Solving Systems of Non-Linear Polynomial Equations Faster

John F. Canny
Computer Sci. Div., EECS Dept., University of California
Berkeley, CA 94720; canny@ernie.berkeley.edu

Erich Kaltofen and Lakshman Yagati
Dept. Computer Sci., Rensselaer Polytechnic Institute
Troy, NY 12180-3590; kaltofen@cs.rpi.edu, yagatil@cs.rpi.edu

1. Introduction

Finding the solution to a system of \( n \) non-linear polynomial equations in \( n \) unknowns over a given field, say the algebraic closure of the coefficient field, is a classical and fundamental problem in computational algebra. For algebraic reasons (refer to footnote 1 in van der Waerden (1953, §80)) one considers projective problems, that is, the polynomials are homogeneous and the solutions are sought in \( n \)-dimensional projective space. Note also that the solutions of an affine system are specializations of the solution rays of its homogenized projective version. Going back to Cayley and Bezout in the last century, solvability of such a projective system is determined by the vanishing of a certain invariant, its resultant. This invariant generalizes the Sylvester resultant of two polynomials in a single variable (Knuth 1981) and the determinant of the coefficient matrix on a homogeneous linear system. In 1916 Macaulay (1916) showed that the resultant can be expressed by a quotient of two determinants whose corresponding matrices have coefficients of the input polynomials as their entries. These matrices have dimension exponential in the number of variables, but since there is an easy reduction to an \( \mathcal{NP} \)-complete problem (Agnarsson et al 1984), there is little hope for a polynomial-time solution in the number of variables. Finally, if a projective system of \( n - 1 \) equations and \( n \) unknowns has finitely many solutions, these can again be found by computing the resultant of the system with the addition of a generic linear form. That resultant, the so called u-resultant, is a polynomial in the generic coefficient variables of the added form, and it factors into linear factors whose scalar coefficients are exactly the components in the corresponding solution rays (refer also to the example below). The results discussed so far are classical; for modern extensions of these to deal with infinitely many solutions at infinity, for instance, refer to (Lazard 1981) and (Canny 1988b).

The main result of this article is a new efficient algorithm to evaluate the resultant. The dimensions of Macaulay’s matrices are bounded by \( D \), where

\[
D = \binom{d + n - 1}{n - 1}, \quad d = 1 + \sum_{i=1}^{n} (d_i - 1),
\]

\( d_i \) the degree of the \( i \)-th projective equation. We present an algorithm that computes the resultant in

\[
O(nD^2 (\log^2(D) \log(\log D) + n))
\]
arithmetic steps over the coefficient field, using $O(D)$ locations for field elements. The best previous methods, described in present day research by (Lazard 1981), (Grigoryev and Chistov 1984), (Canny 1988c), and (Renagar 1987b), for instance, all required to compute the Macaulay determinants by Gaussian elimination or the derived algorithms using fast matrix multiplication. Hence our result improves the time complexity from $O(D^\omega)$, where $\omega$ is the matrix multiplication exponent, to essentially $D^{2+o(1)}$, with $n = D^{o(1)}$ for $d = \Omega(n)$, and even more importantly, we have improved the space requirements from $O(D^2)$ to $O(D)$.

Having a fast resultant evaluation procedure, one can find solutions of a non-singular system quickly. Here non-singular means that the system only has finitely many solutions. One needs to factor the u-resultant of the input system. Our algorithm provides an efficient method to evaluate the u-resultant at a specialization for the generic variables. Fortunately, this is all one needs in order to apply Canny’s primitive element method (Canny 1988a), or the more general factorization method for polynomials given by black boxes for their evaluation (Kaltofen and Trager 1988). Both approaches essentially take $O(N^2)$ arithmetic operations, where $N = \prod_{i=1}^{t} d_i$ is the number of solutions.

There are two alternate ways of computing solutions to polynomial systems, the classical elimination method due to Kronecker (van der Waerden 1953) and the modern Gröbner basis method due to Buchberger (see the survey (Buchberger 1985)). From a theoretical point of view, the complexity bound for the first method is doubly-exponential in $n$. The Gröbner basis algorithm for 0-dimentional ideals has complexity $n^3 \max \{d_i\}^{O(n^3)}$ (Caniglia et al 1988). Moreover, the initial reductions in the Gröbner basis algorithm are identical to the initial Gaussian row elimination steps on the Macaulay matrix. An S-polynomial construction in the Gröbner basis algorithm corresponds to several row reductions in Gaussian elimination. In one variable this makes computation of Sylvester resultants by the Euclidean algorithm quadratic time vs. the cubic time algorithm for triangularizing the Sylvester matrix. However, this phenomenon seems difficult to generalize, at least in a straight-forward fashion, to multivariate resultants and the Gröbner basis algorithm. In fact, the main problem with performing Gaussian elimination on this usually sparse matrix is the fill-in to quadratic size. This is especially costly since this matrix has dimension exponential in $n$.

Our new resultant algorithm is based on two recent results in computational algebra. For one we make use of Wiedemann's (1986) fast method for computing the determinant of a matrix using a linear number of matrix times vector operations. In the case of Macaulay's matrix, the matrix times vector product can be shown to be equivalent to computing a multivariate polynomial product in which the product is a dense polynomial bounded in total degree. In order to compute this product in linear time in the number of terms in the answer, we make use of the new sparse interpolation algorithms (Ben-Or and Tiwari 1988), (Zippe1 1990), and (Kaltofen and Lakshman 1988). In this particular setting, the term-structure of the answer polynomial is known and one only needs to perform the last step of the Ben-Or&Tiwari algorithm. We can show that both the pointwise evaluation and interpolation problems, which correspond to transposed Vandermonde systems, can be solved in the same asymptotic time regular Vandermonde systems are solvable.

We wish to point out that our algorithm is an exact method. There are numerical methods based on homotopical transformation of solution paths (see, e.g., (Drexler 1977), (Garcia and Zangwill 1979), (Li et al. 1988), and (Zulehner 1988)), and on Newton iteration (Renagar 1987a). These methods are, however, not universally applicable.

This paper first introduces some notation for
the Macaulay resultant matrices. Then we provide the fast total degree bounded multivariate polynomial product algorithm. Finally, we show how that result can be combined with Wiedemann’s determinant algorithm to give our fast and space efficient resultant method.

2. The Multivariate Resultant

We now give a brief description of the multivariate resultant of a system of polynomial equations. The interested reader can consult (Macaulay 1916) and (Canny 1988c) for further details. Given \( n \) homogeneous forms \( f_1, \ldots, f_n \) in the variables \( x_1, \ldots, x_n \), their resultant is defined as the ratio of the determinant of a certain matrix \( M \) (whose construction is described below) and the determinant of a particular submatrix \( \Delta \) of \( M \). The rows of \( M \) are indexed by the monomials in \( x_1, \ldots, x_n \) of degree \( d = 1 + \sum_{i=1}^{n} d_i - 1 \), where \( d_i \) is the degree of the polynomial \( f_i \). Therefore, \( M \) has \( D = \binom{d+n-1}{n-1} \) rows.

A polynomial is said to be reduced in the variables \( x_1, x_2, \ldots, x_k \) for \( 1 \leq i_1, i_2, \ldots, i_k \leq n \) iff for all \( j, 1 \leq j \leq k \), its degree in \( x_{i_j} < d_{i_j} \). A polynomial is said to be just reduced if it is reduced in any \( n-1 \) of the \( n \) variables \( x_1, \ldots, x_n \).

Consider the homogeneous form

\[
F = f_1 g_1 + f_2 g_2 + \ldots + f_n g_n
\]

where \( \deg(g_i) = d - d_i \) and \( g_i \) is a generic polynomial in \( x_1, \ldots, x_n \) (i.e., coefficients are not specialized) reduced in \( x_1, \ldots, x_{i-1} \). The columns of \( M \) are labelled by the monomials of \( g_i \) and the rows are labelled by monomials in \( x_1, \ldots, x_n \) of degree \( d \). The entries in the column labelled by a particular monomial \( x_1^{a_1} \ldots x_n^{a_n} \) of \( g_i \) are the coefficients of \( f_i \).

The submatrix \( \Delta \) is obtained by deleting the rows in \( M \) whose labels are reduced (in any \( n-1 \) variables) and the columns containing the coefficients of \( x_i^{d_i} \) in \( f_i \) in the deleted rows. Thus, \( \Delta \) has \( D - D' \) rows and columns where \( D' = \sum_j \prod_{i \neq j} d_i \). The resultant is given by

\[
R = \det(M) / \det(\Delta), \quad \text{provided } \det(\Delta) \neq 0.
\]

Otherwise one chooses a different ordering of the polynomials, say \( f_2, \ldots, f_n, f_1 \). If for all such orderings the determinants of the corresponding \( \Delta \)'s are zero, \( R \) is defined to be zero.

The fundamental property of the resultant is that the \( f_i \) have common zeros if and only if \( R = 0 \).

The common zeros of \( n \) non-homogeneous polynomials \( f_1, \ldots, f_n \) in \( n \) variables \( x_1, \ldots, x_n \) can be recovered by homogenizing the \( f_i \) by the addition of a homogenizing variable \( x_{n+1} \) and introducing a new form \( f_{n+1} = u_1 x_1 + u_2 x_2 + \ldots + u_{n+1} x_{n+1} \) where the \( u_i \) are indeterminates.

The resultant of these \( n+1 \) forms is now a polynomial in the \( u_i \), the \( u \)-resultant of \( f_1, \ldots, f_n \).

Provided the homogeneous system has finitely many solution rays, this \( u \)-resultant factors into linear factors in \( u_1, \ldots, u_{n+1} \) over the algebraic closure of the coefficient field, and the coefficients of the \( u_i \) in each factor correspond to the components in the solution ray of the homogenized system.

In the case of two homogeneous polynomials in two variables or two inhomogeneous polynomials in a single variable, the resultant reduces to the familiar Sylvester resultant. In the case of \( n \) linear forms, the resultant reduces to the determinant of the coefficient matrix. To illustrate these concepts, we shall give a small example.

**Resultant Example** (Lazard 1981): Given is an affine system in two variables augmented by a generic linear form:

\[
\begin{align*}
    f_1 &= x^2 + xy + 2x + y - 1 = 0, \\
    f_2 &= x^2 + 3x - y^2 + 2y - 1 = 0, \\
    f_i &= ux + vy + w = 0.
\end{align*}
\]

Following is the matrix corresponding to the
u-resultant of (2), with z the homogenizing variable. The divisor $\det(\Delta)$ is in this case a non-zero rational. The labels at the rows and columns correspond to its construction. Notice that

$$\det(M) = (u-v+w)(-3u+v+w)(v+w)(u-v)$$

corresponding to the affine solutions $(1, -1)$, $(-3, 1)$, $(0, 1)$, and one solution at infinity.

3. Fast Polynomial Multiplication

In this section, we describe an efficient algorithm for computing the product of two total degree bounded multivariate polynomials. More precisely, we prove the following:

**Theorem 1.** Given two multivariate polynomials $f_1(x_1, \ldots, x_n)$ and $f_2(x_1, \ldots, x_n)$ over a field of characteristic zero and of total degrees $\delta_1$ and $\delta_2$, respectively, their product $g(x_1, \ldots, x_n)$ can be computed in $O(M(T) \log(T))$ arithmetic operations, where $M(T)$ denotes the numbers of arithmetic operations needed to multiply two univariate polynomials of degree $T$, and $T = \binom{\delta_1 + \delta_2 + n}{n}$, the total number of terms in the product $g$.

Notice that a multidimensional FFT-based multiplication algorithm performs $O(M(\delta^n))$ arithmetic operations in this case, where $\delta = \delta_1 + \delta_2$. Also, the best univariate polynomial multiplication algorithm over an arbitrary field has $M(T) = O(T \log(T) \log(\log T))$ complexity (Schönhage 1977). Our algorithm works by evaluation, pointwise scalar multiplication, and interpolation:

— $f_1$ and $f_2$ are evaluated at specially chosen integer points.

— The values of $g$ at these points are computed by multiplying the corresponding values of $f_1$ and $f_2$.

— $g$ is interpolated from its values at the special points.

We now describe the algorithm in detail.

3a. Evaluating a Multivariate Polynomial at Special Points

Let $f(x_1, \ldots, x_n) = a_1m_1 + a_2m_2 + \ldots + a_tm_t$ where the $m_i$ are distinct monomials and $a_i$ are constant coefficients. We want to evaluate $f$ at the points

$$(1, \ldots, 1), (p_1, \ldots, p_n), \ldots, (p_1^{-1}, \ldots, p_n^{-1})$$

where $p_i$ denote distinct primes. Let

$$v_i = (m_i)_{x_i = p_j, 1 \leq j \leq n}$$

and $b_i = f(p_1^i, \ldots, p_n^i)$. We want to compute the $b_i$ for $0 \leq i \leq t - 1$. Let

$$V = \begin{pmatrix}
1 & v_1 & v_1^2 & \ldots & v_1^{t-1} \\
1 & v_2 & v_2^2 & \ldots & v_2^{t-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & v_t & v_t^2 & \ldots & v_t^{t-1}
\end{pmatrix},$$

$$a = \begin{pmatrix}
a_1 \\
a_2 \\
\vdots \\
a_t
\end{pmatrix},
\quad b = \begin{pmatrix}
b_0 \\
b_1 \\
\vdots \\
b_{t-1}
\end{pmatrix}.$$
$V$ is a Vandermonde matrix. Clearly, $V^{\text{Tr}}a = b$. Rewrite this as

$$(V^{\text{Tr}}V)V^{-1}a = b. \quad (3)$$

Let $V^{-1}a = a'$. Solving a $(t \times t)$ Vandermonde system is equivalent to interpolating a univariate polynomial of degree $t - 1$ from its values at $t$ points. This can be performed in $O(M(t)\log(t))$ arithmetic operations (cf. (Aho et al 1974)). Formula (3) now becomes

$$(V^{\text{Tr}}V)a' = b$$

with

$$V^{\text{Tr}}V = \begin{pmatrix}
\sum v_i & \sum v_i^2 & \ldots & \sum v_i^{t-1} \\
\sum v_i^2 & \sum v_i^3 & \ldots & \sum v_i^{t-2} \\
\vdots & \vdots & \ddots & \vdots \\
\sum v_i^{t-1} & \sum v_i^t & \ldots & \sum v_i^{2(t-1)}
\end{pmatrix},$$

which is a Hankel matrix. The product of a $(t \times t)$ Hankel matrix and a vector can be computed in $O(M(t))$ arithmetic operations. It can be read off from the coefficients of the product of polynomials

$$f = n + \sum v_iz + \sum v_i^2z^2 + \ldots + \sum v_i^{2(t-1)}z^{2(t-1)}$$

and

$$g = a'_1z^{t-1} + a'_2z^{t-2} + \ldots + a'_t.$$ 

Therefore, $(V^{\text{Tr}}V)a'$ can be computed using at most $O(M(t)\log(t))$ arithmetic operations if all the entries of $V^{\text{Tr}}V$ can be computed in $O(M(t)\log(t))$ arithmetic operations. But the entries of $V^{\text{Tr}}V$ are the first $(t-1)$ power sums of the $v_i$. Now, Newton’s identities for computing the power sums $s_j = \sum v_i^j$ from the elementary symmetric functions $\sigma_j$ of $v_i$ lead to the Toeplitz system of equations $Ws = w$ where

$$W = \begin{pmatrix}
1 & 0 & \ldots & \ldots & 0 \\
-\sigma_1 & 1 & \ldots & \ldots & 0 \\
\sigma_2 & -\sigma_1 & \ldots & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
\sigma_t & -\sigma_{t-1} & \ldots & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & \ldots & (\sigma_t)^{t-1} & \ldots & 1
\end{pmatrix}$$

$$s = \begin{pmatrix}
s_1 \\
s_2 \\
\vdots \\
s_{t+1} \\
\vdots \\
s_{2t}
\end{pmatrix}, \quad w = \begin{pmatrix}
\sigma_1 \\
-2\sigma_2 \\
3\sigma_3 \\
\vdots \\
t\sigma_t \\
0
\end{pmatrix}.$$ 

A $(t \times t)$ Toeplitz system can be solved in $O(M(t)\log(t))$ arithmetic operations (Brent et al 1980). The elementary symmetric functions $\sigma_i$ can be read off from the coefficients of the polynomial $\prod_{i=1}(z - v_i)$ which can be computed in $O(M(t)\log(t))$ arithmetic operations (cf. (Aho et al 1974)). This method can easily be generalized to evaluate $f(x_1, \ldots, x_n)$ at points $(1, \ldots, 1), (p_1, \ldots, p_n), \ldots, (p_{T}, \ldots, p_{T})$ for $T \geq t$ in $O(M(T)\log(T))$ arithmetic operations. Another approach to the entire problem is to pre-multiply $V^{\text{tr}}a$ by a vector of indeterminates, and apply the Baur and Strassen (1983) all partial derivatives algorithm to the resulting single entry. However, for that solution it is not clear that linear space can be accomplished.

### 3b. Dense Interpolation

The final step of the polynomial multiplication algorithm is the interpolation step. We now describe a dense interpolation scheme. The algorithm needs as input the total degree $\delta$ of the polynomial to be interpolated and its values at special points. Let $g(x_1, \ldots, x_n) = a_1m_1 + a_2m_2 + \ldots + a_Tm_T$ be the polynomial to be interpolated. We have $m_i = x_1^{e_{i1}} \ldots x_n^{e_{in}}$ such that $\sum j e_{i,j} \leq \delta$; $a_i$ and $v_i$ are as before, and $T = \binom{\delta + n}{n}$ is the maximum possible number of terms in $g$.

Evaluate $g$ at points $(1, \ldots, 1), (p_1, \ldots, p_n), (p_1^2, \ldots, p_n^2), \ldots, (p_{T-1}, \ldots, p_{T-1})$. Let the respective values be denoted by $g_0, g_1, \ldots, g_{T-1}$. Clearly,

$$W = \begin{pmatrix}
1 & 1 & \ldots & 1 \\
v_1 & v_2 & \ldots & v_T \\
\vdots & \vdots & \ddots & \vdots \\
v_1^{T-1} & v_2^{T-1} & \ldots & v_T^{T-1}
\end{pmatrix} \begin{pmatrix}
a_1 \\
a_2 \\
\vdots \\
a_T
\end{pmatrix}.$$
This is a transposed Vandermonde system of equations and the \(a_i\) can be computed in \(O(M(t) \log(t))\) steps (Kaltofen and Lakshman 1988). It now follows that the multiplication algorithm performs \(O(M(T) \log(T))\) arithmetic operations in all as both the evaluation step and the interpolation step can be completed in \(O(M(T) \log(T))\) arithmetic operations. The pointwise multiplication step only needs \(O(T)\) arithmetic operations. This proves Theorem 1.

4. Evaluating the Resultant

Let \(A\) be a \((k \times k)\) matrix and \(b\) be any \(k\)-dimensional vector over a sufficiently large field. By an \(Ab\)-step we mean computing the product \(Ab\). Wiedemann (1986) gives a randomized Las Vegas algorithm to compute the determinant of \(Ab\). We have:

**Theorem 2.** The determinant of a \((k \times k)\) matrix \(A\) over a field with \(50k^2 \log(k)\) or more elements can be computed by a Las Vegas type randomized algorithm in \(O(k)\) \(Ab\)-steps and \(O(k^2 \log(k))\) arithmetic operations.

We show next that the product of \(M\), the Macaulay resultant matrix defined in §2, and a vector \(b \in \mathbb{Q}^D\), \(\mathbb{Q}\) the rationals, can be read off from a polynomial sum of products. In fact, this follows from the way the matrix \(M\) is defined. The entries of the vector \(b\) are labelled by the monomials of \(g_i\) in (1) as are the columns of \(M\). The product of a row labelled by the monomial \(m\) and the vector \(b\) is simply the coefficient of the monomial \(m\) in the polynomial sum of products

\[
\hat{F} = f_1 \hat{g}_1 + f_2 \hat{g}_2 + \ldots + f_n \hat{g}_n,
\]

where \(\hat{g}_i\) represents \(g_i\) with the coefficients of the monomial \(m'\) specialized to the value of the component \(b\) which is labelled by the same monomial \(m'\). This idea is best demonstrated by considering the example in §2.

**Resultant Example continued:** In order to multiply \(M\) by

\[
b = (b_1\ b_2\ \ldots\ b_9\ b_{10})^{tr},
\]

we compute \(f_1 \hat{g}_1 + f_2 \hat{g}_2 + f_3 \hat{g}_3\), where \(\hat{g}_1 = b_1 x + b_2 y + b_3\), \(\hat{g}_2 = b_4 x + b_5 y + b_6\), and \(\hat{g}_3 = b_7 x y + b_8 x + b_9 y + b_{10}\). We have

\[
f_1 g_1 + f_2 g_2 + f_3 g_3 = \cdots + \langle M_{m*}, b \rangle m + \cdots
\]

where \(\langle M_{m*}, b \rangle\) represents the dot product of the row of \(M\) labelled by the monomial \(m\) and the vector \(b\).

The product of the sub-matrix \(\Delta\) and a vector \(b' \in \mathbb{Q}_D^{D'-D}\) can be obtained in a similar fashion by starting with the matrix \(M\) and padding \(b'\) to \(b \in \mathbb{Q}^D\) with zeros in those components whose labels are the same as the labels of the columns of \(M\) deleted to obtain \(\Delta\). This observation and the use of theorems 1 and 2 lead to the following:

**Theorem 3.** The resultant of \(n\) homogeneous polynomials over a field of characteristic zero in \(n\) variables can be computed correctly by a Las Vegas randomized algorithm using \(O(nD(M(D) \log(D) + nD))\) arithmetic operations requiring to store at most \(O(D)\) field elements.

**Proof.** Using the polynomial multiplication algorithm described in the section 3, we can compute an \(Mb\)-step in \(O(nD + M(D) \log(D))\) operations. Hence we can find \(\det(M)\) and \(\det(\Delta)\) in \(O(D(M(D) \log(D) + nD))\) arithmetic operations if the values of all \(f_i\) in (4) at the points \(p_1^i, \ldots, p_n^i\) for \(0 \leq j \leq D - 1\) can be computed within that time. We show that it can be done by separating the linear, quadratic, and higher degree \(f_i\). Clearly, the linear \(f_i\) can be evaluated in \(O(nD)\) steps. For the quadratic ones, say there are \(l\) of them, the total number of terms is bounded by

\[
l \binom{n+1}{n-1} \leq \binom{1+l+n-1}{n-1}
\]
\[
\leq \binom{d + n - 1}{n - 1} = D \quad \text{for all } n \geq 4, l \leq n.
\]

For the \( f_i \) with \( \deg(f_i) \geq 3 \), for their total number of terms we have
\[
\sum_{d_i \geq 3} \binom{d_i + n - 1}{n - 1} \leq \binom{n + \sum_{d_i \geq 3}(d_i - 1)}{n - 1}
\]
\[
\leq \binom{d + n - 1}{n - 1} = D \quad \text{for } n \geq 3.
\]

The first inequality follows from
\[
\binom{r + k}{k} + \binom{s + k}{k} \leq \binom{r + s - 1 + k}{k}
\]
for all \( r, s \geq 3, k \geq 2 \),

which in turn is established by induction on \( k \).

Since the total number of terms on all the polynomials is bounded by \( D \), they can be evaluated in \( O(M(D) \log(D)) \) steps. Notice that one computes the values of the sum in (4) before performing a single sparse interpolation.

**5. Conclusion**

We have given a method that allows to compute resultants and \( u \)-resultants of polynomial systems in essentially linear space and quadratic time. We believe that our algorithm constitutes the first improvement over Gaussian elimination-based methods for computing these fundamental invariants. The resultant has many important properties for the geometry of the variety the system defines, see for example (Bajaj et al. 1988). One important property of the \( u \)-resultant is that its linear factors over the algebraic closure of the coefficient field determine the solutions in the non-singular case.

There are several problems that arise from the introduction of our new algorithm. One is that we cannot yet apply Canny’s generalized characteristic polynomial algorithm (Canny 1988b) to locate isolated points in case there are components of higher dimension in the variety. This is an important consideration for the affine case, since projectivization may introduce infinitely many solutions at infinity. The reason we cannot apply Canny’s method is that we do not know how to compute the characteristic polynomial of the Macaulay matrices in time quadratic in the dimension of the Macaulay matrices. However, we can compute the minimal polynomial of the Macaulay matrices in this time using Wiedemann’s algorithm. Using this, we can compute the “generalized minimal polynomial” of a system of homogeneous equations (in the sense of (Canny 1988b)) in the same time it takes us to compute the \( u \)-resultant of the system of equations. We conjecture that the trailing coefficient of the generalized minimal polynomial has linear factors corresponding to the isolated zeros of the system just as the \( u \)-resultant does in the purely 0-dimensional case. If so, we can find all the isolated affine zeros of the system (but not their multiplicities), in essentially the same amount of time it takes to compute all the zeros of the purely 0-dimensional case.

Secondly, it might be possible to compute the resultant in time of essentially linear dependency on the dimension of the Macaulay matrix, as is the case for the Sylvester resultant (Schwartz 1980). And finally, it appears important to us to possibly develop a theory of subresultants, again generalizing the one for Sylvester resultants (Brown and Traub 1971).

*Note added on Sep. 16, 1994:* The definition of the resultant on p. 123 is slightly flawed: \( R \) may be non-zero even when \( \det(\Delta) = 0 \) for all orderings of \( f_1, \ldots, f_n \). In such cases Theorem 3 is invalid.

*Note added on Apr. 14, 2022:* updated arithmetic complexity to \( O(nD(M(D) \log(D)+nD)) \) instead of \( O(nD^2(M(D) \log(D)+nD)) \) in Theorem 3.

**Literature Cited**


Aho, A., Hopcroft, J., and Ullman, J., *The Design and
Analysis of Algorithms; Addison and Wesley, Reading, MA, 1974.


