, and Algorithms, 3rd edition. Springer, New York, NY, 2007.
[35] T. Becker and V. Weispfenning, Gröbner Bases: A Computational Approach to Commutative Algebra. Graduate Texts in Mathematics, Vol. 141, Springer-Verlag, New York, NY, 1993.
[36] M. Kanno and M.C. Smith, Validated numerical computation of the $\mathscr{L}_\infty$-norm for linear dynamical systems. Journal of Symbolic Computation, **41** (2006), 697–707.
[37] K. Zhou, J.C. Doyle and K. Glover, Robust and Optimal Control. Prentice-Hall, Upper Saddle River, NJ, 1996.
[38] G. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition. Proceedings Second GI Conference on Automata Theory and Formal Languages, Lecture Notes in Computer Science, Vol. 33, Springer-Verlag, Berlin, 1975, 134–183.
[39] E.D. Popova and W. Krämer, Visualizing parametric solution sets. BIT Numerical Mathematics, **48** (2008), 95–115.
[40] H. Anai, S. Hara, M. Kanno and K. Yokoyama, Parametric polynomial spectral factorization using the sum of roots and its application to a control design problem. Journal of Symbolic Computation, **44** (2009), 703–725.
[41] A.H. Sayed and T. Kailath, A survey of spectral factorization methods. Numerical Linear Algebra with Applications, **8** (2001), 467–496.
[42] M. Kanno, H. Anai and K. Yokoyama, On the relationship between the sum of roots with positive real parts and polynomial spectral factorization. Numerical Methods and Applications—6th International Conference, NMA 2006, T. Boyanov et al. (eds.), Borovets, Bulgaria, August, 2006; Revised Papers. Lecture Notes in Computer Science, Vol. 4310, Springer-Verlag, Heidelberg, 2007, 320–328.
[43] M. Noro and K. Yokoyama, A modular method to compute the rational univariate representation of zero-dimensional ideals. Journal of Symbolic Computation, **28** (1999), 243–264.
[44] J.C. Faugère, P. Gianni, D. Lazard and T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering. Journal of Symbolic Computation, **16** (1993), 329–344.
[45] M. Kanno, S. Hara, H. Anai and K. Yokoyama, Sum of roots, polynomial spectral factorization, and control performance limitations. Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, Louisiana, USA, 2007, 2968–2973.
[46] S. Hara and M. Kanno, Sum of roots characterization for $\mathscr{H}_2$ control performance limitations. SICE Journal of Control, Measurement, and System Integration, **1** (2008), 58–65.
[47] M. Kanno, K. Yokoyama, H. Anai and S. Hara, Symbolic optimization of algebraic functions. Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2008, Linz, Austria, 2008, 147–154.