Comparison of Various Multivariate Resultant Formulations*

Deepak Kapur and Tushar Saxena
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, NY 12222
{kapur, saxena}@cs.albany.edu

Abstract

Three most important resultant formulations are the Macaulay, Dixon and sparse resultant formulations. For most polynomial systems, however, the matrices constructed in these formulations become singular and the projection operator vanishes identically. In such cases, perturbation techniques for Macaulay formulation such as *generalized characteristic polynomial* (*GCP*) and a method based on *rank submatrix computation* (*RSC*), applicable to all three formulations, can be used, giving four methods, Macaulay/*GCP*, Macaulay/*RSC*, Dixon/*RSC* and Sparse/*RSC*, for computing nontrivial projection operators.

In this paper, these four methods are compared. It is shown that the Dixon matrix is (by a factor up to $\mathbf{O}(e^n)$ for a certain class) smaller than the sparse resultant matrix which is (by a factor up to $\mathbf{O}(e^n)$ for a certain class) smaller than the Macaulay matrix. Empirical results confirm that Dixon/*RSC* is the most efficient, followed by Sparse/*RSC* then Macaulay/*RSC* and finally Macaulay/*GCP*, which is found to be almost impractical. All four methods are found to generate extraneous factors in the projection operator.

Efficient heuristics for interpolation, used to expand the resultant matrices, are also discussed.

1 Introduction

Solving a set of nonlinear polynomial equations or deriving conditions for the existence of their solutions is a fundamental problem in many areas of mathematics, engineering, physical and computer sciences. These methods sometimes involve elimination of variables from a given set of polynomial equations to obtain a set of polynomials whose vanishing is a necessary (and sometimes sufficient) condition for the existence of solutions. Elimination of n variables from $n+1$ equations results in a single such polynomial known as the *resultant*. In recent years, as computational power has increased, a lot of research has been devoted to eliminating variables symbolically and computing the resultant.

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

The resultant of a system of polynomials is an element of a Gröbner basis of its ideal [3] if an elimination monomial ordering is used, but this is an expensive way to compute the resultant. Until recently, a classical method by Sylvester was used to compute the resultant [9]. This method eliminates one variable from two polynomials, so to eliminate n variables from $n+1$ polynomials, it has to be applied successively. It turns out that more efficient methods, which eliminate all variables together from the set of polynomials, had been developed at the beginning of this century by mathematicians such as Cayley [7], Dixon [10] and Macaulay [20]. These multivariate resultant formulations were recently resurrected by many researchers, and have numerous applications [21, 8, 17].

Typically, these multivariate resultant formulations try to express resultants as formulas involving determinants. Ideally, one wants a single matrix whose determinant is the resultant, however this is not always possible[1]. In general, one has to settle for formulations which either express the resultant as a function of more than one determinants or give a single matrix whose determinant is a nontrivial multiple of the resultant. In this paper, we concentrate on such approaches.

Three major multivariate resultant formulations are the Macaulay [20, 4], Dixon [10, 18] and sparse [22, 5] resultant formulations. Given a set of polynomials, these formulations construct matrices, called the Macaulay matrix, the Dixon matrix and the sparse resultant matrix, respectively. In Macaulay formulation, the ratio of the determinants of the Macaulay matrix and one of its submatrices gives the resultant or some multiple of the resultant[2] (called a *projection operator*). In Dixon and sparse resultant formulations, the determinant of the respective matrices gives a projection operator. Unfortunately, sometimes while working with nongeneric, nonhomogeneous polynomial systems, some or all these three matrices can become singular and the projection operator vanishes identically. Such trivial projection operators are useless and give no information about the affine solutions of the system.

One way to extract a nontrivial projection operator is by perturbation of the polynomial system [15, 4]. Ad hoc perturbations can be used with all three formulations, but they may not work. For Macaulay formulation, Canny [4]

[1] Weyman and Zelevinsky [24] classify certain cases in which a pure determinantal formulation is possible. We have been informed by a referee that Jouanolou's work also discusses cases which permit pure determinantal formulation.

[2] In the case of generic homogeneous polynomials, Macaulay formulation results in the exact resultant.

gave a general method to perturb any polynomial system, compute its *Generalized Characteristic Polynomial* (*GCP*) and extract a nontrivial projection operator from the *GCP*. We call this the **Macaulay/**GCP method.

Another method for extraction of a nontrivial projection in the face of singular resultant matrices was outlined in our previous paper [18]. This method uses rank subdeterminant computation (*RSC*) which, though developed in conjunction with Dixon formulation, is applicable to all three formulations. This gives three more methods, **Macaulay/**RSC, **Dixon/**RSC and **Sparse/**RSC, giving a total of four methods for computing nontrivial projection operators. However, the choice between these four methods is not clear, as they have never been compared.

The purpose of this paper is to compare these four methods. We give theoretical bounds on the relative sizes of the three resultant matrices in the worst case for two different measures, viz., total degree of the polynomials and the degree of polynomials in individual variables. It is found that the Dixon matrix is the smallest, followed by the sparse resultant matrix, and the Macaulay matrix is the largest. In fact, in the worst case under the individual variable degree measure, the Dixon matrix is smaller by an exponential ($O(e^n)$) factor than the sparse resultant matrix, which in turn is smaller by an exponential (again, $O(e^n)$) factor than the Macaulay matrix. Moreover, we show by example that all four methods suffer with *extraneous factors*.

The comparison is evidenced by empirical results on 11 problems (implicitization, geometry formula derivation, equilibrium of Lorentz system, Vision etc.) and 3 random examples. Dixon/RSC seems most efficient and was able to solve all 14 problems, followed by Sparse/RSC which also solved all problems, but took much (up to 50 times) longer. Next was Macaulay/RSC which could not solve 3 problems, and on others, took much longer than both, Sparse/RSC and Dixon/RSC. Macaulay/GCP was almost impractical due to the introduction of an extra perturbation variable, and could solve only 2 problems. The software we used is described in section 5. For comparison purposes, we also give timings for Gröbner basis construction.

In section 2, we outline the three multivariate resultant formulations and point out their limitations. In section 3, we give two ways to get around these limitations and extract a nontrivial projection operator. In section 4, we give theoretical bounds on the relative sizes of the three resultant matrices. In section 5, implementational details of the four methods and some improvements on Zippel's interpolation algorithm are outlined. In section 6, we compare the four methods and give empirical evidence. Section 7 concludes giving some future research problems.

2 Multivariate Resultant Formulations

2.1 Notation and Preliminaries

Let $X = \{x_1, \ldots, x_n\}$ be a set of n variables and $A = \{a_1, \ldots, a_m\}$, a set of m parameters distinct from X. Let $\mathcal{F} = \{p_1, \ldots, p_{n+1}\} \subset \mathbb{Q}[X, A]$ be a set of $n+1$ nonhomogeneous polynomials. Let $tdeg(p, X)$ be the total degree of the polynomial p in the variable set X and $deg(p, x_j)$, the degree of the polynomial p in the variable x_j.

The objective is to eliminate the variables X from $\mathcal{F}$ and obtain a polynomial in the parameters A whose vanishing is a necessary condition for the existence of solutions to $\mathcal{F}$. Note that there may not exist a condition which is also sufficient in the nonhomogeneous case. However, there *is* the *smallest* necessary condition, which we call the *resultant*. Multiples of the resultant are called *projection operators* and their vanishing is also a necessary condition for $\mathcal{F}$ to have a common solution. The resultant may be reducible[3], but, if a projection operator is irreducible, it must be the resultant. All factors in a projection operator besides the resultant are *extraneous factors*. If a polynomial system has solutions in $\mathbb{C}^n$ under all possible specializations of the parameters (i.e., the system has an affine variety of dimension $> m-1$ in $\mathbb{C}^{n+m}$), then the resultant is identically zero. Also, an identically zero polynomial trivially qualifies as a projection operator for any system of polynomials since it is a multiple of the resultant with an extraneous factor of zero. Such trivial projection operators are obviously useless as they give no information about the solutions of polynomial systems. For polynomial systems whose affine variety is of dimension $m-1$ in $\mathbb{C}^{n+m}$, we are interested in computing projection operators which are not identically zero, at least when the system has a nontrivial common solution (i.e., a solution in the *algebraic torus* $(\mathbb{C} - \{0\})^n$ for some specialization of the parameters), and hopefully have few extraneous factors.

Let $\mathcal{I} \subseteq \mathbb{Q}[X, A]$ be the ideal generated by $\mathcal{F}$. A straightforward way to compute the resultant is by computing the Gröbner basis of $\mathcal{I}$ using an appropriate elimination ordering. The resultant is then an element of the basis. However, Gröbner basis methods are inefficient. Consider the following alternative approach.

Find another set of polynomials $\hat{\mathcal{F}} \subset \mathcal{I}$ with the following two properties:

1. Set of solutions of $\hat{\mathcal{F}}$ is preferably equal to or a superset (as small as possible) of the set of solutions of $\mathcal{F}$.
2. Number of different terms (in X) contained in the polynomials of $\hat{\mathcal{F}}$ is exactly as many as the number of polynomials in $\hat{\mathcal{F}}$.

If such an $\hat{\mathcal{F}}$ can be found, then it can be viewed as a linear system of homogeneous equations by treating each term in X (including 1) as an independent variable. A projection operator can then be found as the determinant of the coefficient matrix of this linear system.

The three most efficient methods for computing the resultant, viz. Macaulay, Dixon and sparse resultant formulations, try to find such $\hat{\mathcal{F}}$s. In this section we briefly outline these formulations. Our objective is not to describe methods in full detail, rather to give just enough information so that the reader can get an idea of the resultant matrix sizes and limitations of these formulations. Before we do that, we give an example and its resultant. This example is used throughout the paper to illustrate the applicability or limitations of different methods.

Example: (Resultant) Let $\mathcal{F} = \{p_1, p_2, p_3\} \subset \mathbb{Q}[a, b, c, x, y]$ where

$$\begin{aligned} p_1 &= ax^2 + bxy + (b + c - a)x + ay + 3(c-1), \\ p_2 &= 2a^2x^2 + 2abxy + aby + b^3, \\ p_3 &= 4(a-b)x + c(a+b)y + 4ab. \end{aligned}$$

The variables x and y are to be eliminated to obtain the resultant, a polynomial in the parameters a, b and c. From

[3] For generic polynomials the resultant is always irreducible, however under specializations, the resultant can be reducible. Eg. the resultant of $ax + b = 0$ and $bx + a = 0$ is $(a-b)(a+b)$.

the Gröbner basis of this system, the resultant is found to be the following irreducible polynomial with 75 terms:

$$
\begin{aligned}
\rho(a,b,c) = {} & 144ca^2b^3 + 28a^2c^2b^5 - 144a^4b^4 + 36c^2a^3b^2 - 72c^3a^3b^2 + 4cb^9 \\
& + a^3b^6c^2 - 12c^2a^3b^5 - 240a^4b^2 - 264a^4c^3b + 336a^4c^2b - 144a^4cb \\
& - 72a^5cb - 384a^4cb^3 + 36a^5c^2 + 112a^5b^3 - 8ca^6b^2 - 80a^5b^4 - 32a^6b^2 \\
& + 20ca^4b^4 + 96ba^5 + 48a^3b^6 + 288a^4b^3 - 96a^3b^4 + 192a^3b^3 - 16a^2b^7 \\
& + 40a^3c^3b^3 - 192a^5b^2 + 16a^5b^4c + 16a^4b^5c - 8a^3b^6c - 24a^3c^2b^3 \\
& - 8a^2b^6c^2 + 2a^2c^2b^7 - 12a^2b^7c + 80a^3b^5 + 20ca^2b^6 - 264c^2a^2b^3 \\
& - 96a^6c^2b - 80a^5b^2c^2 + 320a^5cb^2 - 40a^5cb^3 - 16a^4c^3b^3 + 44a^4c^2b^3 \\
& - 216a^3cb^3 + 2a^3c^4b^3 + 16a^3cb^4 + 108a^3b^4c^2 - 20a^3b^5c - 28a^3c^3b^4 \\
& - 8a^2c^3b^5 + 2a^5c^2b^3 + 72a^4c^4b + 4a^2c^4b^4 + 36a^3c^4b^2 + 2ac^4b^5 \\
& - 24c^2a^2b^4 + 120c^3a^2b^3 + 64a^7b^2 + 72ca^2b^4 - 52c^2ab^6 + 96a^6cb \\
& - 4cab^7 + 2c^2ab^7 + 48cab^6 - 16a^2b^6 + ac^2b^8 + 4c^3ab^6 - 24a^5c^2b \\
& - 24a^4b^2c^2 + 216a^4cb^2 + 40a^4c^3b^2 - 48a^2b^4 + 36a^5c^4 - 72a^5c^3. \qquad \square
\end{aligned}
$$

2.2 The Macaulay Formulation

For $1 \leq i \leq n+1$, let $d_i = tdeg(p_i, X)$ and $d_M = 1 + \sum_1^{n+1}(d_i - 1)$. Let the set of terms $T = \{x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \mid \alpha_1 + \alpha_2 + \ldots + \alpha_n \leq d_M\}$. Each polynomial, p_i, in $\mathcal{F}$ is multiplied with certain terms to generate $\hat{\mathcal{F}}$ with $|T|$ equations in $|T|$ unknowns (which are terms in T). The coefficient matrix of $\hat{\mathcal{F}}$ so constructed is the **Macaulay matrix**. Columns of the Macaulay matrix are labeled by the terms in T in some order, and each row corresponds to a polynomial in $\mathcal{F}$ multiplied by a certain term.

The order in which polynomials are considered for selecting multipliers results in different (but equivalent) $\hat{\mathcal{F}}$s. Let M_σ denote the Macaulay matrix for a permutation σ of polynomials in $\mathcal{F}$. Macaulay also defined N_σ, a certain square submatrix of M_σ, such that the projection operator is the ratio of the determinants of M_σ and N_σ. See [20, 4, 17] for details of the construction and proofs of these properties.

Example: (Macaulay) For the example at the beginning of this section, $T = \{x^3, x^2y, x^2, xy^2, y^3, y^2, xy, x, y, 1\}$. $\hat{\mathcal{F}}$ is obtained by multiplying p_1 by x, y and 1, p_2 by x, y and 1 and p_3 by xy, x, y and 1. The Macaulay matrix, M_σ is the following 10×10 matrix:

$$
\begin{array}{c}
\\ x \times p_1 \\ y \times p_1 \\ p_1 \\ x \times p_2 \\ y \times p_2 \\ p_2 \\ xy \times p_3 \\ x \times p_3 \\ y \times p_3 \\ p_3
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} x^3 & x^2y & x^2 & xy^2 & y^3 & y^2 & xy & x & y & 1 \end{array} \\
\left(\begin{array}{cccccccccc}
a & b & (b+c-a) & 0 & 0 & 0 & a & 3(c-1) & 0 & 0 \\
0 & a & 0 & b & 0 & a & (b+c-a) & 0 & 3(c-1) & 0 \\
0 & 0 & a & 0 & 0 & 0 & b & (b+c-a) & a & 3(c-1) \\
2a^2 & 2ab & 0 & 0 & 0 & 0 & ab & b^3 & 0 & 0 \\
0 & 2a^2 & 0 & 2ab & 0 & ab & 0 & 0 & b^3 & 0 \\
0 & 0 & 2a^2 & 0 & 0 & 0 & 2ab & 0 & ab & b^3 \\
0 & 4(a-b) & 0 & c(a-b) & 0 & 0 & 4ab & 0 & 0 & 0 \\
0 & 0 & 4(a-b) & 0 & 0 & 0 & c(a-b) & 4ab & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & c(a-b) & 4(a-b) & 0 & 4ab & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 4(a-b) & c(a-b) & 4ab
\end{array}\right)
\end{array}
$$

and N_σ is

$$
\begin{array}{c} \\ p_1 \\ p_2 \end{array}
\begin{array}{c}
\begin{array}{cc} x^2 & y^2 \end{array} \\
\left(\begin{array}{cc} a & 0 \\ 2a^2 & 0 \end{array}\right).
\end{array}
$$

Unfortunately, here both $det(M_\sigma)$ and $det(N_\sigma)$ are 0, so the resultant or a nontrivial projection operator cannot be computed directly. $\square$

Limitation: As is demonstrated in this example, this formulation may be unsuccessful in computing a nontrivial projection operator. One of the reasons is that $det(N_\sigma)$ can be identically zero, and the division cannot be carried out. Even if that is not the case, $det(M_\sigma)$ can be identically zero, hence giving only a trivial projection operator.

2.3 The Dixon Formulation

Let $\bar{X} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$ be a new set of variables and

$$
\delta(X, \bar{X}) = \begin{vmatrix}
P_{1,1} & \cdots & P_{1,n+1} \\
P_{2,1} & \cdots & P_{2,n+1} \\
\vdots & \vdots & \vdots \\
P_{n,1} & \cdots & P_{n,n+1} \\
p_1(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) & \cdots & p_{n+1}(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)
\end{vmatrix},
$$

where for $1 \leq j \leq n$ and $1 \leq i \leq (n+1)$,

$$
P_{j,i} = \frac{p_i(\bar{x}_1, \ldots, \bar{x}_{j-1}, x_j, \ldots, x_n) - p_i(\bar{x}_1, \ldots, \bar{x}_j, x_{j+1}, \ldots, x_n)}{(x_j - \bar{x}_j)},
$$

and $p_i(\bar{x}_1, \ldots, \bar{x}_k, x_{k+1}, \ldots, x_n)$ stands for uniformly replacing x_j by $\bar{x}_j$ for $1 \leq j \leq k$ in p_i. The polynomial δ is known as the **Dixon polynomial**. Now, $\hat{\mathcal{F}}$ is the set of all coefficients (which are polynomials in X) of terms in δ, when viewed as a polynomial in $\bar{X}$. The coefficient matrix of $\hat{\mathcal{F}}$ is known as the **Dixon matrix**. It can be proven that for *generic ndegree*[4] polynomials, the Dixon matrix is square - hence its determinant is a projection operator [10, 18].

Example: (Dixon) The Dixon matrix for the previous example is of dimension 2×3:

$$
\begin{array}{c} \\ \bar{x} \\ 1 \end{array}
\begin{array}{c}
\begin{array}{ccc} 1 & x & y \end{array} \\
\left(\begin{array}{lll}
\begin{array}{l} 4ca^2b^2 - c^2b^4 + 12cab^2 \\ -12ca^2b + 12a^2b - 12ab^2 \\ -c^2b^3a + cb^3a^2 - 4ab^4 \\ -cb^5 - 4a^3b^2 + 8a^2b^3 \end{array} &
\begin{array}{l} 4a^3b^2 - 8a^4b \\ -6a^3c + 6a^3c^2 \\ +6a^2c^2b - 6ca^2b \\ -cb^3a^2 - cb^4a \end{array} &
\begin{array}{l} 6a^2c^2b - 6cab^2 \\ +4a^2b^3 - 8a^3b^2 \\ -cb^5 - cb^4a \\ -6ca^2b + 6ac^2b^2 \end{array} \\
\begin{array}{l} 8ca^2b^2 - 4b^5 + 24cab^2 \\ -30ca^2b + 24a^2b - 24ab^2 \\ -6a^3c + 6a^3c^2 - cb^3a^2 \\ -8a^4b - 4a^3b^2 - cb^4a \\ +8a^2b^3 + 6a^2c^2b + 4ab^4 \end{array} &
\begin{array}{l} 12a^3b - 4a^2b^2 \\ -8a^4 + 2a^3c^2 \\ +2ca^2b^2 - 2a^4c \\ +2a^2c^2b \end{array} &
\begin{array}{l} 2ac^2b^2 + 12a^2b^2 \\ +2a^2c^2b - 2a^3cb \\ -8a^3b - 4b^3a \\ +2cb^3a \end{array}
\end{array}\right).
\end{array}
$$

Since the Dixon matrix is rectangular, its determinant cannot be computed. Hence the resultant cannot be computed by this formulation either. $\square$

Limitation: Again this example demonstrated the limitation of Dixon method. The Dixon matrix can be rectangular, hence eliminating the possibility of computing the determinant. Even if it is square, it may be singular, resulting in a trivial projection operator.

2.4 The Sparse Resultant Formulation

This approach is based on the recent results pertaining to sparse polynomial systems (Bernshtein [2], Gelfand et. al. [16] and Sturmfels [22]). Sparse resultants appeared in Sturmfels & Zelevinsky [23], its matrix formulation was given by Canny & Emiris [5] and was successively improved in [6, 11, 12] using better heuristics. The Bezout bound on the number of solutions to a set of polynomials is quite loose. Instead, this formulation uses the recently developed, and significantly tighter, *BKK bound* [16] to construct a smaller resultant matrix than Macaulay.

Newton Polytope of a polynomial p is the convex hull of the set of exponents (treated as points in $\mathbb{R}^n$) of all terms in p. *Minkowski sum* of two Newton polytopes Q and S is the set of all vector sums, $q + s$, $q \in Q$, $s \in S$. Let $Q_i \subset \mathbb{R}^n$ be the Newton polytope (wrt. X) of the polynomial p_i in

[4] $n+1$ nonhomogeneous polynomials $p_1, \ldots, p_{n+1}$ in $x_1, \ldots, x_n$ are called *generic ndegree* if all coefficients are independent parameters and there exist nonnegative integers $m_1, \ldots, m_n$ such that each $p_j = \sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} a_{j,i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n}$ *for* $1 \leq j \leq n+1$.

$\mathcal{F}$. Let $Q = Q_1 + \cdots + Q_{n+1} \subset \mathbb{R}^n$ be the Minkowski sum of Newton polytopes of all the polynomials in $\mathcal{F}$. Let $\mathcal{E}$ be the set of exponents (lattice points of $\mathbb{Z}^n$) in Q obtained after applying a small perturbation to Q to move as many boundary lattice points as possible outside Q.

Construction of $\hat{\mathcal{F}}$ here is similar to Macaulay formulation - each polynomial, p_i in $\mathcal{F}$ is multiplied with certain terms to generate $\hat{\mathcal{F}}$ with $|\mathcal{E}|$ equations in $|\mathcal{E}|$ unknowns (which are terms in $\mathcal{E}$), and its coefficient matrix is the **sparse resultant matrix**. A projection operator for $\mathcal{F}$ is simply the determinant of this matrix. Columns of the sparse resultant matrix are labeled by the terms in $\mathcal{E}$ in some order, and each row corresponds to a polynomial in $\mathcal{F}$ multiplied by a certain term.

Recall that the size of the Macaulay matrix is $|T|$. Size of the sparse resultant matrix, $|\mathcal{E}|$, is typically smaller than $|T|$, especially when the BKK bound is tighter than the Bezout bound. Algorithms in [11, 12, 6] construct matrices using some greedy heuristics which may result in smaller matrices, but in the worst case, the size can still be $|\mathcal{E}|$.

Example: (Sparse) Continuing with the same example, construction of the Newton polytopes and their Minkowski sum (after a perturbation in the positive direction) reveals that $\mathcal{E} = \{xy, xy^2, xy^3, x^2y, x^2y^2, x^2y^3, x^3y, x^3y^2, x^4y\}$. The 9×9 sparse resultant matrix is:

$$\begin{array}{cc}
 & \begin{array}{ccccccccc} xy & xy^2 & xy^3 & x^2y & x^2y^2 & x^2y^3 & x^3y & x^3y^2 & x^4y \end{array} \\
\begin{array}{c} xy \times p_1 \\ xy^2 \times p_1 \\ x^2y \times p_1 \\ xy \times p_2 \\ xy^2 \times p_2 \\ x^2y \times p_2 \\ xy \times p_3 \\ xy^2 \times p_3 \\ x^2y^2 \times p_3 \end{array} &
\left(\begin{array}{ccccccccc}
3(c-1) & a & 0 & b-a+c & b & 0 & a & 0 & 0 \\
0 & 3(c-1) & a & 0 & b-a+c & b & 0 & a & 0 \\
0 & 0 & 0 & 3(c-1) & a & 0 & b-a+c & b & a \\
b^3 & ab & 0 & 0 & 2ab & 0 & 2a^2 & 0 & 0 \\
0 & b^3 & ab & 0 & 0 & 2ab & 0 & 2a^2 & 0 \\
0 & 0 & 0 & b^3 & ab & 0 & 0 & 2ab & 2a^2 \\
4ab & c(a+b) & 0 & 4(a-b) & 0 & 0 & 0 & 0 & 0 \\
0 & 4ab & c(a+b) & 0 & 4(a-b) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4ab & c(a+b) & 0 & 4(a-b) & 0
\end{array}\right)
\end{array}.$$

The determinant of the above matrix is a projection operator with 205 terms (in expanded form):

$$\varrho = -2a^2\,(a-b-c)\left(a^2c + abc - 4ab + 4b^2\right)\rho\,(a,b,c)\,.$$

Notice that there are 3 extraneous factors. □

Limitation: Even though, in this particular example we are able to obtain a nontrivial projection operator, it may not always be the case. Although it seems rare, the sparse resultant matrix can be singular (Example 2 in Table 3), in which case the determinant will be a trivial projection operator.

3 Two Ways to Extract the Projection Operator

There are two ways to deal with degenerate resultant matrices encountered in all three formulations. One involves perturbation of the original set of polynomials and the other involves *rank submatrix computation*.

3.1 Perturbation (*GCP*)

The main problem in all three formulations is that resultant matrices can be rectangular, or even if square, they may be singular. Perturbation of polynomials can result in square and nonsingular resultant matrices [15, 4, 8]. One can try ad hoc perturbations of $\mathcal{F}$ in the hope of obtaining a nontrivial projection operator as follows. Randomly perturb the system using a perturbation variable, compute the projection operator for this perturbed system, factor it, remove any extraneous factors and finally, substitute zero for the perturbation variable to get the projection operator of the original system. Such ad hoc perturbations can be used in conjunction with all three resultant formulations, but they may not work. The following general perturbation mechanism applicable to Macaulay formulation was given by Canny in [4].

Let s be a new perturbation variable. Create a new set of polynomials $\mathcal{F}(s)$ by replacing each $p_j \in \mathcal{F}$ (for $1 \leq j \leq n$) by $p_j - s x_j^{d_j}$ and p_{n+1} by $p_{n+1} - s$. Compute the projection operator, $\varrho(s)$, of $\mathcal{F}(s)$ using Macaulay formulation. $\varrho(s)$ is known as the *Generalized Characteristic Polynomial* (*GCP*) and is a polynomial in s. The trailing coefficient (wrt. s) of *GCP* is a nontrivial projection operator of $\mathcal{F}$. Note that the size of the Macaulay matrix is same for $\mathcal{F}$ and $\mathcal{F}(s)$. For a proof of the fact that the projection operator so derived is not identically zero and it does vanish on all the affine zeros of the system, see [4]. We call this method **Macaulay/**GCP. We now finish the previous example using this method.

Example: (Macaulay/GCP**)** For the example, $det(M_\sigma)$ and $det(N_\sigma)$, both are identically zero, so we compute the *GCP*. Construct $\mathcal{F}(s)$ and letting $M_\sigma(s)$ and $N_\sigma(s)$ stand for its Macaulay and the denominator matrix, $det(N_\sigma(s)) = det(N_\sigma - Is) = s(s-a)$. $det(M_\sigma(s)) = det(M_\sigma - Is)$ is a polynomial with 705 terms. $GCP = det(M_\sigma(s))/det(N_\sigma(s))$, is a polynomial with 462 terms. Finally, the projection operator, which here is the constant term of *GCP*, has 159 terms (in expanded form):

$$\varrho = c\,(a+b)\left(a^2c + abc - 4ab + 4b^2\right)\rho\,(a,b,c)$$

Notice that there are 3 extraneous factors. □

3.2 Rank Submatrix (*RSC*)

Another way to extract a nontrivial projection operator from degenerate resultant matrices is through their submatrices. Following method is applicable to all three formulations:

1. Set up the Resultant matrix (R) of $\mathcal{F}$.
2. If any column in R is linearly independent of the other columns, then return the determinant of any *rank submatrix* of R.
3. Else, this heuristic fails.

The resultant matrix, R, can be any of the three resultant matrices, giving three methods for computing a nontrivial projection operator, viz. **Macaulay/**RSC, **Dixon/**RSC and **Sparse/**RSC. By *rank submatrix* we mean a maximal nonsingular submatrix, and its determinant can be computed by performing Gaussian elimination and returning the product of the (nonzero) pivots [18, 14]. In an earlier paper, we required a more complicated algorithm since we were deriving conditions for existence of solutions in $\mathbb{C}^n$, but for solutions in the *algebraic torus* $(\mathbb{C} - \{0\})^n$, this simple algorithm suffices. See [18] for details. Though there exist examples where the condition in step (2) is not true, they seem rare. In all examples we tried, R was degenerate many times, but the condition of step (2) was always satisfied. For the continuing example in this paper, we had not been able to compute a nontrivial projection operator using the original Macaulay and Dixon formulation. We do that now using Macaulay/RSC and Dixon/RSC respectively.

Example: (Macaulay/RSC**)** It is easily confirmed that the last column of the singular Macaulay matrix, M_σ, is linearly independent of the other columns, hence the rank of

n	d	Dix	Sparse[5]	Mac
2	2	5	15	15
2	3	12	33	36
2	4	22	66	66
3	2	14	54	56
3	3	55	198	220
4	2	42	179	210

Table 1: Matrix Sizes, Worst Total Degree Case.

n	d	Dix	Sparse	Mac
2	2	8	32	66
2	3	18	72	153
2	4	32	144	276
3	2	48	488	2024
3	3	162	1263	7140
4	2	384	*	91390

Table 2: Matrix Sizes, Worst Ind. Var. Deg. Case.

the matrix (9) decreases (becomes 8) on its removal. Projection operator is the product of the 9 pivots after Gaussian elimination and has 159 terms (in expanded form):

$$\varrho = ac\,(a+b)\left(a^2c + abc - 4ab + 4b^2\right)\rho\,(a,b,c)\,.$$

Notice that there are 4 extraneous factors. □

Example: (Dixon/*RSC*) It is easily confirmed that the first column of the rectangular Dixon matrix is linearly independent of the other columns, hence the rank of the matrix (2) decreases (becomes 1) on its removal. We do Gaussian elimination on the Dixon matrix, and the Projection operator is the product of the 2 pivots and has 75 terms (in expanded form):

$$\varrho = -b\rho\,(a,b,c)\,.$$

Again there is 1 extraneous factor but among all four methods for this example, the reader might observe that this method generates the least number of extraneous factors. □

4 Theoretical Comparison of Matrix Sizes

The size of the resultant matrix is critical in determining the efficiency of a given method. We compare the sizes of the three resultant matrices in the worst case using two measures: (i) the total degree of the input polynomials and (ii) the degree of input polynomials in each variable.

Total Degree based: For $1 \le i \le (n+1)$, let $tdeg(p_i, X) = d$. It is easily established that the sizes of Dixon, Macaulay and sparse resultant matrix are bounded by $\binom{nd}{n}$, $\binom{(n+1)d}{n}$ and $\binom{(n+1)d}{n}$ respectively. These formulas show that the Dixon matrix is smaller[6] than the other two. See Table 1 for typical worst case sizes in practice.

The total degree measure is not very fair to both Dixon and sparse resultants. Better bounds are obtained for these two if the measure for the degree of each polynomial in individual variables is considered.

Variable Degree based: For $1 \le i \le (n+1)$ and $1 \le j \le n$, let $d = deg(p_i, x_j)$. Sizes of the Dixon, Macaulay and sparse resultant matrices are bounded by $n!d^n$, $\binom{(n+1)nd}{n}$ and $((n+1)d)^n$ respectively. Using these formulas and Stirlings approximation for factorial (for asymptotically large n and d), it can be proven that Dixon matrix is smaller, by an exponential $\left(\mathbf{O}\left(e^{n+1}\right)\right)$ factor, than sparse resultant matrix, which is smaller, also by an exponential $(\mathbf{O}(e^n))$ factor, than Macaulay, and transitively the Dixon matrix is smaller, by an exponential $\left(\mathbf{O}\left(e^{2n+1}\right)\right)$ factor, than the Macaulay matrix. Typical worst case sizes are shown in Table 2. A (*) entry means the program ran out of memory.

The two worst case scenarios sketched above provide convincing arguments that the Dixon matrix is smallest of the three, followed by sparse and Macaulay is the largest. That these bounds are closely followed even in the non-worst case, real-world examples is evident from the columns labeled **Matrix Size** in Table 3. We will use this knowledge to carry out a comparison of various methods in a later section. For a comprehensive analysis of matrix sizes and comparison for multi-homogeneous systems, the reader may wish to consult [19].

5 Implementational details

Since the resultant matrices are symbolic (each entry being a polynomial in the parameters), their determinants (or rank subdeterminants) are computed using polynomial interpolation rather than direct Gaussian elimination.

Implementation of Macaulay/*GCP*: $M_\sigma(s)$ and $N_\sigma(s)$ (see section 3.1) for $\mathcal{F}(s)$ were constructed using our implementation in MAPLE. We used our C^{++} implementation of Zippel's sparse polynomial interpolation algorithm [25] to compute their determinants. Division of the two determinants and extraction of the projection operator from the *GCP* was done on MAPLE.

Implementation of *RSC* methods: We used our own MAPLE programs to construct the Macaulay and Dixon matrices, and Emiris' C program based on the algorithm in [11] to construct the sparse resultant matrix. The linear independence of any column in these matrices was checked probabilistically by substituting random numbers for parameters and performing the check on numeric (rather than symbolic) matrices. Finally, the rank subdeterminant was interpolated using our C^{++} implementation of Zippel's sparse polynomial interpolation algorithm.

Implementation of Zippel's interpolation algorithm: Given a resultant matrix, a *blackbox* is constructed which returns the value of the polynomial (determinant or the rank subdeterminant) for a particular value assignment to the parameters. This is done by (i) evaluating each entry of the symbolic resultant matrix after substituting parameter values then (ii) performing Gaussian elimination on the matrix (which now has integer entries) and finally (iii) returning the product of diagonal entries (for *GCP*) or nonzero pivots (for *RSC*).

[5] It should be pointed out that "sparse" is really a misnomer. Though the algorithm for sparse resultant is not completely insensitive to zero terms, the worst case is easily achieved for relatively sparse polynomials. Eg., for $n = 2, d = 3$, a set of three polynomials with the supports $(x^3, x^2, xy^2, y^3, 1)$ are relatively sparse (only 5 terms as opposed to possible 10), but the sparse matrix still has the worst case size.

[6] The formula for Dixon matrix size above is loose and the exact size is always smaller. Also, more efficient algorithms for sparse resultants use heuristics which may construct matrices of smaller size than the formula above indicates. However, such heuristics are not considered here because the order of the bound does not change.

Ex	Application	No. of Polys	**Dixon**			**Macaulay**			**Sparse**			**Gröbner**
			Matrix Size	Cnstr. Time	*RSC* Time	Matrix Size	*GCP* Time	*RSC* Time	Matrix Size	Cnstr. Time	*RSC* Time	*Macaulay* Time
1	Geom. Reason.	3	13	0.18	342	55	*	4608	34	0.3	566	65863
2	Formula Deriv.	5	14	0.36	76	330	*	*	86	9.8	4094	585
3	Formula Deriv.	3	5	0.08	*[7]	15	N/R	*[7]	14	0.2	*[7]	*
4	Geometry	3	4	0.08	1021[7]	15	*	*[7]	14	0.2	*[7]	15429
5	Random	4	6	0.06	*[7]	84	N/R	*[7]	27	0.7	*[7]	2207
6	Implicitization	3	10	0.1	0.58	45	3741	13.15	16	0.2	0.71	1
7	Implicitization	3	18	0.45	47	66	N/R	604	55	2	519	12530
8	Implicitization	3	18	9.33	126	153	*	94697	54	1.9	1369	70485
9	Random	4	7	0.21	220	56	*	9972	24	1.2	412	13508
10	Random	5	36	1.1	8130	1365	*	*	151	40	201558	*
11	Implicitization	3	11	0.06	3.01	36	N/R	32.58	33	0.7	31.04	57
12	Equilibrium	4	5	0.05	0.24	20	N/R	1.53	20	0.7	2.36	5
13	Vision	6	6	3.3	0.05	792	*	*	60	248	10.53	6
14	Comput. Bio.	3	8	0.067	0.27	28	11.18	0.68	16	0.2	0.33	1

Table 3: Empirical Data

The interpolation algorithm then loops through m stages (where m is the number of parameters), each time lifting one parameter by making a number of calls to the *blackbox*. At the end of the m^{th} stage, the polynomial has been completely interpolated. See [25] for details.

We performed all computations in a finite field modulo a large prime. If the exact projection operator is required, the computations can be performed in various finite fields modulo different primes until the product of the primes is larger than a predetermined coefficient bound. Finally exact coefficients can be lifted from their images in various finite fields using Chinese remainder theorem. If the coefficient bound is unknown (or too loose), lift the coefficients successively after each finite field computation until they stabilize.

In practice, the most expensive operation during interpolation is the *blackbox* computation, and the possibilities for improvement lie in reducing the number of calls to *blackbox*. Two main heuristics we used are as follows:

1. **Estimating individual variable degree bounds:** Zippel's algorithm assumes the same degree bound (say D) for each parameter at any stage. This assumption is costly for later stages if the actual degree of the parameter being lifted is less than D. Fortunately, when the first parameter is being lifted, this assumption is not too costly. Based on this observation, using a crude bound D, we interpolate univariate polynomials in all the parameters before starting multivariate interpolation. From these univariate polynomials in each parameter, we determine individual degree bound for each parameter. Later, when interpolating the multivariate polynomial, we use these individual degree bounds at each stage. This results in a speedup of the algorithm since it reduces the number of calls to *blackbox*.

2. **Order of interpolation:** Once the individual degree bound for each parameter is known, it turns out that the number of calls to *blackbox* is sensitive to the order in which the parameters are lifted. If parameters are lifted in increasing order of their individual degree bounds, fewer calls are made to the *blackbox* which resulted in a speedup.

Ex	**Dix/** *RSC*	**Mac/** *GCP*	**Mac/** *RSC*	**Sparse/** *RSC*
3	6.56	N/R	26.5	76.2
4	1.6	*	84	290
5	0.9	N/R	850	23.4

Table 4: Timings - direct determinant expansion in MAPLE.

Using these two heuristics, we are able to obtain a substantial speedup over a straightforward implementation of Zippel's algorithm.

6 Empirical Results and Comparison

All empirical results are listed together in Table 3. The 14 examples used in Table 3 can be obtained in MAPLE format by anonymous ftp from `ftp://ftp.cs.albany.edu /pub/saxena/examples.14`. All timings are in seconds on a 64Mb SUN Sparc10. A (*) in a *Time* column either means that the resultant cannot be computed even after running for more than a day or the program ran out of memory. An N/R in the GCP column means there exists a polynomial ordering for which the Macaulay and the denominator matrix are nonsingular and therefore GCP computation is not required. Besides resultant formulation timings, we also present timings for computing resultant using Gröbner basis construction with block ordering (variables in one block and parameters in another) using the *Macaulay* [1] system. Gröbner basis computations are also done in a finite field, in fact, with a much smaller characteristic (31991) than in the resultant computations. We also tried Gröbner basis using *GB* system of Faugere [13], but it does not support block ordering, and lexicographic ordering resulted in much inferior performance.

Examples 3, 4 and 5 consist of generic polynomials with numerous parameters and dense resultants. Interpolation methods are not appropriate for such examples and timings using straightforward determinant expansion in MAPLE are in Table 4.

[7] See Table 4 for timing using direct determinant expansion.

In all four methods, the time taken to compute the resultant depends on the cost of the two steps - (i) constructing the resultant matrix and (ii) interpolating the resultant from the resultant matrix.

Cost of Matrix Construction is insignificant (see the columns labeled Cnstr. Time in Table 3) for all formulations on all examples. It should be noted that the construction of Dixon matrices was done on MAPLE using a straightforward implementation without any improvements. An efficient implementation in C would further reduce the already insignificant Dixon matrix construction time. So, the total time to compute the resultant is dominated by the interpolation step.

Interpolation Cost (see section 5) depends on the cost of an individual *blackbox* call and the total number of calls to *blackbox*. The cost of an individual call to *blackbox* includes the cost of substituting values for parameters in the matrix and the cost of performing Gaussian elimination on the resulting matrix. Even though the cost of evaluating an individual entry in Dixon matrix is higher (because the entries are polynomials of degree at most $n+1$ in the coefficients of the original polynomials), the number of such evaluations is determined by the matrix size and is much smaller than for Macaulay and sparse matrices. Consequently, the time taken for substituting values for parameters in the Dixon matrix is smaller than the time taken for substitution in the Macaulay and sparse matrices.

The time taken by a call to *blackbox* is however totally dominated by Gaussian elimination after substitutions have been made into a matrix. Below we compare the four methods based on time taken to perform Gaussian elimination in a *blackbox* call and number of calls to the *blackbox*.

6.1 Dixon/RSC vs. Macaulay/RSC vs. Sparse/RSC

1. **Cost of Gaussian Elimination:** Once the substitution phase is over, the matrix entries are only entries from some finite field, and the cost of Gaussian elimination depends solely on the size of the matrices. Due to the much smaller size of the Dixon matrix, time taken for Gaussian elimination is much less than for the other two. Since Gaussian elimination dominates the *blackbox* call, the *cost of an individual blackbox call is the least for Dixon matrix*, followed by the sparse and finally the Macaulay matrix.

2. **Number of calls to** ***blackbox*****:** The number of calls to *blackbox* depends on the structure of the polynomial being interpolated. If there are no extraneous factors, the polynomial being interpolated in all three cases is the same, and so the number of calls to *blackbox* is also the same in all three methods. If extraneous factors are present, then the cost depends on the number of terms in the projection operator. Usually the number of terms are less if there are less extraneous factors (note that this is not true in general).

Since the interpolation cost depends mostly on the two factors discussed above, *total time taken to interpolate the resultant is smallest for Dixon/RSC followed by Sparse/RSC and largest for Macaulay/RSC.*

This gradation of efficiency is reflected in Table 3. All the examples were completed using the Dixon/RSC method. Sparse/RSC also solved all the examples, but always took much longer than Dixon/RSC. Macaulay/RSC could not complete 2 problems and took much longer than Sparse/RSC (for most problems) and Dixon/RSC (for all problems).

6.2 RSC vs. GCP

1. **Cost of Gaussian Elimination:** Size of the resultant matrix of a polynomial system after perturbation is the same as that before perturbation (see section 3). So, the cost of Gaussian elimination is the same in GCP and RSC methods.

2. **Number of calls to** ***blackbox*****:** Due to the introduction of a new perturbation variable, one more iteration (or stage) is need during interpolation for that variable. This results in more calls to all the procedures including *blackbox*. Secondly, due to the new variable, the number of terms in the polynomials at each stage of interpolation also becomes larger. This raises the number of calls to *blackbox* even further[8].

In practice, this rise in the number of calls to *blackbox* almost makes the GCP method impractical. On the other hand, cost of the RSC method remains the same. These observations about the GCP method are confirmed by comparing the entries of column labeled Macaulay/GCP against the columns Dixon/RSC, Macaulay/RSC and Sparse/RSC.

While working with Macaulay formulation, it is often (9 out of the 14 examples in Table 3) the case that one of the two methods, RSC or GCP, need to be resorted to, and analysis suggests that RSC should be chosen due to its efficiency.

The GCP method does have a distinct merit in that it can always extract the proper affine components of the variety, whereas other methods may fail. However, the failure of RSC methods seems to occur very rarely and they have been successfully applied to all nontrivial examples in the paper.

7 Conclusion

While working with nongeneric, nonhomogeneous polynomials, all four elimination methods generate extraneous factors in the projection operators (see the example used in this paper). So the only factor in making a choice between the four methods is efficiency. To this end, we find that the Dixon formulation coupled with rank submatrix computation is the fastest way to compute a nontrivial projection operator of a set of polynomials. The main reason is that Dixon matrix can be smaller, by a factor that is up to exponential in the number of variables, than the other two resultant matrices, viz., Macaulay and sparse.

The comparison of matrix sizes was done in the worst case scenario under two different measures. Being worst cases, they don't take into account the sparsity of polynomials encountered in real-world. Even though the real-world examples we presented show that the Dixon matrix is indeed much smaller than the other two, a theoretical analysis which takes into account sparsity needs to be done. Our preliminary work in this direction [19] shows that, like the sparse resultant formulation, Dixon formulation also exploits the Newton polytopes and takes into account the sparsity of the input polynomials, though in a different fashion.

Extraneous factors occur in all methods in this paper, however, it is not clear which method generates least factors. Any bounds on the number of extraneous factors generated by the four methods will be very useful. Another important research direction is to discover ways of eliminating extraneous factors from these methods without compromising their efficiency.

[8] It may be possible to avoid the GCP and directly compute its trailing coefficient, but this is likely to require as many *blackbox* calls.

Acknowledgement: We thank Ioannis Emiris for giving us a preliminary version of his sparse resultant program.

References

[1] Bayer D., Stillman M., *Macaulay User's Manual*, Cornell University, Ithaca, NY.

[2] Bernshtein D.N., The Number of Roots of a System of Equations, *Funktsional'nyi Analiz i Ego Prilozheniya,* 9(3):1-4.

[3] Buchberger B., Gröbner bases: An Algorithmic Method in Polynomial Ideal Theory, *Multidimensional Systems Theory,* N.K. Bose, ed., D. Reidel Publ. Co., 1985.

[4] Canny J., Generalized Characteristic Polynomials, *Journal of Symbolic Computation*, 9:241-250.

[5] Canny J. and Emiris I., An Efficient Algorithm for the Sparse Resultant, *Proc. of AAECC 93,* LNCS 263, Springer-Verlag, May 1993, pp. 89-104.

[6] Canny J. and Pedersen P., An Algorithm for Newton Resultant, Technical Report, *Cornell University*, CornellCS:TR93-1394, 1993.

[7] Cayley A., On the Theory of Elimination. *Cambridge and Dublin Mathematical Journal,* III, 1865, 210-270.

[8] Chionh E., *Base Points, Resultants and Implicit Representation of Rational Surfaces.* Ph.D. thesis, Dept. Comp. Sci., Univ. of Waterloo, 1990.

[9] Collins G.E., The Calculation of Multivariate Polynomial Resultants. *JACM,* 18 (4), (1971), 515-532.

[10] Dixon A.L., The Eliminant of Three Quantics in Two Independent Variables. *Proc. London Mathematical Society,* 6, 1908, 468-478.

[11] Emiris I. and Canny J., A Practical Method for the Sparse Resultant, *Proc. ACM ISSAC*, 183–192, Kiev, July 1993.

[12] Emiris I. and Canny J., Efficient Incremental Algorithms for the Sparse Resultant and the Mixed Volume, Technical Report, *Univ. of California, Berkeley*, 1994.

[13] Faugère J.C., Gb Reference Manual, PoSSo project, *Institut Blaise Pascal*, France.

[14] Gantmatcher F.R., *Matrix Theory*, vol. 1. Chelsea Publishing, 1959, Chap. 2.

[15] Grigoryev D.Yu. and Chistov A.L., *Sub-exponential Time Solving of Systems of Algebraic Equations.* LOMI Preprints E-9-83 and E-10-83, Leningrad, 1983.

[16] Gelfand I.M., Kapranov M.M. and Zelevinsky A.V., *Discriminants, Resultants and Multidimensional Determinants*, Birkhäuser, Boston, 1994.

[17] Kapur D. and Lakshman Y.N., Elimination Methods: an Introduction. *Symbolic and Numerical Computation for Artificial Intelligence* B. Donald et. al. (eds.), Academic Press, 1992.

[18] Kapur D., Saxena T. and Yang L., Algebraic and Geometric Reasoning using Dixon Resultants, *Proc. ACM ISSAC*, Oxford, England, July 1994.

[19] Kapur D and Saxena T., Sparsity Considerations in the Dixon Resultant Formulation, manuscript under preparation.

[20] Macaulay F.S., *The Algebraic Theory of Modular Systems*, Cambridge Tracts in Math. and Math. Phys., 19, 1916.

[21] Sederberg T. and Goldman R., Algebraic Geometry for Computer-Aided Design, *IEEE Computer Graphics and Applications*, Vol. 6, No. 6, pp. 52-59, 1986.

[22] Sturmfels B., Sparse Elimination Theory, *Proc. Computat. Algebraic Geom. and Commut. Algebra*, D. Eisenbud and L. Robbiano, eds., Cortona, Italy, June 1991.

[23] Sturmfels B. and Zelevinsky A., Multigraded resultants of the Sylvester type, *Journal of Algebra*, 1992.

[24] Weyman J. and Zelevinsky A., Determinantal Formulas for Multigraded Resultants, *Journal of Algebraic Geometry*, pp 569-597, 1994.

[25] Zippel R. *Effective Polynomial Computation*, Kluwer Academic Publishers, Boston, 1993.