

The Method of Resultants for Computing Real Solutions of Polynomial Systems

Eugene L. Allgower^{1,2}, Kurt Georg^{1,2} and Rick Miranda^{1,2}

July 1990, revised May 1991

Abstract: A new method for determining the real solutions to a set of polynomial equations is presented. It is based on the theory of multi-resultants. The inherently unstable calculation of the determinant is replaced by a stable minimization procedure which is able to take advantage of the sparseness of the resultant matrix. Two numerical examples illustrate the method. The paper contains preliminary work which demonstrates the feasibility of the given approach.

Keywords: roots, polynomial systems of equations, resultant, conjugate gradient method, Lanczos method

AMSMOS: 65H10, 65H20, 26C10, 65F15, 15-04, 15A15

1 Introduction

In recent years, a number of authors have considered the task of numerically determining all of the zero points of polynomial systems of equations. In particular, we mention the resultant method of Collins [5] and the homotopy methods, see, e.g., [3, 4, 10, 17, 21, 22, 28]. Further references can be found in the recent books [1, 20]. Since the calculation of the determinant of the resultant is an unstable problem, Collins' method has heretofore been confined to systems involving integer coefficients, and the use of exact integer arithmetic plays a crucial role.

In the homotopy approach one calculates all of the complex zero points by numerical continuation. In view of Bezout's theorem the number of zero points is generically equal to the product of the degrees of the component maps (with accounting for multiplicities and solutions at infinity). The homotopy method is generally both stable and exhaustive, i.e., usually finds all complex roots.

In this paper we address the problem of finding only the real zeros in a prescribed n -dimensional rectangle for polynomial systems having real coefficients. Most of the applications arising in science concerning polynomial systems are of this nature. Our approach uses aspects of both the resultant method and the

¹Department of Mathematics, Colorado State University, Ft. Collins, Colorado 80523

²This work was supported by the National Science Foundation under grant numbers DMS-8947761, DMS-8501007 and by the Deutsche Forschungsgemeinschaft

method of numerical continuation. For a given polynomial system

$$P(y) = \begin{pmatrix} P_0(y_1, \dots, y_n) \\ \vdots \\ P_{n-1}(y_1, \dots, y_n) \end{pmatrix} = 0, \quad (1)$$

we consider for each of the variables y_i a multi-resultant $R_i(y_i) = \det M_i(y_i)$, where M_i generally is a large sparse matrix. It can be shown, see [27], that $R_i(y_i) = 0$ is a necessary condition on the i -th component of any zero point y of P . If Z_i denotes the set of zero points of $R_i(y_i) = 0$, then any zero point of P must belong to the Cartesian product

$$\prod_{i=1}^n Z_i. \quad (2)$$

The points of this set can easily be tested and iteratively refined by standard numerical methods to sift out the actual zero points of P .

The two major tasks which must be dealt with are the construction of the multi-resultant matrix $M_i(y_i)$, and the instability of the equation $R_i(y_i) = 0$, since typically $R_i(y_i)$ is a polynomial of very high degree in the unknown y_i .

We handle the latter problem by replacing the condition $R_i(y_i) = 0$ with the equivalent condition

$$\min_{\|u\|=1} \|M_i(y_i)u\|^2 = 0. \quad (3)$$

Here u denotes a column vector of the same length as the size of the matrix $M_i(y_i)$, and $\|\cdot\|$ always denotes the Euclidean norm. Problem (3) is well known to be stable. It amounts to calculating the smallest eigenvalue of the semi-definite matrix $M_i(y_i)^*M_i(y_i)$ and testing whether it is zero. Here and in the following we denote transposition by $*$. Although the eigenvalue calculation could be done by a Lanczos method, see, e.g., [12], we find it convenient to implement a conjugate gradient method constrained to the unit sphere. Since we are only interested in finding out if zero is an eigenvalue of $M_i(y_i)$, another promising avenue would be to apply a Lanczos method for large sparse nonsymmetric matrices such as those discussed in, e.g., [6, 7, 24].

In view of the form of the functional

$$\varphi_i(u) := \frac{1}{2} \|M_i(y_i)u\|^2, \quad (4)$$

it is only necessary to implement a method of calculating products of the form

$$v = M_i(y_i)u \quad \text{and} \quad w = M_i(y_i)^*v. \quad (5)$$

Typically, either the Lanczos method or the conjugate gradient method needs one of each of these evaluations per step. These evaluations will be described in detail since they involve a certain degree of combinatorial complexity. Although the dimensions m of resultant matrices $M_i(y_i)$ and $M_i(y_i)^*$ can be very large, the evaluation of the products (5) is in typical applications equivalent to only

a few scalar products of dimension m . Some numerical examples illustrate the performance of our approach.

Let us conclude our introduction by stressing that we propose the present approach as a possible alternative to the homotopy method when only the real roots in a region are desired. If the number of real solutions of a system is indeed close to the Bezout number mentioned above, then the homotopy method is probably to be preferred. In fact, our method is primarily meant for systems which have only relatively few real solutions. The numerical results given at the conclusion only represent a first attempt to apply the multi-resultant method. In particular, many numerical questions concerning degeneracy and efficiency remain open. On the other hand, the results of this first attempt give us grounds for optimism.

Other possible alternatives would be interval methods or generalized bisection methods. Of related interest are the recent book [23], the papers [13, 14, 16] and the references therein. In these general methods, the specific properties of polynomial systems do not seem to our knowledge to have been exploited yet.

Another method of related interest [2] is based upon the Gröbner bases approach for symbolically handling polynomial systems.

2 The Multi-Resultant

Let

$$P(x) = \begin{pmatrix} P_0(x_0, \dots, x_{n-1}) \\ \vdots \\ P_{n-1}(x_0, \dots, x_{n-1}) \end{pmatrix}$$

be n homogeneous polynomials with real coefficients in n variables. If the coefficients are chosen generically, there will be no common solutions to the system

$$P(x) = 0 \tag{6}$$

in the complex projective space \mathbf{CP}^{n-1} having homogeneous coordinates x . The condition that the system (6) has a solution in \mathbf{CP}^{n-1} is a condition on the coefficients of the P_i 's. If we think of the coefficients as undetermined variables, then there is in fact a polynomial R in these coefficients of the P_i 's, which vanishes if and only if the system (6) has a complex solution in \mathbf{CP}^{n-1} . This polynomial is called the *multi-resultant* for the system, see [27, chap. 11]. Note that the existence of a solution in \mathbf{CP}^{n-1} is equivalent to the existence of a non-zero solution in \mathbf{C}^n , since we have assumed that the equations are homogeneous. In this section we will describe a method for computing the multi-resultant of the system (6), following [27, sec. 82].

Let d_i be the degree of P_i , and let $L := 1 + \sum(d_i - 1)$; note that any monomial of degree L in the x_i 's must be divisible by $x_j^{d_j}$ for some j . Let $V_{n,L}$ be the vector space of homogeneous polynomials in x of degree exactly L ; a basis for $V_{n,L}$ is given by the set of monomials in x of degree exactly L . The

dimension of $V_{n,L}$ is the binomial coefficient

$$m := \binom{L+n-1}{n-1} = \binom{\sum d_i}{n-1}. \quad (7)$$

We may order the m basis elements for $V_{n,L}$ in “reverse lexicographical” order, with x_{n-1}^L first, next $x_{n-1}^{L-1}x_{n-2}$, etc. There is a second way to organize these m monomials, which is applicable in forming the multi-resultant of the system (6). We partition these m monomials up into n sets S_0, \dots, S_{n-1} as follows.

Let S_0 be the set of monomials which are divisible by $x_0^{d_0}$. Let S_1 be the set of monomials which are not divisible by $x_0^{d_0}$, but which are divisible by $x_1^{d_1}$. In general, let S_i be the set of monomials which are not divisible by any of $x_0^{d_0}, x_1^{d_1}, \dots, x_{i-1}^{d_{i-1}}$, but which are divisible by $x_i^{d_i}$.

From the definition of L , one sees immediately that the S_i are disjoint, and their union is the entire basis of $V_{n,L}$.

For each i , let T_i be the monomials in S_i divided by $x_i^{d_i}$; the elements of T_i are certain monomials of degree $L - d_i$. The sets T_i need not be disjoint. Note that we may obtain the set T_i directly as the set of monomials of degree $L - d_i$ which are not divisible by $x_0^{d_0}, \dots, x_{i-1}^{d_{i-1}}$.

We are now ready to describe a matrix M whose determinant is a multiple of the multi-resultant of the system (6). It is an $m \times m$ matrix, and we describe M by giving m row vectors of length m . The m row vectors are obtained one for each element of each S_i . Choose therefore an index i and a monomial $f = x_0^{e_0} \cdots x_{n-1}^{e_{n-1}}$ of S_i ; note then that $e_0 < d_0, \dots, e_{i-1} < d_{i-1}$, and $e_i \geq d_i$. Let $g := f/x_i^{d_i}$ be the corresponding element of T_i . Consider the polynomial gP_i ; since g has degree $L - d_i$ and P_i has degree d_i , gP_i has degree L and can therefore be written in terms of the m monomials given above. Since these elements have been ordered (in reverse lexicographical order), writing gP_i in terms of the basis vectors yields a row vector of coefficients of length m . It is this vector which forms one row of the matrix M . This process, performed for each element of each S_i , gives the m rows of M . We will call this matrix M the *multi-resultant matrix* of the system (6).

The reader may check that if $n = 2$, the multi-resultant matrix is the familiar Sylvester matrix for the resultant of two inhomogeneous polynomials in one variable (or equivalently, two homogeneous polynomials in two variables as we have set it up). Moreover, for any n , if each of the P_i 's is linear, then the above matrix is simply the matrix of coefficients of the P_i 's. In either case, the determinant of this matrix detects the existence of a nonzero solution to the system (6) in \mathbf{C}^n , or equivalently, a solution in \mathbf{CP}^{n-1} : this determinant is the multi-resultant in these cases. Note that in general, the number of nonzero entries in each row of the multi-resultant matrix is equal to the number of terms in the corresponding P_i . Thus the matrix is always sparse. Let R be the determinant of this multi-resultant matrix; R will be called the *multi-resultant* of the system (6).

If the system (6) has a common solution in \mathbf{CP}^{n-1} , the determinant R of the matrix M must be zero: each of the rows corresponds to a homogeneous polynomial of degree L in x , and if the original system (6) has a solution, then each of these homogeneous polynomials of degree L has a common nonzero root, since they are all merely multiples of the P_i 's. Therefore this set of polynomials cannot form a basis for $V_{n,L}$, and so the determinant of M must be zero. Hence we have the following

Lemma 1 *If the system (6) has a common solution in \mathbf{CP}^{n-1} , then the multi-resultant R vanishes.*

To illustrate the process, let us consider a small example with $n = 3$. For convenience, we switch to using the variables x , y , and z . Let

$$\begin{aligned} P_0 &:= x^2 - yz - 3y^2, \\ P_1 &:= xy - 2z^2, \\ P_2 &:= y^2 + yz - xz. \end{aligned}$$

Each of these homogeneous polynomials has degree 2, so $d_0 = d_1 = d_2 = 2$ and $L = 4$; therefore $m = 15$. The reverse lexicographical order on these 15 monomials in x , y , and z of degree 4 is

$$\{z^4, yz^3, y^2z^2, y^3z, y^4, xz^3, xyz^2, xy^2z, xy^3, x^2z^2, x^2yz, x^2y^2, x^3z, x^3y, x^4\}.$$

The sets S_i are

$$\begin{aligned} S_0 &= \{x^2z^2, x^2yz, x^2y^2, x^3z, x^3y, x^4\}, \\ S_1 &= \{y^2z^2, y^3z, y^4, xy^2z, xy^3\}, \\ S_2 &= \{z^4, yz^3, xz^3, xyz^2\}. \end{aligned}$$

The corresponding sets T_i are then

$$\begin{aligned} T_0 &= \{z^2, yz, y^2, xz, xy, x^2\}, \\ T_1 &= \{z^2, yz, y^2, xz, xy\}, \\ T_2 &= \{z^2, yz, xz, xy\}. \end{aligned}$$

The multi-resultant matrix M is now formed by multiplying elements of T_i by P_i and writing the coefficients out in reverse lexicographical order. For example, the 8-th row of the matrix is formed by taking the second element of T_1 (skipping over the 6 elements in T_0), which is “ yz ”, multiplying it by P_1 to obtain $xy^2z - 2yz^3$, and writing this out as a row vector, giving

$$(0 \ -2 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)$$

(since yz^3 is the second monomial and xy^2z is the eighth in the reverse lexicographical order). The full 15×15 multi-resultant matrix M for this example

is

$$\begin{pmatrix} 0 & -1 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -3 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -3 & 0 & 0 & 1 \\ -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

This matrix M does indeed have determinant zero, and a common root to the system is $x = 2, y = 1, z = 1$.

Remark 2 The multi-resultant matrix M , which has as entries the coefficients of the P_i 's, is clearly dependent on the order of the polynomials. If one changes the order of the polynomials, the list of degrees may change, the sets S_i and T_i may change, and the corresponding rows of the matrix may change. The vanishing of the determinant of the multi-resultant matrix M is only a necessary condition for the system (6) to have a common solution. The actual multi-resultant is the greatest common divisor (in the ring of polynomials of generic coefficients) of the determinants of these matrices, taken over all orderings of the polynomials in the original system, see [27, sec. 82].

Remark 3 Note that the vanishing of this multi-resultant detects common solutions in \mathbf{CP}^{n-1} ; the multi-resultant in this form is not especially sensitive only to finite real solutions.

3 The Use of the Multi-Resultant for Inhomogeneous Systems

Given a system (1) of n non-homogeneous polynomials in n variables, one expects that there will be a finite number of solutions in \mathbf{R}^n . Let us assume that this is so, and choose a variable y_i . If we fix a value for this y_i , we obtain from the system (1) a new system of n non-homogeneous polynomials in the other $n - 1$ variables; this new system can be homogenized by adding an auxiliary variable y_0 to obtain a system of n homogeneous polynomials in the n variables consisting of the other $n - 1$ y_j 's and the new y_0 :

$$Q(y_i) = \begin{pmatrix} Q_0(y_0, \dots, y_{i-1}, y_{i+1}, \dots, y_n) \\ \vdots \\ Q_{n-1}(y_0, \dots, y_{i-1}, y_{i+1}, \dots, y_n) \end{pmatrix} = 0. \quad (8)$$

Note that the coefficients of this homogeneous system are polynomial expressions in the coefficients of the original system (1) and the chosen variable y_i . Hence, the coefficients of the system (8) are polynomials in y_i . Let R_i be the multi-resultant of the system (8) as described in the previous section. R_i is the determinant of the multi-resultant matrix M_i , which has as entries the coefficients of the Q_j 's; hence R_i is itself a polynomial in y_i , (whose coefficients are polynomial expressions in the coefficients of the P_j 's). We will sometimes write $R_i = R_i(y_i)$ to emphasize this. We now can state our fundamental principle:

Lemma 4 *If the system (1) has a solution $(\bar{y}_1, \dots, \bar{y}_n) \in \mathbf{C}^n$, then for each i we have $R_i(\bar{y}_i) = 0$.*

The above statement is clear: if system (1) has such a solution, then the reduced system obtained by fixing $y_i = \bar{y}_i$ has the solution $(\bar{y}_1, \dots, \bar{y}_{i-1}, \bar{y}_{i+1}, \dots, \bar{y}_n)$; hence the homogenized system (8) has the corresponding solution in \mathbf{CP}^{n-1} obtained by setting $y_0 = 1$, for this value \bar{y}_i of y_i . Therefore $R_i(\bar{y}_i)$ must be zero, by the results of the previous section.

The reader will no doubt wonder about a converse to Lemma 4, especially for real solutions. Let us discuss this briefly. Suppose conversely, that $R_i(y_i)$ has a real root at $y_i = \bar{y}_i$. One would like to deduce that there is a real solution to the system (1) with i -th coordinate equal to \bar{y}_i . This is not the case, for several reasons.

- Firstly, the condition $R_i(\bar{y}_i) = 0$ is only necessary, not sufficient, for the system (8) to have a common solution in \mathbf{CP}^{n-1} with the value of y_i equal to \bar{y}_i : the actual resultant is only a divisor of $R_i(y_i)$, and hence $R_i(y_i)$ may have some roots which are irrelevant for our problem.
- Secondly, there may be a solution to (8) at infinity (with the extra variable y_0 set equal to 0) which will not show up as a solution to the system (1); all solutions to (1) correspond to solutions to (8) with $y_0 = 1$.
- Thirdly, there may be a complex solution to (8) for a real value of y_i ; the fact that y_i is real does not imply that all of the other coordinates of a solution must be real!

For these three reasons there is no possibility for a converse to Lemma 4. Since we are interested in real solutions to the system (1), we use Lemma 4 in the following form:

Lemma 5 *If the system (1) has a real solution $(\bar{y}_1, \dots, \bar{y}_n) \in \mathbf{R}^n$, then for each i , \bar{y}_i is a real root of $R_i(y_i)$.*

We will call the polynomial $R_i(y_i)$ obtained from the system (1) the *multi-resultant of (1) with respect to y_i* . Again note that it is dependent on the ordering of the polynomials P_j . The degree of the multi-resultant polynomial $R_i(y_i)$ is in general, approximately the product of the degrees of the P_j 's, which is typically rather large. Therefore if we are given the multi-resultant $R_i(y_i)$, the

computation of its real roots will be a numerically unstable problem: small errors in the coefficients of $R_i(y_i)$ can give large errors in the location of the roots. Furthermore, $R_i(y_i)$ is the determinant of the corresponding multi-resultant matrix $M_i(y_i)$, which is typically large and sparse. Thus it is in general very difficult to calculate the coefficients of the polynomial $R_i(y_i)$. For the case that all coefficients of (1) are integers, Collins [5] uses a rather costly exact integer arithmetic. We therefore propose to replace the condition that $R_i = 0$ by the equivalent, but stable, condition that the minimum eigenvalue of $M_i^* M_i$ is 0. We formulate this by requiring

$$\lambda_{\min}(y_i) := \min_{\|u\|=1} \|M_i(y_i)u\|^2 = 0. \quad (9)$$

Therefore Lemma 5 can be rephrased as

Lemma 6 *If the system (1) has a solution $(\bar{y}_1, \dots, \bar{y}_n) \in \mathbf{R}^n$, then for each i , $\lambda_{\min}(\bar{y}_i)$ is zero.*

We would like to make a few remarks about the formation and representation of the matrix $M_i(y_i)$ in the computer. Of course, the starting point is the “raw data” of the system (1). After choosing a variable y_i , it is not difficult to represent each P_j as a polynomial in the other y_k ’s, with coefficients being polynomials in y_i . Indeed, a polynomial in y_i is stored as a structure (using the conventions of the C programming language)

```
polynomial_in_y_i = {
    degree (an integer d, say);
    coefficient array c[0], c[1], ..., c[d];
    value;}
```

and the terms of the polynomials P_j are then stored as a structure

```
term_of_P_j = {
    exponent array e[1], ..., e[n-1];
    polynomial_in_y_i;}
```

The polynomials P_j themselves are then stored as an array of term_of_P_j’s. It is an easy matter to implement the addition of the homogenizing variable y_0 , and to calculate the necessary degrees, etc.

Once the variable y_i has been chosen, and the polynomials P_j have been represented, the construction of the matrix $M_i(y_i)$ proceeds directly. Each row is stored as a linked list of entries, and in each entry there is recorded the column number, a pointer to the polynomial for this entry, and a pointer to the next entry in that column. This allows us to efficiently multiply a vector by either the matrix M_i or by its transpose M_i^* .

There are really only two interesting procedures: for each j , we need to proceed through the list T_j of monomials (so that we can multiply P_j by each element of T_j), and we need to calculate the reverse lexicographical order of a

monomial (so that we can figure out the column of the multi-resultant matrix M_i in which to put any particular term). These two combinatorial procedures are easily implemented, the second by a recursive method.

4 A Conjugate Gradient Method on the Unit Sphere

The above minimal value $\lambda_{\min}(y_i)$ in (9) can be viewed as the smallest eigenvalue of the semi-definite matrix $M_i(y_i)^*M_i(y_i)$. Since the multi-resultant (of dimension m , see (7), is typically very large and very sparse, a Lanczos method is the standard method for calculating $\lambda_{\min}(y_i)$, see, e.g., [12]. However, since we are only interested in the smallest eigenvalue, and since a conjugate gradient method for (9) is simple to implement and incorporate, and has less overhead, we have chosen to use a modification of the latter for our purposes. Both methods need each of the following two products

$$v = M_i(y_i)u \quad \text{and} \quad w = M_i(y_i)^*v. \quad (10)$$

per step. From a theoretical point of view, it is not difficult to see that the Lanczos method generates a better estimate of $\lambda_{\min}(y_i)$ after k steps (for $k < m$) than the conjugate gradient method. It remains to be investigated whether the better estimate rewards the higher overhead of the Lanczos method.

Let us begin by formulating a modification of the nonlinear conjugate gradient method for the case that a nonlinear functional has to be minimized on the sphere:

$$\min_{\|u\|=1} \varphi(u). \quad (11)$$

For a survey of the linear conjugate gradient method and convergence results we refer to [25], for the nonlinear case to [9, 18]. The important modification for the constrained case (11) is that the gradients always have to be projected onto the current tangent space. This leads to the following conjugate gradient method on the unit sphere:

Algorithm 7

input $u_0 \in \mathbf{R}^N$ such that $\|u_0\|_2 = 1$; *initial point*
 $\tilde{u}_0 := \nabla\varphi(u_0)$; *gradient*
 $\lambda_0 := \langle u_0, \tilde{u}_0 \rangle$; *scalar product*
 $g_0 := \tilde{u}_0 - \lambda_0 u_0$; *projected gradient*
 $d_0 := g_0$; *initial conjugate gradient*
for $n = 0, 1, \dots$ **do**
 $\rho_n := \arg \min_{\rho > 0} \varphi(\cos(\rho)u_n - \sin(\rho)\|d_n\|_2^{-1}d_n)$; *search*
 $u_{n+1} := \cos(\rho_n)u_n - \sin(\rho_n)\|d_n\|_2^{-1}d_n$; *new point*
 $\tilde{u}_{n+1} := \nabla\varphi(u_{n+1})$; *new gradient*
 $\lambda_{n+1} := \langle u_{n+1}, \tilde{u}_{n+1} \rangle$; *scalar product*
 $g_{n+1} := \tilde{u}_{n+1} - \lambda_{n+1}u_{n+1}$; *projected gradient*
 $d_n := d_n - \langle d_n, u_{n+1} \rangle u_{n+1}$; *newly projected conjugate gradient*
 $\gamma_n := \langle g_{n+1} - g_n, g_{n+1} \rangle \|g_n\|^{-2}$; *Polak and Ribière modification*
 $d_{n+1} := g_{n+1} + \gamma_n d_n$; *new conjugate gradient*
until convergence. *stopping criterion*

The fact that the functional φ_1 in (4) is quadratic allows us to simplify Algorithm 7 to the following conjugate gradient algorithm for obtaining the minimal eigenvalue. For convenience, let us introduce the matrix $A := M_i(y_i)^* M_i(y_i)$.

Algorithm 8

```

input
  begin
     $\epsilon_1, \epsilon_2, \epsilon_3 > 0$  tolerances
     $u_0 \in \mathbf{R}^N$  such that  $\|u_0\|_2 = 1$ ; initial point
  end
   $\tilde{u}_0 := Au_0$ ; gradient
   $\lambda_0 := \langle u_0, \tilde{u}_0 \rangle$ ; scalar product, Rayleigh quotient
   $g_0 := \tilde{u}_0 - \lambda_0 u_0$ ; projected gradient
   $d_0 := g_0$ ; initial conjugate gradient
   $\tilde{d}_0 := Ad_0$ ;
  for  $n = 0, 1, \dots$  do
    if  $\|d_n\| < \epsilon_3$  then quit; avoids cancellation effects
     $\rho_n := \arg \min_{-\frac{\pi}{2} < \rho < \frac{\pi}{2}} \varphi_1(\cos(\rho)u_n - \sin(\rho)\|d_n\|_2^{-1}d_n)$ ; search
     $u_{n+1} := \cos(\rho_n)u_n - \sin(\rho_n)\|d_n\|_2^{-1}d_n$ ; new point
     $\tilde{u}_{n+1} := \cos(\rho_n)\tilde{u}_n - \sin(\rho_n)\|d_n\|_2^{-1}\tilde{d}_n$ ; new gradient
     $\lambda_{n+1} := \langle u_{n+1}, \tilde{u}_{n+1} \rangle$ ; scalar product, Rayleigh quotient
     $g_{n+1} := \tilde{u}_{n+1} - \lambda_{n+1}u_{n+1}$ ; projected gradient
     $d_n := d_n - \langle d_n, u_{n+1} \rangle u_{n+1}$ ; newly projected conjugate gradient
     $\gamma_n := \langle g_{n+1} - g_n, g_{n+1} \rangle \|g_n\|^{-2}$ ; Polak and Ribière modification
     $d_{n+1} := g_{n+1} + \gamma_n d_n$ ; new conjugate gradient
     $\tilde{d}_{n+1} := Ad_{n+1}$ ; one call of (10) per step
  until  $|\lambda_{n+1} - \lambda_n| < \epsilon_1 + \epsilon_2 |\lambda_{n+1}|$ . stopping criterion

```

Remark 9 Since φ_1 is quadratic, the step “search on a great circle” can easily be calculated in the following way: we consider the positive definite 2×2 -matrix

$$B := \begin{pmatrix} \langle u_n, Au_n \rangle & \left\langle u_n, A \frac{d_n}{\|d_n\|} \right\rangle \\ \left\langle u_n, A \frac{d_n}{\|d_n\|} \right\rangle & \left\langle \frac{d_n}{\|d_n\|}, A \frac{d_n}{\|d_n\|} \right\rangle \end{pmatrix}$$

Then the smaller eigenvalue of B is λ_{n+1} , and

$$\begin{pmatrix} \cos(\rho_n) \\ -\sin(\rho_n) \end{pmatrix}$$

is the corresponding eigenvector. The calculations are now easy to perform.

Remark 10 The stopping criterion step is governed by a mixture of a maximal relative error ϵ_2 and a maximal absolute error ϵ_1 for the minimal eigenvalue

$\lambda_{\min}(y_i)$. It is motivated by a wish to avoid excessive iterations which will just improve the approximation of the corresponding eigenvector, since the latter is of no interest to us. On the other hand, the starting vector u_0 is typically obtained from a previous run of the algorithm for a different y_i -value. Hence, it may occur that u_0 is very near an eigenvector corresponding to an eigenvalue $\lambda(y_i) \neq \lambda_{\min}(y_i)$. Numerical experiments have shown that in this case the method generates a sequence $\{\lambda_k\}$ which usually moves away from $\lambda(y_i)$ after one initially slow step, even if the multiplicity of $\lambda(y_i)$ is higher. Hence it is recommended to force the algorithm to perform at least one step.

Remark 11 Our examination of the literature reveals only one algorithm (by Geradin [11]) related to that above. It has found several applications in the engineering literature with regard to the generalized eigenvalue problem. Instead of minimizing a quadratic functional on the unit sphere, Geradin's algorithm incorporates the constraint into the functional, namely the Rayleigh quotient, and then performs standard nonlinear conjugate gradient steps (Fletcher-Reeves). Since it incorporates a linear line search instead of a search along a great circle, it appears to be less suited for our purposes.

Remark 12 We remind the reader that little is known about the convergence of CG methods for the nonlinear case, see [18]. However, our functional (4) is quadratic, and the constraint is quite simple. We conjecture that it should be possible to modify the convergence proof of [19] for this case.

Our aim now is to use the above algorithm as a device for detecting all zeros of the function $y_i \mapsto \lambda_{\min}(y_i)$ in a given interval $[a_i, b_i]$. The above process is performed for all variables y_i , $i = 1, \dots, n$. Normally, this task could be handled by a numerical continuation method for the simplest possible case: following a path in \mathbf{R}^2 . See, e.g., [1] for a survey of general continuation methods. Unfortunately, the function λ_{\min} is only piecewise smooth. Hence, the sophisticated steplength strategies (i.e., choice of increments for y_i) which are available, see, e.g., [8, 15], must be handled with care. This is an aspect which requires further study, possibly along the lines of [26]. For the present, we have used a fixed stepsize $y_i \in \{t_0, \dots, t_k\}$ with $t_i := a + i(b - a)/k$.

If a certain criterion is satisfied for t_{i-1} , t_i and t_{i+1} , then one can minimize λ_{\min} on $[t_{i-1}, t_{i+1}]$ via either a standard line search algorithm or a method which is especially adapted to this situation. If the obtained minimal value is small enough, we add the corresponding y_i -value to our list Z_i , see (2). The design of a suitable line search strategy adapted for our purposes is currently being investigated.

5 Examples

We illustrate the method with two examples. Because the purpose of this paper is only to introduce this method, our examples will be simple ones with a small number of variables and little numerical complexity. Furthermore, we do

not employ any line search strategy to obtain the best minimal point, but we content ourselves with a fixed step size. More complex examples will be studied elsewhere in the context of a discussion on how the present method can be improved and be made more efficient and more generally applicable.

Our first example concerns a simple model of a robot in the plane with two arms. The first arm has length b and is fixed at one end (its tail) in such a way that it can rotate with an angle α . The second arm has length 1 and is fixed with its tail at the head of the first arm in such a way that it can rotate. The angle of rotation with respect to the first arm is denoted by β . Both angles are parametrized with respect to a stereographic projection, i.e.,

$$\begin{aligned}\cos \alpha &= \frac{1-x^2}{1+x^2}, & \sin \alpha &= \frac{2x}{1+x^2}, \\ \cos \beta &= \frac{1-y^2}{1+y^2}, & \sin \beta &= \frac{2y}{1+y^2}.\end{aligned}$$

If we denote by q_1 and q_2 the two co-ordinates of the head of the second arm, then we obtain the following two equations:

$$\begin{aligned}x^2y^2 + \frac{2}{q_2}x^2y + \frac{2}{q_2}(1-b)xy^2 + x^2 + y^2 - \frac{2}{q_2}(1+b)x - \frac{2}{q_2}y + 1 &= (12) \\ (1-b-q_1)x^2y^2 + (-1-b-q_1)x^2 - 4xy + (-1+b-q_1)y^2 + (1+b-q_1) &= 0\end{aligned}$$

Of course the object is to find the angle parameters x and y for a given position (q_1, q_2) . Our example uses the following values: $b = 0.5$, $q_1 = 0.8$, $q_2 = 0.4$. The calculations were performed in double precision. In the stopping criterion of Algorithm 8 we used the parameters $\epsilon_1 = 10^{-14}$ and $\epsilon_2 = 10^{-8}$. We scanned for a solution of the y -parameter in the interval $[-2, 2]$, using an increment of 0.1. The resultant matrix has dimension 4. Table 1 shows the minimal eigenvalues λ and the number of gradient steps ν which were necessary to obtain λ .

The table shows a considerable decrease in the λ -value near the points $y = -1.6$ and $y = 1.6$. From the geometrical meaning of the equations it is not difficult to see that the system has exactly two solutions, and the corresponding two y -values must be symmetric with respect to the origin. In fact, using some trigonometry, one can see that the actual solution has $y \approx \pm 1.6237$.

Our second example describes the intersection of a sphere in \mathbf{R}^3 with two paraboloids:

$$\begin{aligned}x^2 + y^2 + z^2 - 1 &= 0, \\ z - x^2 - y^2 &= 0, \\ y - x^2 - z^2 &= 0.\end{aligned}\tag{13}$$

It is readily seen that the two unique solutions to the problem are

$$\begin{aligned}y &= z = \frac{\sqrt{5}-1}{2} \approx 0.618, \\ x &= \pm\sqrt{z-y^2} \approx \pm 0.486.\end{aligned}$$

Table 1:

y	λ	ν	y	λ	ν
-2.0	2.46863570 e-01	9	0.0	3.60756482 e-01	8
-1.9	1.28132892 e-01	6	0.1	3.49687290 e-01	8
-1.8	5.00900404 e-02	6	0.2	3.42325181 e-01	8
-1.7	8.97631589 e-03	7	0.3	3.37446277 e-01	7
-1.6	8.23301590 e-04	8	0.4	3.33388868 e-01	7
-1.5	2.12266955 e-02	7	0.5	3.28238812 e-01	7
-1.4	6.50965198 e-02	7	0.6	3.20023882 e-01	7
-1.3	1.26443063 e-01	7	0.7	3.06922164 e-01	7
-1.2	1.98305363 e-01	7	0.8	2.87477855 e-01	6
-1.1	2.72965508 e-01	7	0.9	2.60813037 e-01	6
-1.0	3.42571677 e-01	7	1.0	2.26825399 e-01	6
-0.9	4.00164684 e-01	7	1.1	1.86365806 e-01	6
-0.8	4.40869912 e-01	7	1.2	1.41392732 e-01	6
-0.7	4.62801090 e-01	7	1.3	9.51016474 e-02	6
-0.6	4.67228862 e-01	6	1.4	5.20269492 e-02	6
-0.5	4.57885610 e-01	7	1.5	1.81131862 e-02	6
-0.4	4.39716545 e-01	8	1.6	7.51983608 e-04	7
-0.3	4.17611582 e-01	8	1.7	8.78174730 e-03	6
-0.2	3.95529363 e-01	8	1.8	5.24487744 e-02	6
-0.1	3.76124570 e-01	8	1.9	1.43330509 e-01	6

We scan for a solution of the y -parameter in the interval $[-0.9, 0.9]$ and of the x -parameter in the interval $[-0.7, 0.7]$. In both cases, the increment is 0.05, and the resultant matrix has dimension 15. Tables 2–3 show the minimal eigenvalues λ and the number of gradient steps ν which were necessary to obtain λ .

We see again that the λ -values decrease considerably near the solutions. When we try to scan the z -values, it turns out that the resultant matrix is singular for every z -value. Therefore the conjugate gradient method as described in this paper i.e., to calculate the minimal eigenvalue of the semi-definite matrix $M(z)^*M(z)$ where $M(z)$ is the resultant with respect to the z -variable, will not produce any information. The remedy which we suggest in this case is to use a Lanczos method and scan the variable z for the second smallest eigenvalue. More generally, one can use the Lanczos method to calculate the smallest eigenvalue which is not identical to zero with respect to z , and therefore the method can be applied with arbitrary nullity of the resultant matrix. The use of this technique to handle these singular situations arising in the resultant matrix application will be discussed in a further paper.

6 Concluding Remarks

We have presented a method for finding the real solutions to a system of polynomial equations with real coefficients. The numerical experiments we have performed, though of a preliminary nature, indicate that the method is very

Table 2:

y	λ	ν	y	λ	ν
-0.90	1.11142172 e-02	38	0.00	1.36209400 e-02	11
-0.85	1.22074168 e-02	14	0.05	1.26294917 e-02	11
-0.80	1.32075526 e-02	12	0.10	1.15013381 e-02	12
-0.75	1.41004617 e-02	12	0.15	1.02433447 e-02	12
-0.70	1.48773611 e-02	12	0.20	8.87245170 e-03	12
-0.65	1.55339989 e-02	12	0.25	7.41884362 e-03	12
-0.60	1.60695907 e-02	10	0.30	5.92815875 e-03	12
-0.55	1.64857079 e-02	10	0.35	4.46154156 e-03	12
-0.50	1.67852362 e-02	10	0.40	3.09238203 e-03	12
-0.45	1.69714691 e-02	10	0.45	1.89923233 e-03	13
-0.40	1.70473662 e-02	10	0.50	9.55648350 e-04	13
-0.35	1.70149781 e-02	10	0.55	3.19136569 e-04	14
-0.30	1.68750302 e-02	10	0.60	2.22183242 e-05	17
-0.25	1.66266596 e-02	10	0.65	6.82224117 e-05	16
-0.20	1.62673095 e-02	11	0.70	4.32829909 e-04	14
-0.15	1.57928036 e-02	10	0.75	1.07041543 e-03	14
-0.10	1.51976509 e-02	10	0.80	1.92286029 e-03	12
-0.05	1.44756558 e-02	11	0.85	2.92828850 e-03	12

Table 3:

x	λ	ν	x	λ	ν
-0.70	5.97472082 e-02	23	0.00	3.68714615 e-02	14
-0.65	3.15704112 e-02	18	0.05	3.61856001 e-02	14
-0.60	1.35308623 e-02	18	0.10	3.41545696 e-02	15
-0.55	3.75664960 e-03	18	0.15	3.08617434 e-02	17
-0.50	1.59449310 e-04	20	0.20	2.64585458 e-02	17
-0.45	8.93505381 e-04	19	0.25	2.11834272 e-02	17
-0.40	4.43483527 e-03	17	0.30	1.53887688 e-02	17
-0.35	9.57555523 e-03	17	0.35	9.57555523 e-03	17
-0.30	1.53887688 e-02	17	0.40	4.43483527 e-03	17
-0.25	2.11834272 e-02	17	0.45	8.93505381 e-04	19
-0.20	2.64585458 e-02	17	0.50	1.59449310 e-04	20
-0.15	3.08617434 e-02	17	0.55	3.75664960 e-03	17
-0.10	3.41545696 e-02	17	0.60	1.35308623 e-02	17
-0.05	3.61856001 e-02	15	0.65	3.15704112 e-02	17

promising. In contrast to the homotopy method, our algorithm is especially designed to find real solutions; in addition, the user may restrict the algorithm to specific intervals of interest.

Let us indicate some directions of future refinements which we judge to be important to a more professional implementation of the algorithm.

1. The method should incorporate some scaling techniques to avoid bad conditioning. Some preprocessing should also be performed. For instance, automatic detection of zero rows or columns in the resultant matrix and a possible reordering of the equations or variables should be considered.
2. To make a more efficient use of the gradient (respectively Lanczos) steps, a more sophisticated line search algorithm needs to be devised. This search need not be very precise since at the final stage a Newton type method would be employed to refine the precision of the suggested zero points.
3. As mentioned above, since the rank of the resultant matrix is not necessarily maximal or known a priori, the phenomenon of singularities in the resultant matrix will require a delicate incorporation of a Lanczos method in the minimization process.
4. The relationship between the multiplicity of a root (in one of the variables) and the multiplicity of the zero eigenvalue of the resultant matrix needs to be investigated. A more precise understanding of this relationship could be useful, for example in the combinatorial analysis of the suggested coordinate values of zero points in \mathbf{R}^n .
5. This paper represents preliminary work without extensive experimentation. Though we demonstrate the workability of our approach, what is ultimately needed are experiments in which a full implementation of our approach is tested in ways that demonstrate its practical feasibility and that allow comparison of timings, robustness, storage requirements, etc., with alternative methods, especially homotopy methods.

These points are currently under study and will be the subject of further papers.

References

- [1] E. L. ALLGOWER AND K. GEORG, *Numerical Continuation Methods: An Introduction*, vol. 13 of Series in Computational Mathematics, Springer Verlag, Berlin, Heidelberg, New York, 1990. pp 388.
- [2] W. AUZINGER AND H. J. STETTER, *A study of numerical elimination for the solution of multivariate polynomial systems*. Preprint, unpublished provisional version, Technische Universität Wien, Austria, 1989.

- [3] P. BRUNOVSKÝ AND P. MERAVÝ, *Solving systems of polynomial equations by bounded and real homotopy*, Numer. Math., 43 (1984), pp. 397–418.
- [4] S. N. CHOW, J. MALLET-PARET, AND J. A. YORKE, *A homotopy method for locating all zeros of a system of polynomials*, in Functional Differential Equations and Approximation of Fixed Points, H.-O. Peitgen and H.-O. Walther, eds., vol. 730 of Lecture Notes in Math., Berlin, Heidelberg, New York, 1979, Springer Verlag, pp. 77–88.
- [5] G. E. COLLINS, *The calculation of multivariate polynomial resultants*, J. Assoc. Comput. Mach., 18 (1971), pp. 515–532.
- [6] J. K. CULLUM, W. KERNER, AND R. A. WILLOUGHBY, *A generalized nonsymmetric Lanczos procedure*, Comp. Phys. Comm., 53 (1989), pp. 19–48.
- [7] J. K. CULLUM AND R. A. WILLOUGHBY, *Computing eigenvalues of large matrices, some Lanczos algorithms and a shift and invert strategy*, in Proceedings of the Fifth Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, S. Gomez and J. P. Hennart, eds., SIAM Publications, 1990. to appear.
- [8] C. DEN HELJER AND W. C. RHEINBOLDT, *On steplength algorithms for a class of continuation methods*, SIAM J. Numer. Anal., 18 (1981), pp. 925–948.
- [9] R. FLETCHER, *Practical Methods of Optimization. Vol. I: Unconstrained Optimization, Vol. II: Constrained Optimization*, John Wiley and Sons, Chichester, 1980–1981.
- [10] C. B. GARCIA AND W. I. ZANGWILL, *Finding all solutions to polynomial systems and other systems of equations*, Math. Programming, 16 (1979), pp. 159–176.
- [11] M. GERADIN, *The computational efficiency of a new minimization algorithm for eigenvalue analysis*, J. Sound and Vibration, 19 (1971), pp. 319–331.
- [12] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.
- [13] E. R. HANSEN, *An overview of global optimization using interval analysis*, in Reliability in Computing, R. E. Moore, ed., Academic Press, 1988.
- [14] R. B. KEARFOTT, *Some tests of generalized bisection*, ACM TOMS, 13 (1987), pp. 197–220.
- [15] ———, *An interval step control for continuation methods*. To appear in Math. Comp., 1989.

- [16] ———, *Preconditioners for the interval Gauss-Seidel method*, SIAM J. Numer. Anal., 27 (1990), pp. 804–822.
- [17] T.-Y. LI, T. SAUER, AND J. A. YORKE, *The cheater’s homotopy: An efficient procedure for solving systems of polynomial equations*, SIAM J. Numer. Anal., 26 (1989), pp. 1241–1251.
- [18] G. P. MCCORMICK, *Nonlinear Programming. Theory, Algorithms and Applications*, John Wiley and Sons, New York, 1983.
- [19] G. P. MCCORMICK AND K. RITTER, *Alternate proofs of the convergence properties of the conjugate-gradient method*, J. Optim. Theory Appl., 13 (1974), pp. 497–518.
- [20] A. P. MORGAN, *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1987.
- [21] A. P. MORGAN AND A. J. SOMMESE, *Computing all solutions to polynomial systems using homotopy continuation*, Appl. Math. Comput., 24 (1987), pp. 115–138.
- [22] A. P. MORGAN, A. J. SOMMESE, AND L. T. WATSON, *Finding all solutions to polynomial systems using HOMPACK*. Res. Pub. GMR-6109, G. M. Research Labs, 1988.
- [23] A. NEUMAIER, *Interval Methods for Systems of Equations*, Cambridge University Press, 1990.
- [24] Y. SAAD, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric linear systems*, SIAM J. Numer. Anal., 19 (1982), pp. 485–506.
- [25] J. STOER, *Solution of large linear systems of equations by conjugate gradient type methods*, in *Mathematical Programming: The State of the Art*, A. Bachem, M. Grötschel, and B. Korte, eds., Berlin, Heidelberg, New York, 1983, Springer Verlag, pp. 540–565.
- [26] A. USHIDA AND L. O. CHUA, *Tracing solution curves of nonlinear equations with sharp turning points*, Internat. J. Circuit Theory Appl., 12 (1984), pp. 1–21.
- [27] B. L. VAN DER WAERDEN, *Modern Algebra*, Ungar, New York, 1953. Volumes I and II.
- [28] L. T. WATSON, S. C. BILLUPS, AND A. P. MORGAN, *HOMPACK: A suite of codes for globally convergent homotopy algorithms*, ACM Trans. Math. Software, 13 (1987), pp. 281–310.