A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots

Robert M. Corless, Patrizia M. Gianni, and Barry M. Trager*

Department of Applied Mathematics
University of Western Ontario, London, CANADA N6A 5B7
Department of Mathematics
University of Pisa, 56127 Pisa, ITALY
IBM T. J. Watson Research Center
P. O. Box 218, Yorktown Heights, New York, 10598 USA

Abstract

We discuss the use of a single generic linear combination of multiplication matrices, and its reordered Schur factorization, to find the roots of a system of multivariate polynomial equations. The principal contribution of the paper is to show how to reduce the multivariate problem to a univariate problem, even in the case of multiple roots, in a numerically stable way.

1 Introduction

The technique of solving systems of multivariate polynomial equations via eigenproblems has become a topic of active research (with applications in computer-aided design and control theory, for example) at least since the papers [2, 6, 9]. One may approach the problem via various resultant formulations or by Gröbner bases. As more understanding is gained, it is becoming clearer that eigenvalue problems are the "weakly nonlinear nucleus to which the original, strongly nonlinear task may be reduced" [13].

Early works concentrated on the case of simple roots. An example of such was the paper [5], which used a numerical adaptation of a resultant technique due to Lazard to attack the problem directly, without reference to Gröbner bases.

In this present paper we continue with this approach, and give a numerically stable method for reducing a multivariate polynomial system to a univariate problem, paying particular attention to the multiple root case.

The series of papers by Stetter and co-workers, exemplified by [11, 12], also consider multiple roots and develop a very useful theory. This present paper addresses an aspect of the multiple root problem taken in [12] as a working datum, namely the approximate location z_0 of a k-fold zero of a nearby polynomial system.

*This research was supported in part by the Natural Science and Engineering Research Council of Canada, by IBM T. J. Watson Research Center, and also benefited from the contribution of C.N.R., M.U.R.S.T, ESPRIT reactive LTR 21024 FRISCO.

Intended for ISSAC '97, Maui

The main result of this paper is that we can reduce the multivariate problem, by unitary and hence numerically stable manipulations, to a univariate problem. The procedure is applicable in the inexactly known coefficient case, but is also applicable in the exact-coefficient case, once the decision to use floating-point arithmetic has been taken. Indeed, the procedure of this paper begins by assuming that the multiplication matrices M_j for the system have been computed already, and for the purposes of this procedure it does not matter if the SVD adaptation of Lazard's direct approach, detailed in [5], is used, or if instead the multiplication matrices are computed from a normal set obtained by first computing a Gröbner basis, as in [4] for example.

If the multiple or near-multiple roots of the resulting univariate problem can easily be clustered, where the roots of each cluster are reasonably well separated from the roots of other clusters, then the procedure in this paper gives good results (and the *a posteriori* condition number analysis will tell us so). If we are faced with a pathological problem in which the clusters are smeared and cannot be resolved automatically, human judgement may be required.

Finally, the main technique explored in this paper, namely the post-processing of the Schur factorization of a generic combination of nearly commuting matrices, is of independent numerical interest because a considerable number of algorithms in numerical linear algebra can be thought of as simultaneous diagonalization of commuting matrices (see the references of [3]). We show in this paper that simple post-processing to re-order the Schur factorization can allow the use of off-the-shelf technology to bring a nearly commuting family to nearly block upper triangular form. This works even in the case of multiple eigenvalues and the procedure is quite numerically stable.

2 The Numerical Procedure

Notes and justifications for the following steps appear in section 4. We assume that the system of m approximately known polynomials $F = f_i$ in the n variables x_j, $1 \leq j \leq n$, are given, together with a tolerance or a schema which can be used to decide how many roots there are[1].

[1]It has been pointed out to us (G. Chudnovsky, private communication) that the simple use of a tolerance for the SVD is not sufficient for large problems, which suffer from 'smearing'. More sophisticated schemes are needed for large problems.

The procedure is as follows.

1. Form the multiplication matrices M_j, for $1 \le j \le n$, as in [5]. [Alternatively, if the F are exactly known, then compute a Gröbner basis and then the multiplication matrices M_j. See for example [4] for an introduction to this approach.]

2. Choose n random numbers $\alpha_j > 0$, scale them so that $\sum \alpha_j = 1$, and form the generic linear (convex) combination $M = \sum \alpha_j M_j$.

3. Compute the Schur factorization of $M = U^* T U$, where U is unitary and T is upper triangular. The ordering of the eigenvalues (the diagonal elements of T) is not important at this stage. The matrix U will be needed, so it must be computed (though it may be stored as a sequence of products of simpler unitary matrices).

4. Apply a clustering procedure, for example the methods of [7] or the heuristics used in [9], to decide the multiplicity and grouping of each root cluster.

5. Use plane rotations or other stable techniques to reorder the Schur factorization of M so that all roots in a cluster appear together on the diagonal of T, and update the unitary matrix U so that it corresponds to this ordering. After this step has been completed, we have $M = U^* T U$ with all clustered roots appearing next to one another on the diagonal of T (*i.e.* we have relabelled T and U). As before, storing the new U as a product of simpler matrices is acceptable.

6. Compute the nearly block upper triangular matrices $M_j' = U M_j U^*$. The number of blocks in any M_j' (each M_j has the same number) is the number of distinct roots. The trace of corresponding blocks in the M_j', $1 \le j \le n$, divided by the size of the blocks, gives the values of x_j at each root. In other words the values of roots of F are given by the averages of the diagonal entries of each block.

7. The condition numbers of each eigenvalue cluster are already available if the clustering heuristics of [9] are used. Use these and the size of the entries in each M_j' below the diagonal blocks to estimate the sizes of the errors in the roots of F.

3 Theory in the Exact Case

We give a brief sketch of the main ideas, in the exact arithmetic and exact data case. This makes the algebraic content of what we wish to accomplish more accessible. We will then give the analogous development for inexact data and floating-point arithmetic.

Proposition 1. Let M and N be commuting matrices over a field k, and let $p(x)$ be a polynomial over k. Then kernel $p(M)$ is an invariant subspace for both M and N.

Proof: Since M and N commute, N and $p(M)$ also commute. Choose a vector v in $\ker p(M)$, *i.e.* such that $p(M)v = 0$. Then $p(M)Nv = Np(M)v = 0$ and therefore Nv is in the kernel of $p(M)$. Similarly $p(M)Mv = Mp(M)v = 0$ so Mv is in the kernel of $p(M)$.

Definition. A block (upper) triangular matrix is a block matrix such that the blocks along the diagonal are square and the blocks below the diagonal are zero.

If we talk about several block matrices, we assume they are partitioned identically.

Proposition 2. If M and N are block triangular matrices then the product is block triangular. Note the product is well defined given our assumptions.

Proposition 3. If M is block triangular and $p(x)$ is a polynomial then $p(M)$ is block triangular.

Proposition 4. If

$$M = \begin{bmatrix} A & B \\ 0 & C \end{bmatrix} \tag{1}$$

is a block triangular matrix such that the eigenvalues of A and C are disjoint, and M commutes with a matrix N, then N is a block triangular matrix of the same shape.

Proof. There exists a polynomial $p(x)$ (for example the minimal polynomial for A) such that $p(A) = 0$ and $p(C)$ is invertible. Since M commutes with N, $p(M)$ commutes with N. Now

$$p(M) = \begin{bmatrix} p(A) & X \\ 0 & p(C) \end{bmatrix} \tag{2}$$

$$= \begin{bmatrix} 0 & X \\ 0 & p(C) \end{bmatrix}, \tag{3}$$

and if we partition

$$N = \begin{bmatrix} N_1 & N_2 \\ N_3 & N_4 \end{bmatrix} \tag{4}$$

then the lower left block of $p(M)N$ is equal to

$$0 \cdot N_1 + p(C)N_3 = p(C)N_3$$

while the lower left block of $Np(M)$ is equal to

$$N_3 \cdot 0 + N_4 \cdot 0 = 0 .$$

Since $Np(M) = p(M)N$ we have that $p(C)N_3 = 0$. Since $p(C)$ is invertible, $N_3 = 0$ so N is block triangular of the same shape.

Proposition 5. If $M_1, \ldots, M_n$ are a commuting family of matrices over a field of characteristic zero then a generic (that is, random) linear combination of the M_j's has as many distinct eigenvalues as the family has distinct collections of eigenvalues, with probability 1.

Proposition 6. If a generic linear combination of the M_j is in block triangular form such that each block has constant eigenvalue (that is, each block has only one eigenvalue) and different blocks have different eigenvalues, then each M_j will be in block triangular form such that each block has constant eigenvalue. These eigenvalues can be recovered by taking the trace of each block divided by its size.

Propositions 5 and 6 may be proved by techniques similar to those in [4].

Note that being given a family of commuting matrices is equivalent to being given a zero dimensional ideal in a polynomial ring, *i.e.* consider the surjective map from $k[x_1, \ldots, x_n] \to k[M_1, \ldots, M_n]$ sending x_i to M_i. The kernel is a zero dimensional ideal I such that $k[x_1, \ldots, x_n]/I$ is isomorphic to $k[M_1, \ldots, M_n]$. The eigenvalue sequences associated with corresponding blocks of the M_i in block triangular form give all solutions of the ideal I.

If we want to perform a primary decomposition of I over a ground field of characteristic zero, and $p(z)$ is a multiple of the minimal polynomial for a generic combination of the M_j, then the irreducible factorization of $p(z) = \prod p_i(z)^{e_i}$ can be used to construct a primary decomposition of I. That is, we factor the univariate polynomial $p(z)$, and then replace z by the corresponding generic combination $z = \sum \alpha_j x_j$. Then $I = \cap(I + p_i(\sum \alpha_j x_j)^{e_i})$ gives the complete primary decomposition of I.

A trivial corollary to this is that if you have a family of commuting matrices, one of which is in triangular form with distinct eigenvalues, then they are all in triangular form (use Prop. 4 by induction down to 1 by 1 blocks).

Thus, bringing a generic linear combination into Schur form will bring them all into Schur form if there are no multiple solutions. Further, if there are multiple solutions and we bring a generic linear combination into block Schur form with each block having constant eigenvalues, then the same transformation will bring them all into block Schur form with constant eigenvalues in the blocks.

4 Notes and Justification

In this section we comment on and justify the numerical procedure outlined in Section 2, for the inexact data and floating-point arithmetic case.

4.1 Formation of M_j

The first remark is that the overall computational complexity, for many interesting polynomials which have fewer roots than might be predicted from the Bezout bound, is dominated by the cost of step 1. The formation of the Lazard matrix and the subsequent formation of the multiplication matrices M_j is (singly) exponential in the number of variables. Once formed, the M_j are s by s matrices, where there are s roots of F counting multiplicity. For many interesting polynomials s is much less than the Bezout bound. The cost of all subsequent steps in the procedure is polynomial in s (floating-point arithmetic is used). Use of sparse matrix technology can lower the complexity still further.

It is this relatively low computational complexity for the subsequent steps that makes this process attractive in the exact coefficient case as well; once the multiplication matrices M_j have been formed, finding the roots has polynomial cost.

Numerical stability of the formation of the M_j is an issue, which we addressed in [5] by using the SVD.

4.2 Generic combinations

The purpose of step 2 is to avoid spurious multiplicities. Without this step, it is possible (indeed probable) that simple roots such as $(x, y) = (1, -1)$ and $(x, y) = (1, 1)$ might be seen as multiple roots if we worked with the multiplication matrix for x. Taking a generic combination of the M_j is equivalent to choosing a random hyperplane to project the variety onto, and the genericity ensures, with probability 1, that only truly multiple roots appear as multiple eigenvalues of M. Using a convex combination with positive random numbers ensures that numerical instability can be expected to be minimal.

4.3 Schur factorization

The purpose of step 3 is the computation of the matrix U, so that we may use it to compute the roots of F. We may use any good implementation of the Schur factorization, such as is found in MATLAB or in LAPACK [1]. The LAPACK routine has several advantages, including speed, portability, and the fact that routines for re-ordering the eigenvalues, which we will need in step 5, are pre-programmed.

4.4 Clustering of Roots

The results of this paper reduce the multivariate problem to a univariate problem. A key step in the solution of the univariate problem is the identification of root clusters, because in the inexact data case and the floating-point arithmetic case exact multiplicity is of course destroyed. Several very good heuristics have been published in the literature, and we give the details of one such procedure below, which we take from [9]. We have chosen to use these heuristics because they make use of the condition number of the root clusters, which we also need in the final step of the procedure of this paper. Other approaches exist for identifying root clusters which are equally good or even better, and we point out in particular the methods of [7], which are more than mere heuristics (although the difficult issues raised by nearby clusters and inexact tolerances make their methods slightly less than a complete algorithm, which to our knowledge does not exist for this problem).

The difficulty with clustering is that it is a nonlinear process, and nearby clusters can interfere strongly with one another even with very small perturbations of the initial data. The best approaches must take the problem context into consideration, and if human judgement is available one would expect superior performance over a wholly automated method. The heuristic we describe below uses condition number estimates and nearness, but only makes indirect use of symmetry. We believe that the use of the natural symmetry of a perturbed multiple root (for small perturbations one expects regularly spaced simple roots clustered in a circle) would be the simplest further step to take to improve this heuristic.

If the clustering heuristics succeed in finding clusters each well separated from the others, then the condition number of each cluster will be small, and the overall accuracy of the method of this paper will be good. If the clusters are not well separated, as may happen with a collection of smeared clusters, the best we can do (automatically) is to treat the smeared clusters as one big cluster. Human judgement may well improve the results, however.

The heuristics are as follows.

1. Let M be the matrix whose eigenvalues we want to cluster.

2. Take each computed eigenvalue λ_i and compute its individual (unclustered) condition number $||P_i||$ (see the following section).

3. Round λ_i to λ_i' where all decimal places of λ_i' smaller than the error estimate $\varepsilon||P_i|| \cdot ||M||$ are set to zero. [Here ε is the tolerance for the uncertainty in the input data, or in the floating-point arithmetic.]

4. Put those λ_i' that equal one another in their nonzero digits in the same cluster.

The arithmetic mean of the λ_i in a cluster will be a good estimate of the actual multiple root (this is the indirect use of symmetry). This observation seems to be originally due to

Wilkinson. It has been confirmed by use in many contexts. For example, in [8] it is shown that the arithmetic mean of a cluster converges quadratically to a multiple root of a polynomial, when Weierstrass iteration is used.

4.4.1 Eigenvalue Condition Numbers

The computation of the condition number of an eigenvalue or an eigenvalue cluster is crucial to understanding the reliability of the results computed by the procedure of this paper. We follow the treatment of [9], and we note that using the LAPACK routines for the Schur factorization of M makes these condition number estimates easily available.

Given the block Schur decomposition of M as

$$M = \begin{bmatrix} M_{11} & M_{12} \\ 0 & M_{22} \end{bmatrix} \tag{5}$$

where the eigenvalues of the k by k matrix M_{11} are precisely those we are interested in, we want to estimate the perturbation in the average of the eigenvalues of the cluster, $\overline{\lambda} = \mathrm{trace}\,(M_{11})/k$. We define the *spectral projector*

$$P = \begin{bmatrix} I_k & R \\ 0 & 0 \end{bmatrix}, \tag{6}$$

where R satisfies the system of linear equations

$$M_{11}R - RM_{22} = M_{12}\,. \tag{7}$$

Note that this equation is ill-conditioned precisely when the eigenvalues of M_{11} are close to those of M_{22}. Thus $||P|| = (1 + ||R||^2)^{1/2}$ (all norms are the 2-norm). Let E be a perturbation of M and $\varepsilon = ||E||$. Then if $\overline{\lambda}'$ is the average of the perturbed eigenvalues we have

$$|\overline{\lambda} - \overline{\lambda}'| \le \varepsilon ||P|| + O(\varepsilon^2)\,. \tag{8}$$

This shows that $||P||$ is a condition number for an eigenvalue cluster. The adaptation of this analysis to the case where the cluster is positioned more generally in the matrix is straightforward.

Computing $||P||$ is expensive and a cheaper overestimate can be obtained by using $||P||' = (1 + ||R||_F^2)^{1/2}$, where $||R||_F$ is the Frobenius norm of R.

Note that in the case $k = 1$ we may compute $||P||$ more simply as $||P|| = 1/|y^H x|$ where y and x respectively are unit normal left and right eigenvectors of M associated with the eigenvalue λ.

4.5 Reordering

We will explain in the next section why re-ordering is necessary. The immediate goal of the re-ordering is to ensure that roots in clusters appear together on the diagonal of T.

Re-ordering eigenvalues in LAPACK is possible using the built-in routine xTRESC. We consider here the use of plane rotations to achieve the same purpose in another environment (for example MATLAB), where only an un-ordered Schur factorization is available. The usual implementation of the Schur factorization uses splitting and deflation techniques which may easily cause approximately equal eigenvalues to appear widely separated from each other on the diagonal of T.

The basic operation used to re-order diagonal entries is the use of plane rotations to swap adjacent diagonal entries. Plane rotations are unitary and hence no further numerical instability is introduced.

We may write a plane rotation matrix as (see for example [3])

$$R = R(i,j,c,s) = I + (c-1)e_ie_i^T - \overline{s}e_ie_j^T + se_je_i^T + (\overline{c}-1)e_je_j^T\,. \tag{9}$$

If we wish to interchange $a = t_{ii}$ and $e = t_{jj}$, where $j = i+1$, we may do so by choosing c and s as follows. Recall that a, e, and $b = t_{ij}$ may be complex. A short calculation shows that putting

$$r = \left(|a-e|^2 + |b|^2\right)^{1/2} \tag{10}$$

and choosing (if $|b| > 0$)

$$c = \frac{|b|}{r} \tag{11}$$

$$s = \frac{(a-e)}{r}\frac{\overline{b}}{|b|} \tag{12}$$

will ensure that RTR^* is upper triangular with t_{ii} and its adjacent element t_{jj} interchanged. If $b = 0$, then the simple choice R equal to the i-j permutation matrix will work.

We cannot directly swap eigenvalues t_{ii} and t_{jj} which are not adjacent because the plane rotation matrix R affects the entries in the edges of the i-j lower triangle. However, it is clear that one may 'sweep' a diagonal element t_{ii} from its original location to any other diagonal location by successive adjacency swaps.

4.6 Block Upper Triangularization of M_j

Because of inexactness in the coefficients of F (not to mention further errors introduced by working with floats) the matrices M_j will not truly commute; instead, they will form what we call here a *nearly-commuting family*. We must investigate the effects of this near-commutation.

Proposition 7. If we define $E_{ij} = M_iM_j - M_jM_i$, and suppose that the 2-norm $||E_{ij}|| = \varepsilon_{ij} \le \varepsilon$, then if

$$M = \sum_{k=1}^{n} \alpha_k M_k = U^*TU\,, \tag{13}$$

where $\alpha_j > 0$ and $\sum \alpha_j = 1$, and defining $M_j' = UM_jU^*$, we have

$$||TM_j' - M_j'T|| \le \varepsilon\,. \tag{14}$$

In other words, the upper triangular matrix from the Schur factorization of the generic combination of the nearly-commuting family M_j nearly commutes, with the same ε, with the M_j'.

Proof. From (13) it follows that

$$\left(\sum_k \alpha_k M_k\right) M_j - M_j \left(\sum_k \alpha_k M_k\right) = \sum_k \alpha_k E_{kj} = E_j\,, \tag{15}$$

say, and since the random numbers α_k are chosen so that $\alpha_k > 0$ and $\sum \alpha_k = 1$, we have that $||E_j|| \le \varepsilon$. Rearranging (15) and using (13) we have

$$T\,(UM_jU^*) - (UM_jU^*)\,T = UE_jU^*\,. \tag{16}$$

Finally, we have that $||UE_jU^*|| \leq \varepsilon$ because U is unitary. ♮

Proposition 8. If the eigenvalues of clustered blocks are each well separated from the eigenvalues of the other blocks, with eigenvalue differences being $O(1)$, then the matrices $M_j' = UM_jU^*$ are nearly block upper triangular, with entries below the blocks being of size $O(\varepsilon)$.

Proof. We consider the 3 by 3 case, from which the generalization to n by n is immediate. We show that near-commutation with an upper triangular matrix T, as in equation (16), implies near block upper triangularity. Consider $E =$

$$\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ & t_{22} & t_{23} \\ & & t_{33} \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} - \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ & t_{22} & t_{23} \\ & & t_{33} \end{bmatrix} .$$

Denote the entries of E by $\varepsilon\sigma_{ij}$. Then we have the following equations, written explicitly:

$$(t_{33} - t_{11})m_{31} = \varepsilon\sigma_{31} \tag{17}$$
$$(t_{22} - t_{11})m_{21} + t_{23}m_{31} = \varepsilon\sigma_{21} \tag{18}$$
$$(t_{33} - t_{22})m_{32} - t_{12}m_{31} = \varepsilon\sigma_{32} \, . \tag{19}$$

It is clear that m_{31} will be $O(\varepsilon)$ if $t_{33} - t_{11}$ is $O(1)$. If this is true, then m_{32} will be $O(\varepsilon)$ if $t_{33} - t_{22}$ is $O(1)$, and likewise m_{21} will be $O(\varepsilon)$ if $t_{22} - t_{11}$ is $O(1)$.

If however t_{33} and t_{11} are not widely separated, then the entire lower block will be large, even if t_{22} is quite different from t_{11} and t_{33}. This points out why it is necessary to re-order the eigenvalues of T.

The generalization of this calculation to the n by n case is straightforward. For larger blocks, the effect of the non-diagonal elements of T on the conditioning of the (triangular) linear system defining the m_{ij} may become an issue, but we would not expect this to be serious unless the blocksize is very large. In any event the principal difficulty can be expected to come from small entries on the diagonal, which happens if and only if the eigenvalue clusters are not well separated. ♮

4.7 *A Posteriori* error analysis

We may compute the condition number of each block of eigenvalues. Once this has been done, we may use the size of the entries below the blocks of M_j' together with the condition number to estimate the errors in the values of the roots for x_i, using equation (8). If any of these errors are large, this might indicate that the clustering heuristic failed. If they are all small, this gives us good error estimates for the roots of the system.

5 Examples

5.1 A symmetric problem

We begin with a symmetric matrix example from [3]. They used this example to show that the DODO method (Diagonalize One then Diagonalize the Other) approach to simultaneous diagonalization does not work. We show here that the use of generic combinations plus unitary re-ordering essentially makes the DODO method work after all.

Their example runs as follows. Suppose

$$A = \begin{bmatrix} I_\varepsilon & 0 \\ 0 & J \end{bmatrix} \tag{20}$$
$$B = \begin{bmatrix} J & 0 \\ 0 & I_\varepsilon \end{bmatrix} \tag{21}$$

where

$$I_\varepsilon = \begin{bmatrix} 1-\varepsilon & 0 \\ 0 & 1+\varepsilon \end{bmatrix} \tag{22}$$
$$J = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} . \tag{23}$$

Then we have that $E = AB - BA$ is $O(\varepsilon)$, because

$$I_\varepsilon J - JI_\varepsilon = \begin{bmatrix} 0 & -2\varepsilon \\ 2\varepsilon & 0 \end{bmatrix} . \tag{24}$$

They point out that if

$$C = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{25}$$

then diagonalizing the matrix A would gives the matrix

$$U = \begin{bmatrix} I & 0 \\ 0 & C \end{bmatrix} , \tag{26}$$

but U^*BU contains off-diagonal elements of size $O(1)$. Similarly, interchanging I and C gives a matrix which diagonalizes B but does not nearly diagonalize A. The naive DODO strategy does not work, as pointed out in [3].

They also note that the matrix Q with C in place of I above in U (*i.e.* $Q = \text{subs}(I = C, U)$) makes the off-diagonal elements of Q^*AQ and Q^*BQ simultaneously $O(\varepsilon)$. The question is how to compute Q in a reasonable way.

Let $\varepsilon = 10^{-4}$ in what follows. If we form $M = \alpha A + (1 - \alpha)B$, for some random $0 < \alpha < 1$, say $\alpha = 0.2190$, then MATLAB computes the Schur factorization of M as $M = Q^*TQ$ with

$$Q = \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ -0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & -0.7072 & 0.7070 \\ 0 & 0 & 0.7070 & 0.7072 \end{bmatrix} \tag{27}$$

and $T = \text{diag}(-0.5621, 1.000, 0.5621, 1.000)$. Any conceivable clustering heuristic would identify the double root at 1. Notice that though A and B each have nearly *triple* roots at 1, the pair of matrices A and B have only a double root. This is because the eigenvector corresponding to -1 in A is an eigenvector corresponding to 1 in B, and vice-versa; here the generic combination of A and B automatically sorted out this spurious multiplicity.

Once the root cluster has been identified, we can (for example) switch the middle two elements of the diagonal of T to put the double roots together. This changes Q to

$$\hat{Q} = \begin{bmatrix} 0.7071 & 0 & -0.7071 & 0 \\ -0.7071 & 0 & -0.7071 & 0 \\ 0 & -0.7072 & 0 & 0.7070 \\ 0 & 0.7070 & 0 & 0.7072 \end{bmatrix} , \tag{28}$$

and T to diag $(-0.5621, 05621, 1.000, 1.000)$. The fact that this is different from the Q in [3] is a simple consequence of the fact that any ordering that put the double roots together would work, and produce a different Q. It is a simple matter to see that $\hat{Q}^* A \hat{Q}$ and $\hat{Q}^* B \hat{Q}$ both have off-diagonal elements only of $O(\varepsilon)$. To 10 decimal places (remember $\varepsilon = 10^{-4}$ here) we get that the roots are $(x, y) = (1, -1)$, $(-1, 1)$, and a double root at $(1, 1)$. Averaging in the double root cluster is not, in fact, necessary in this example, and the roots happen to be computed very well without it. This accidental accuracy is an artifact of the symmetry of the perturbation of the original commutative pair of matrices.

5.2 An intersection problem

This is Example 3 from [4]. Consider $F = \{x^2 + y^2 - 1, x^3 + (2 + z)xy + y^3 - 1, z^2 - 2\}$. Using the exact arithmetic approach of [4], we first compute a Gröbner basis, and a basis for the normal set N. The basis for N that we get is

$$[1, z, y, x, yz, y^2, xz, xy, y^2 z, y^3, xyz, y^3 z]^T$$

so there are 12 roots. There are 4 simple roots and 4 double roots. The double roots corresponding to $x = 0$ and $y = 1$ each have two eigenvectors, meaning that in the eigenvector approach of [4] we would have to form a linear combination to find the common eigenvector for m_x, m_y, and m_z. For this simple example this can be done by inspection, but the procedure of this paper is automatic and needs no human intervention.

We form a generic combination of all three multiplication matrices, find the Schur factorization of the generic combination, re-order the Schur factorization (the $(0, 1)$ double roots do not appear together though the double roots at $\pm 1/\sqrt{2}$ accidentally do so already), and compute the 3 block upper triangular matrices to read off the roots.

We find that in the case of the double $(0, 1)$ and $(1, 0)$ roots, averaging is again not necessary, but we find that for the double roots at $-1/\sqrt{2}$, averaging improves the computed roots from $-0.70710678118655 + 2.061398i \cdot 10^{-8}$ and $-0.70710678118655 - 2.061398i \cdot 10^{-8}$ to -0.70710678118655, which is correct (rounded) to all places. For some reason, the positive roots were not perturbed to complex roots but rather to 0.70710673753969 and 0.70710682483344, which when averaged give $1/\sqrt{2}$ correct up to 3 units in the last place.

The condition numbers of these roots individually are about 1.E8, so the clustering heuristic of [9] would correctly identify these clusters. The condition numbers of the clusters are about 10, giving confirmation that valid clusters have been found.

These calculations were done in MATLAB in standard floating-point arithmetic. The multiple roots are obtained to nearly complete accuracy. Of course they are just double roots, and all the double roots are widely separated.

5.3 A harder problem

We consider the intersection of the following degree 7 and 8 curves, which we take from [9]. Consider the curves $F(x, y) = 0$ and $F_y(x, y) = 0$ where $F(x, y) =$

$$\begin{aligned} & -2 - 7x + 14x^3 - 7x^5 + x^7 + (7 - 42x^2 + 35x^4 - 7x^6)y \\ + \quad & (16 + 42x - 70x^3 + 21x^5)y^2 + (-14 + 70x^2 - 35x^4)y^3 \\ + \quad & (-20 - 35x + 35x^3)y^4 + (7 - 21x^2)y^5 + (8 + 7x)y^6 \\ - \quad & y^7 - y^8 \,. \end{aligned}$$

A graph of these curves may be found in [9]. There are many high-order intersections.

The code discussed in [4] sets up the sparse 49 by 49 multiplication matrices M_x and M_y for us. We can import these matrices into MATLAB, form a generic combination, and find the Schur factorization. We sort the eigenvalues by magnitude, and sweep the upper triangular factor with adjacency swaps to put roots with similar magnitudes together (this is a crude clustering heuristic). We can then use the (dense) 49 by 49 orthogonal matrix U to find $M_x' = U^* M_x U$ and $M_y' = U^* M_y U$, which turn out to be nearly block upper triangular, with sub-block entries of size about 1.E$-$16. There are 21 double roots (making 42 roots) and 7 simple roots. All are real. We exhibit the raw, un-averaged roots in Table 1, from which we can see the spurious imaginary parts which are removed by averaging. We have left these errors in the table so the potential effect of averaging is more evident.

6 Efficiency

It is clear that sparse matrix technology is important for this process. Even for the last example, which after all only creates 49 by 49 matrices, sparsity is very important—only 617 out of $2 \cdot 49^2 = 4802$ entries, or less than 14%, are nonzero. Sparse methods become even more important as the size of the problem increases. MATLAB does not have a sparse Schur factorization, so for our experiments sparsity really only helped in the construction of the M_j and the transfer from Maple to MATLAB. For larger problems, with a full implementation, we would want to take complete advantage of sparsity, even for the construction and storage of U as a product of sparse factors, and especially for the reordering by plane rotations.

Manocha and co-workers (see for example [10]) already take advantage of sparsity in their resultant/matrix-eigenproblem approach; to be more competitive with their methods (which are appropriate for special classes of problems for which good resultant formulations are available) it is clear that the general approach of this paper must also take advantage of sparsity. Otherwise, the two approaches are clearly similar in spirit, although the details of which eigenvalue problem is solved and what use is made of the solution are quite different.

7 Concluding Remarks

For the simple example problems shown in this paper, we did not need the *a posteriori* condition number calculations. For larger, non-artificial, problems, the condition number analysis becomes crucial. We have only an experimental implementation of these ideas at present—all of our experiments were carried out in Maple and MATLAB. A full implementation in ALDOR[2] using LAPACK is currently under construction.

This paper presents what we believe are useful ideas for the numerically stable solution of zero-dimensional polynomial systems. We use unitary matrices throughout, for stability, and we have shown in Propositions 7 and 8 that the

[2] This language was known as $A^\sharp$ during its development.

Table 1: Raw, un-averaged roots computed by reordered Schur factorization of $\alpha M_x + (1-\alpha)M_y$.

x	y
-3.91298142177011	-1.95065207736178
3.65578305461127	1.80399138041296
3.64971230842687	1.84773551443691
3.64968136262356	1.84778172208383
-3.23983918526081	-1.56367727284582
$-3.21615129092702 + 0.00002328073289i$	$-1.41421357039218 + 0.00004580718179i$
$-3.21615134768276 - 0.00002328073288i$	$-1.41421275808871 - 0.00004580718179i$
$-3.09473861915948 + 0.00004696592621i$	$-1.84775972503633 - 0.00001055013453i$
$-3.09473866446841 - 0.00004696592621i$	$-1.84775907656834 + 0.00001055013453i$
2.68379040110613	1.23369350526449
2.66119794749733	1.41420555211701
2.66118838416448	1.41422157723975
$2.56730460050631 - 0.00000262710713i$	$0.76536686489214 - 0.00000369012655i$
$2.56730460064228 + 0.00000262710713i$	$0.76536686294650 + 0.00000369012655i$
$2.29281203202697 + 0.00000000000000i$	$1.84775758053412 + 0.00000000000000i$
2.29278983415834	1.84776054924562
-2.01311774220408	-0.81210271285545
$-2.01234647144427 + 0.00000407516224i$	$-0.76536672795258 + 0.00006304524318i$
$-2.01234647043344 - 0.00000407516224i$	$-0.76536674241949 - 0.00006304524318i$
$-1.85925542871619 + 0.00001251199393i$	$-1.41421358148609 - 0.00000843752701i$
$-1.85925542983032 - 0.00001251199393i$	$-1.41421356554069 + 0.00000843752701i$
$-1.80193773581455 + 0.00000057641877i$	$0.00000000018690 + 0.00000076436374i$
$-1.80193773578331 - 0.00000057641877i$	$-0.00000000025939 - 0.00000076436374i$
$-1.40271719700009 + 0.00000461689146i$	$-1.84775906552626 - 0.00000061746720i$
$-1.40271719703012 - 0.00000461689146i$	$-1.84775906509630 + 0.00000061746720i$
$1.26105608032033 + 0.00000000000000i$	$0.26535936946091 + 0.00000000000000i$
$1.21040954733513 - 0.00000000000000i$	$0.76536507366567 + 0.00000000000000i$
$1.21040792098700 - 0.00000000000000i$	$0.76536861240570 - 0.00000000000000i$
$1.24698010569211 + 0.00000000000000i$	$0.00000254491492 - 0.00000000000000i$
$1.24697909871740 + 0.00000000000000i$	$-0.00000250143549 - 0.00000000000000i$
$0.96917235789609 - 0.00000000000000i$	$1.41421311503626 - 0.00000000000000i$
$0.96917103092287 + 0.00000000000000i$	$1.41421401123522 - 0.00000000000000i$
$1.03657087115355 + 0.00000062855223i$	$-0.76536686585439 + 0.00000088288645i$
$1.03657087110391 - 0.00000062855223i$	$-0.76536686514304 - 0.00000088288645i$
0.60078046530288	1.84775883947923
0.60077845731226	1.84775929055378
$-0.44504197994469 + 0.00000000000000i$	$0.00000046374683 - 0.00000000000000i$
$-0.44504175682568 + 0.00000000000000i$	$-0.00000045023939 + 0.00000000000000i$
$-0.48377825205275 + 0.00000000000000i$	$0.63069242102367 + 0.00000000000000i$
$-0.48161273894260 - 0.00000044375307i$	$0.76536686443195 - 0.00000686515593i$
$-0.48161273859811 + 0.00000044375307i$	$0.76536685950153 + 0.00000686515593i$
$-0.32032499719021 - 0.00000092400294i$	$-0.76536685940900 + 0.00000203324221i$
$-0.32032499679213 + 0.00000092400293i$	$-0.76536686510625 - 0.00000203324215i$
$-0.38772457722093 + 0.00000000000000i$	$1.41421272485396 - 0.00000000000002i$
$-0.38772359406993 + 0.00000000000001i$	$1.41421188705233 - 0.00000000000014i$
$-0.16723413397514 + 0.00000149735867i$	$-1.41421105316361 - 0.00000250913831i$
$-0.16723395857522 - 0.00000149735868i$	$-1.41421356353513 + 0.00000250913843i$
$-0.04582132955002 + 0.00000086889636i$	$-1.84775906020616 - 0.00000127966609i$
$-0.04582132921678 - 0.00000086889636i$	$-1.84775906497831 + 0.00000127966608i$

size of the elements below the diagonal blocks in the M'_j is $O(\varepsilon)$, where ε measures the size of the lack of commutativity of the M_j (which may be due to inexactness or due to floating-point arithmetic), provided that the eigenvalue clusters are well-separated. This gives, from equation (8), a good *a posteriori* error estimate for the roots of F.

A key remaining goal is to understand the relative conditioning of the original multivariate problem versus the conditioning of the univariate problem we ultimately solve. While we claim the procedure for reducing the multivariate problem to a univariate one is stable, it is not yet known whether the resulting univariate problem is as well-conditioned as the original.

That said, the preliminary experiments reported in this paper show quite good stability and accuracy in the presence of multiple roots. The key to achieving these good results lies in the use of a generic combination to eliminate spurious multiple roots (at the cost of some loss of sparsity, which we do not feel is significant), and re-ordering the Schur factorization so we can easily get at each root cluster.

References

[1] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorenson, D. *Lapack User's Guide*. SIAM, 1995.

[2] Auzinger, W., and Stetter, H. J. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Numerical Mathematics* (1988), R. P. Agarwal, Y. M. Chow, and S. J. Wilson, Eds., vol. 86 of *ISNM*, Birkhäuser, pp. 11–30.

[3] Bunse-Gerstner, A., Byers, R., and Mehrmann, V. Numerical methods for simultaneous diagonalization. *SIAM Journal of Matrix Analysis and Applications 14*, 4 (1993), 927–949.

[4] Corless, R. M. Gröbner bases and matrix eigenproblems. *Sigsam Bulletin: Communications in Computer Algebra 30*, 4 (December 1996), 26–32.

[5] Corless, R. M., Gianni, P. M., Trager, B. M., and Watt, S. M. The Singular Value Decomposition for polynomial systems. In *Proceedings ISSAC Montréal* (1995), A. H. M. Levelt, Ed., pp. 195–207.

[6] Faugère, J. C., Gianni, P., Lazard, D., and Mora, T. Efficient computation of zero-dimensional gröbner bases by change of ordering. *J. Symbolic Computation 16* (1993), 329–344.

[7] Hribernig, V., and Stetter, H. J. Detection and validation of clusters of polynomial zeros. *Journal of Symbolic Computation to appear* (1997).

[8] Hull, T. E., and Mathon, R. The mathematical basis and a prototype implementation of a new polynomial rootfinder with quadratic convergence. *ACM Transactions on Mathematical Software 22*, 3 (September 1996), 261–280.

[9] Manocha, D., and Demmel, J. W. Algorithms for intersecting parametric and algebraic curves II; multiple intersections. *Computer Vision, Graphics and Image Processing: Graphical Models and Image Processing 57*, 2 (1995), 81–100.

[10] Manocha, D., and Krishnan, S. Solving algebraic systems using matrix computations. *Sigsam Bulletin: Communications in Computer Algebra 30*, 4 (December 1996), 4–21.

[11] Möller, H. M., and Stetter, H. J. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numerische Mathematik 70*, 3 (1995), 311–330.

[12] Stetter, H. J. Analysis of zero clusters in multivariate polynomial systems. In *Proceedings ISSAC Zürich* (1996), pp. 127–136.

[13] Stetter, H. J. Matrix eigenproblems are at the heart of polynomial system solving. *Sigsam Bulletin: Communications in Computer Algebra 30*, 4 (December 1996), 22–25.