Extracting Sparse Resultant Matrices from Dixon Resultant Formulation*

Arthur Chtcherba Deepak Kapur

Department of Computer Science
University of New Mexico
Albuquerque, NM 87131
e-mail: {*artas,kapur*}*@cs.unm.edu*

February 21, 2000

1 Introduction

A new method for constructing sparse resultant matrices for a multivariate polynomial system is proposed based on the classical constructions of Cayley and Dixon for setting up resultant matrices. Unlike other sparse resultant matrix construction algorithms [EC95, CE96], the proposed method does not explicitly use the support of the polynomial system in its construction. Instead, the algorithm implicitly exploits the sparse structure of the polynomial system since the construction is based on the Dixon matrix. It was shown in [KS96] that the generalized Dixon resultant formulation implicitly exploits the sparse structure of the polynomial system.

Multivariate resultants and related elimination methods have been found useful in many engineering and design applications including robotics, kinematics, design and analysis of nano devices in nanotechnology, image understanding and vision, geometric and solid modeling, molecular dynamics, control theory and chemical and combustion applications. Particularly, the problems of solving a system of nonlinear polynomial equations as well as deriving conditions on parameters, if any appearing in equations, such that the system has a solution, arise in many application domains including the ones listed above. For an overview as well as details, the reader may consult [Hof89, DKM92, Emi94].

Three major multivariate resultant formulations are the Macaulay [Mac16, Can90], Dixon [Dix08, KSY94, KS95] and sparse [Stu91, CE93, Emi94] resultant formulations. Given a polynomial system, these formulations construct matrices, called the Macaulay matrix, the Dixon matrix and the sparse resultant matrix, respectively. In the Macaulay formulation, the ratio of the determinants of the Macaulay matrix and one of its sub-matrices gives the resultant or some multiple of the resultant[1] (called a *projection operator*). In Dixon and sparse resultant formulations, the determinant of the respective matrices gives a projection operator. In cases when non-generic, non-homogeneous polynomial systems lead to singular matrices, a projection operator can be extracted using perturbation and linear algebra techniques, as discussed in [Can90, KSY94, KS95, Emi94].

In [KS95], the performance of these methods have been empirically compared on a suite of examples from diverse applications. In [KS96], it was shown that the generalized Dixon formulation as proposed in [KSY94] implicitly exploits the sparse structure of the polynomial system. This is in contrast to sparse resultant formulation where the sparse structure of the polynomial system is explicitly exploited while building the sparse resultant matrix from the power products of the polynomial system. Specifically, for the unmixed case (in which every polynomial in the polynomial system has the same set of power-products), it was proved in [KS96] that the computational complexity of the generalized Dixon formulation is governed by the mixed volume of the Newton

*This research is supported in part by NSF grant nos. CCR-9996144, and CDA-9503064.

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

polytope formed from the power-products appearing in the polynomials, not by the Bezout bound as is the case for Macaulay resultants.

Since that result, we have often wondered whether it was possible to construct sparse resultant matrices in a simple and easy manner, in contrast to the incremental algorithm developed in [EC95] as well as the lifting and subdivision based algorithm developed in [CE96]. Subdivision as well as incremental algorithms are well-known to take considerable time in practice in building the sparse resultant matrices.

The above question is settled affirmatively in this paper. Particularly, it is shown how the sparse resultant matrices can be constructed directly and easily from the Dixon formulation, without having to explicitly construct the support of a polynomial system. An algorithm for constructing sparse matrices from the polynomial system is given. Its complexity is analyzed. This algorithm is empirically and theoretically compared with the subdivision and incremental methods for constructing sparse resultant matrices proposed in [EC95, CE96].

Another important issue in elimination theory is that of extraneous factors in the projection operators computed from such sparse matrices. For a suite of examples, extraneous factors arising from the sparse matrices constructed using the proposed algorithm are compared with the extraneous factors arising from the sparse resultant matrices based on subdivision and incremental algorithms.

In section 3.2, a structural relationship between sparse matrices and Dixon matrices arising from the Dixon formulation is explored. Such an analysis, it is believed, will reveal how sparse matrices so constructed might be useful in determining extraneous factors arising in projection operators computed from the generalized Dixon resultant formulation as discussed in [KSY94, Sax97].

2 Background

This section introduces some notation and definitions. A reader can find a general introduction in [CLO96] and [CLO98].

Let $\mathbb{C}[\mathbf{x}] = \mathbb{C}[x_1, \ldots, x_d]$, and denote by $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_d^{\alpha_d}$.

Let $\mathbf{P} = \{f_1, f_2, \ldots, f_k\}$ where $f_i = \sum_\alpha c_\alpha \mathbf{x}^\alpha \in \mathbb{C}[\mathbf{x}]$. In general, this polynomial system $\mathbf{P}$ of k equations can be expressed in matrix notation as

$$\mathbf{P} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{k,1} & c_{k,2} & \cdots & c_{k,n} \end{pmatrix} \begin{pmatrix} \mathbf{x}^{\alpha_1} \\ \mathbf{x}^{\alpha_2} \\ \vdots \\ \mathbf{x}^{\alpha_n} \end{pmatrix} = C_P \times Y,$$

where $Y = [\mathbf{x}^{\alpha_1}, \mathbf{x}^{\alpha_2}, \cdots, \mathbf{x}^{\alpha_n}]^T$ is a column vector corresponding to an ordered set of terms appearing in $\mathbf{P}$, using some total ordering on terms, and C_P is the coefficient matrix of $\mathbf{P}$.

Definition 2.1 (Specialization map) *Given* $f(x_1, \ldots, x_d) = \sum_{i=1}^n c_i \mathbf{x}^{\alpha_i}$ *in* $\mathbb{C}[x_1, \ldots, x_d, c_1, \ldots, c_n]$ *let*

$$\begin{aligned} \phi : \mathbb{C}[x_1, \ldots, x_d, c_1, \ldots, c_n] &\rightarrow \mathbb{C}[x_1, \ldots, x_d] \\ \phi(f) &= \sum_{i=1}^n \phi(c_i)\mathbf{x}^{\alpha_i} \end{aligned}$$

where $\phi(c_i) \in \mathbb{C}$. *The mapping* ϕ *is called a specialization map.*

Let $\phi(P)$ and/or $\phi(C_P)$ stand for the polynomial system and its coefficient matrix, respectively, when all the parameters of the polynomial system have been specialized.

Given a specialized polynomial system, its zeros (equivalently, the solution space of the associated polynomial equation system), is called a variety. One can consider different kinds of varieties depending upon the zeros under consideration.

$$Projective(P) = \{ \varepsilon = (\varepsilon_1, \ldots, \varepsilon_d) \,|\, f(\varepsilon_1, \ldots, \varepsilon_d) = 0 \quad \text{for all} \quad f \in P \quad \text{and} \quad \varepsilon \in \mathbb{P}^d_{\mathbb{C}} \}$$

$$
\begin{array}{rcllll}
Affine(P) & = & \{\,\varepsilon = (\varepsilon_1,\ldots,\varepsilon_d)\,|\,f(\varepsilon_1,\ldots,\varepsilon_d) = 0 & \text{for all} \quad f \in P & \text{and} & \varepsilon \in \mathbb{C}^d\ \} \\
Toric(P) & = & \{\,\varepsilon = (\varepsilon_1,\ldots,\varepsilon_d)\,|\,f(\varepsilon_1,\ldots,\varepsilon_d) = 0 & \text{for all} \quad f \in P & \text{and} & \varepsilon \in \mathbb{C}^d, \varepsilon_i \neq 0\ \}
\end{array}
$$

In the affine case, we are not interested in the trivial solution, since if polynomial system admits the trivial solution, there is no condition on its coefficients. Thus, let $\mathbb{A}^d = \mathbb{C}^d - \{\,0\}$ and $\mathbb{T}^d = (\mathbb{C} - \{\,0\})^d$. Note that $\mathbb{T}^d \subset \mathbb{A}^d \subset \mathbb{P}^d$.

Definition 2.2 *Given a polynomial system of $d+1$ polynomials in d variables, the (projective, affine, toric, respectively) resultant is 0 if and only if they have a nontrivial common zero in a given variety, for example, $\mathbb{P}^d$, $\mathbb{A}^d$ or $\mathbb{C}^d$.*

Given a polynomial system of $d+1$ polynomials in d variables and k parameters, the vanishing of the (projective, affine, toric, respectively) resultant in terms of the parameter values is a necessary and sufficient condition for the system to have a nontrivial common zero in the corresponding variety. In that sense, it is the smallest such polynomial.

A matrix M is called a **resultant matrix** for a given polynomial system $\mathbf{P}$ if the resultant of $\mathbf{P}$ or a nontrivial multiple of the resultant (called the *projection operator*) w.r.t some variety can be obtained from it. Typically, the projection operator is the determinant, ratios of some determinants or a function of determinants of maximal minors. Hence, depending where the zeros of the polynomial system are being considered, i.e., whether $\mathbb{P}^d$, $\mathbb{A}^d$ or $\mathbb{T}^d$, different corresponding resultant matrices can be constructed.

Definition 2.3 *Given a polynomial $f = c_{\alpha_1}\mathbf{x}^{\alpha_1} + c_{\alpha_2}\mathbf{x}^{\alpha_2} + \cdots + c_{\alpha_n}\mathbf{x}^{\alpha_n}$, let the support of f, $S_{\mathbf{x}}(f) =, \{\,\alpha\,|\,\mathbf{x}^\alpha \in f$ and $c_\alpha \neq 0, \alpha \in \mathbb{N}^d\}$.*

The convex hull of the support $S_{\mathbf{x}}(f)$ is called its Newton polytope, denoted by $\mathcal{N}(f)$.

Definition 2.4 *Let P and Q be two supports (or the associated Newton polytopes). The **Minkowski sum** of $\mathcal{P}$ and $\mathcal{Q}$, denoted by $\mathcal{P}+\mathcal{Q}$, is $\{p+q \mid p \in \mathcal{P}$ and $q \in \mathcal{Q}\}$, where $p+q$ is the vector sum.*

Definition 2.5 (Mixed Volume) *Given convex polytopes $Q_1,\ldots,Q_d \in \mathbb{R}^d$, the mixed volume $\mu(Q_1,\ldots,Q_d)$ is the coefficient of $\lambda_1\lambda_2\cdots\lambda_d$ in $Vol(\lambda_1 Q_1 + \cdots + \lambda_d Q_d)$.*

Theorem 2.1 (BKK bound) *(Bernstein'75) Given a polynomial system $\mathbf{P} = \{f_1,\ldots,f_d\}$ and $Q_i = \mathcal{N}(f_i)$, the number of solutions $\mathbf{P}$ has in $(\mathbb{C}^*)^d$ is equal to $\mu(Q_1,\ldots,Q_d)$ for most choices of the coefficients of $\mathbf{P}$.*

See [GKZ94] for a full treatment of the subject.

There are many ways to construct resultant matrices. The two most popular ones are (i) Sylvester's dialytic method, in which polynomials are multiplied by power products, and (ii) Cayley-Dixon's discrete differential method.

Definition 2.6 *Given a polynomial $f(x_1,\ldots,x_d)$ and a monomial set $X = \{\alpha_1,\alpha_2,\ldots,\alpha_n\}$, let*

$$Xf = \{\mathbf{x}^\alpha f | \mathbf{x}^\alpha \in X\}$$

be the set of polynomials obtained from f. The set X is sometimes referred as a multiplier set for f.

It is easy to see that if ε is solution of f, it is also a solution Xf. Also if ε is solution of Xf and there exists a multiplier $\mathbf{x}^\alpha \in X$ such that $\varepsilon^\alpha \neq 0$, then ε is also a solution of f.

Most resultant methods are based on identifying multiplier sets for each polynomial $f \in P$ so that

1. the zeros of $\mathbf{P}$ are the same as the zeros of the new polynomial system $\mathbf{P}'$ consisting of polynomials obtained by multiplying with the multiplier sets, and
2. there are at least as many polynomials as the terms in the resulting polynomial system so that linear algebra constructions can be exploited for finding zeros.

Definition 2.7 *A resultant matrix is* **sparse** *if it consists of entries that are coefficients of terms in the polynomial system, and its size is determined by the polytopes corresponding to the supports of the polynomials in the system.*

The focus of this paper is mainly on constructing sparse resultant matrices. As already mentioned, there are three main resultant formulations: Macaulay, Newton sparse, and Dixon. The Macaulay matrix corresponds to computing zeros in the projective variety $\mathbb{P}^d$, and Newton sparse matrix is constructed for finding zeros in the toric variety $\mathbb{T}^d$. The proposed Dixon sparse resultant matrix (as well the Dixon resultant formulation on which the construction is based) corresponds to finding zeros in the affine variety $\mathbb{A}^d$.

There are at least two different heuristics to construct Newton matrices: subdivision and incremental [EC95, CE96]; these heuristics are developed based on the BKK bound, and explicitly construct the support of the polynomial system. In contrast, our proposal uses classical construction and does not explicitly use the support of the polynomial system. We will compare our proposal with these two heuristics even though our proposal computes affine resultants whereas both subdivision and incremental algorithms are used to set up sparse matrices for toric resultants. Since $\mathbb{T}^d \subset \mathbb{A}^d$, matrices for toric resultants are smaller than those for affine.

2.1 Properties of a resultant matrix

As stated above, resultant matrices can be constructed using the dialytic method by constructing multiplier sets for each polynomial in a polynomial system $\mathbf{P}$. Below, $\mathbf{P}'$ stands for the polynomial system obtained from $\mathbf{P}$ by multiplying polynomials in $\mathbf{P}$ by their respective multiplier sets.

By linear algebra, a polynomial system $\mathbf{P}' = M_{\mathbf{P}} \times Y = 0$ *may* have a nontrivial solution only when the rank of $M_{\mathbf{P}}$ is smaller than the number of columns. It is assumed that some of the coefficients $c_{i,j}$ in $M_{\mathbf{P}}$ are symbolic, and $\mathbf{P}'$ (and hence $\mathbf{P}$) has no solution for arbitrary values of parameters in the variety under consideration (i.e. $\mathbb{P}^d$, $\mathbb{A}^d$ or $\mathbb{T}^d$). The goal is to derive condition on these coefficients for $\mathbf{P}$ to have a solution in a given variety. Clearly, if the rank of $M_{\mathbf{P}}$ is equal to number of columns, then vanishing of the determinant of $M_{\mathbf{P}}$ is a condition on the existence of solution for $\mathbf{P} = 0$. But there is a weaker condition on a matrix to be resultant matrix.

Theorem 2.2 *Given a matrix $M_{\mathbf{P}}$ for a polynomial system $\mathbf{P}$, let C_i be a linearly independent column of $M_{\mathbf{P}}$ labeled by a monomial $\mathbf{x}^{\alpha_i}$. Then for any specialization map ϕ such that $\phi(P) = 0$ has a solution ε with $\varepsilon^{\alpha_i} \neq 0$, the determinant of a maximal minor of $M_{\mathbf{P}}$ is a non-trivial multiple of the resultant of* $\mathbf{P}$.

A proof can be found in [Sax97] and [KSY94]. Below, the main steps are sketched.
Proof: If $\varepsilon^{\alpha_i} \neq 0$, then

$$\mathrm{rank}(\phi(M_{\mathbf{P}})) = \mathrm{rank}(\phi(M_{\mathbf{P}} - \{C_i\})), \text{i.e.,}$$

$$\phi(C_i) = \sum_{\substack{j=1 \\ j \neq i}}^{k} \frac{\varepsilon^{\alpha_j}}{\varepsilon^{\alpha_i}} \phi(C_j),$$

where k is the number of columns, i.e., $\phi(C_i)$ is not an independent column in $\phi(M_P)$.

Let Q be any maximal minor of $M_{\mathbf{P}}$.

$$\text{If} \quad \mathrm{rank}(M_{\mathbf{P}} - \{C_i\}) < \mathrm{rank}(M_{\mathbf{P}}) \quad \text{then} \quad \phi(\det(Q)) = 0.$$

Since $rank(\phi(M_{\mathbf{P}})) = rank(\phi(M_{\mathbf{P}} - \{C_i\})) \leq rank(M_{\mathbf{P}} - \{C_i\}) < rank(M)$, i.e., $\phi(\det(Q)) = 0$.
$\square$.

The above also implies that the resultant must divide the determinant $\det(Q)$ of any maximal minor $Q = \mathrm{minor}_{\max}(M_{\mathbf{P}})$, since both vanish whenever there exist solution to $\mathbf{P} \equiv 0$, and the resultant is the smallest such condition.

Below, we identify multiplier sets for polynomials in $\mathbf{P}$ to construct an equivalent polynomial system $\mathbf{P}'$ with the desired property that ε is a non-trivial solution of $\mathbf{P}$ if and only if it is a nontrivial solution of $\mathbf{P}'$. Further, the coefficient matrix from $\mathbf{P}'$ has the independent column so that Theorem 2.1 is applicable.

3 Constructing Dixon Sparse Matrix

As stated above, there are at least two methods for constructing sparse resultant matrices [CE96], [EC95] based on the support of the polynomial system $\mathbf{P}$. The idea is to get multiplier sets so that the resulting matrix has the rank equal to the number of columns. This is quite explicit in the incremental construction proposed in [EC95].

As already noted in Theorem 2.2, we only need to have an independent column, not necessarily the full rank.

Below, the proposed construction will assume that the roots are located in the affine space, i.e., we will construct a matrix M_P which we will call **Dixon Sparse matrix**. The size of this matrix has larger lower bound than a corresponding Newton sparse matrix as the underlying varieties are different. But the proposed construction has a considerably smaller upper bound than the upper bounds for the subdivision and incremental algorithms, for Newton sparse matrices.

Definition 3.1 *Given $\mathbf{P} = \{f_0, f_1, \ldots, f_d\} \subset \mathbb{C}[\mathbf{x}]$, define its **Dixon polynomial** to be*

$$\theta(f_0,\ldots,f_d) = \prod_{i=1}^{d} \frac{1}{\overline{x}_i - x_i} \begin{vmatrix} f_0(x_1,x_2,\ldots,x_d) & f_1(x_1,x_2,\ldots,x_d) & \cdots & f_d(x_1,x_2,\ldots,x_d) \\ f_0(\overline{x}_1,x_2,\ldots,x_d) & f_1(\overline{x}_1,x_2,\ldots,x_d) & \cdots & f_d(\overline{x}_1,x_2,\ldots,x_d) \\ \vdots & \vdots & \ddots & \vdots \\ f_0(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_d) & f_1(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_d) & \cdots & f_d(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_d) \end{vmatrix}.$$

Hence $\theta(f_0, f_1, \ldots, f_d) \in \mathbb{C}[\mathbf{x}, \overline{\mathbf{x}}]$, where $\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_d$ are new variables.
Let $\pi_i(\mathbf{x}^\alpha) = \overline{x}_1^{\alpha_1} \cdots \overline{x}_i^{\alpha_i} x_{i+1}^{\alpha_{i+1}} \cdots x_d^{\alpha_d}$, where $i \in \{0, \ldots, d\}$. Note that $\pi_0(\mathbf{x}^\alpha) = \mathbf{x}^\alpha$ and

$$\pi_i(f(x_1,\ldots,x_d)) = f(\overline{x}_1,\ldots,\overline{x}_i,x_{i+1},\ldots,x_d).$$

Note that in the expression for Dixon polynomial, the $(i,j)^{th}$ entry in the determinant is $\pi_i(f_j)$, for $i = 0, \ldots, d$ and $j = 0, \ldots, d$.

Definition 3.2 *A Dixon polynomial $\theta(f_0, f_1, \ldots, f_d)$, can be written in the bilinear form*

$$\theta(f_0, f_1, \ldots, f_d) = \overline{X} \Theta X^T,$$

where $\overline{X} = [\overline{x}^\alpha | \overline{x}^\alpha \in \theta]$ and similarly $X = [x^\alpha | x^\alpha \in \theta]$ are treated as row vectors.
*Matrix Θ is called the **Dixon matrix**.*

The Dixon polynomial can be decomposed into a sum of smaller Dixon polynomials of polynomial systems with $d+1$ monomials.

Proposition 3.1 *Given a polynomial system $\mathbf{P} = \{f_0, f_1, \ldots, f_d\}$, let $\mathcal{A} = \cup_{i=0}^{d} S_{\mathbf{x}}(f_i)$, and view f_i as $f_i = \sum_{\alpha \in \mathcal{A}} c_{i,\alpha} \mathbf{x}^\alpha$, where some $c_{i,\alpha} = 0$ if $\alpha \notin S_{\mathbf{x}}(f_i)$. Then*

$$\theta(f_0, f_1, \ldots, f_d) = \prod_{i=1}^{d} \frac{1}{\overline{x}_i - x_i} \sum_{\substack{\sigma \subseteq \mathcal{A} \\ |\sigma| = d+1}} \begin{vmatrix} c_{0,\sigma_0} & c_{0,\sigma_1} & \cdots & c_{0,\sigma_d} \\ c_{1,\sigma_0} & c_{1,\sigma_1} & \cdots & c_{1,\sigma_d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,\sigma_0} & c_{d,\sigma_1} & \cdots & c_{d,\sigma_d} \end{vmatrix} \begin{vmatrix} \pi_0(\mathbf{x}^{\sigma_0}) & \pi_1(\mathbf{x}^{\sigma_0}) & \cdots & \pi_d(\mathbf{x}^{\sigma_0}) \\ \pi_0(\mathbf{x}^{\sigma_1}) & \pi_1(\mathbf{x}^{\sigma_1}) & \cdots & \pi_d(\mathbf{x}^{\sigma_1}) \\ \vdots & \vdots & \vdots & \vdots \\ \pi_0(\mathbf{x}^{\sigma_d}) & \pi_1(\mathbf{x}^{\sigma_d}) & \cdots & \pi_d(\mathbf{x}^{\sigma_d}) \end{vmatrix}.$$

Proof: The Dixon polynomial

$$\begin{aligned} \theta(f_0,\ldots,f_d) &= \prod_{i=1}^{d} \frac{1}{\overline{x}_i - x_i} \left| \begin{pmatrix} \pi_0(f_0) & \pi_0(f_1) & \cdots & \pi_0(f_d) \\ \pi_1(f_0) & \pi_1(f_1) & \cdots & \pi_1(f_d) \\ \vdots & \vdots & \ddots & \vdots \\ \pi_d(f_0) & \pi_d(f_1) & \cdots & \pi_d(f_d) \end{pmatrix}^T \right| \\ &= \prod_{i=1}^{d} \frac{1}{\overline{x}_i - x_i} \left| \begin{pmatrix} c_{0,\alpha_1} & c_{0,\alpha_2} & \cdots & c_{0,\alpha_n} \\ c_{1,\alpha_1} & c_{1,\alpha_2} & \cdots & c_{1,\alpha_n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,\alpha_1} & c_{d,\alpha_2} & \cdots & c_{d,\alpha_n} \end{pmatrix} \begin{pmatrix} \pi_0(\mathbf{x}^{\alpha_1}) & \pi_1(\mathbf{x}^{\alpha_1}) & \cdots & \pi_d(\mathbf{x}^{\alpha_1}) \\ \pi_0(\mathbf{x}^{\alpha_2}) & \pi_1(\mathbf{x}^{\alpha_2}) & \cdots & \pi_d(\mathbf{x}^{\alpha_2}) \\ \vdots & \vdots & \ddots & \vdots \\ \pi_0(\mathbf{x}^{\alpha_n}) & \pi_1(\mathbf{x}^{\alpha_n}) & \cdots & \pi_d(\mathbf{x}^{\alpha_n}) \end{pmatrix} \right| \end{aligned}$$

$$= \prod_{i=1}^{d} \frac{1}{\overline{x}_i - x_i} \sum_{\substack{\sigma \subseteq \mathcal{A} \\ |\sigma| = d+1}} \begin{vmatrix} c_{0,\sigma_0} & c_{0,\sigma_1} & \cdots & c_{0,\sigma_d} \\ c_{1,\sigma_0} & c_{1,\sigma_1} & \cdots & c_{1,\sigma_d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,\sigma_0} & c_{d,\sigma_1} & \cdots & c_{d,\sigma_d} \end{vmatrix} \begin{vmatrix} \pi_0(\mathbf{x}^{\sigma_0}) & \pi_1(\mathbf{x}^{\sigma_0}) & \cdots & \pi_d(\mathbf{x}^{\sigma_0}) \\ \pi_0(\mathbf{x}^{\sigma_1}) & \pi_1(\mathbf{x}^{\sigma_1}) & \cdots & \pi_d(\mathbf{x}^{\sigma_1}) \\ \vdots & \vdots & \vdots & \vdots \\ \pi_0(\mathbf{x}^{\sigma_d}) & \pi_1(\mathbf{x}^{\sigma_d}) & \cdots & \pi_d(\mathbf{x}^{\sigma_d}) \end{vmatrix}$$

(by Cauchy-Binet Formula).

3.1 Multiplier Sets for Polynomials

In this subsection, we show how appropriate multiplier sets can be obtained for each polynomial $f_i \in \mathbf{P}$ so as to construct a sparse resultant matrix.

Definition 3.3 *Define the Dixon polynomial θ_i corresponding to a polynomial f_i w.r.t. the polynomial system obtained by replacing a single polynomial $f_i \in \mathbf{P}$ by 1:*

$$\theta_i = \theta(f_0, \ldots, f_{i-1}, 1, f_{i+1}, \ldots, f_d).$$

Let $X_i = \{\mathbf{x}^\alpha | \mathbf{x}^\alpha \in \theta_i\}$, which serve as the multiplier set for f_i. $X_i f_i = \{\mathbf{x}^\alpha f_i | \mathbf{x}^\alpha \in X_i\}$, the new set of polynomials corresponding to f_i. We write

$$\mathbf{P'}_i \equiv X_i f_i = M_{\mathbf{P'}_i} \times {Y_i}^T,$$

where Y_i is the row vector corresponding to the set $\{\mathbf{x}^{\alpha+\beta} \mid \mathbf{x}^\alpha \in X_i \text{ and } \mathbf{x}^\beta \in f_i\}$, and $M_{\mathbf{P'}_i}$ is the coefficient matrix corresponding to polynomial system $\mathbf{P'}_i$.

Taking the union of polynomial sets $\mathbf{P'}_i$ and Y_i corresponding to each polynomial $f_i \in \mathbf{P}$ and the and concatnating the corresponding matrices, we get

$$\mathbf{P'} = \bigcup_{i=0}^{d} \mathbf{P'}_i = \begin{pmatrix} M_{\mathbf{P}'_0} \\ M_{\mathbf{P}'_1} \\ \vdots \\ M_{\mathbf{P}'_d} \end{pmatrix} \times Y = M_{\mathbf{P'}} \times Y,$$

where $Y = \bigcup_{i=0}^d Y_i$.

Henceforth, the matrix $M_{\mathbf{P'}}$ is called the **Dixon sparse resultant matrix** for the polynomial system $\mathbf{P}$.

3.2 Relationship between Dixon and Sparse Matrices

From the above construction, it is obvious that the Dixon matrix and Dixon sparse resultant matrix are intimately related. It can be shown that $\theta(f_0, \ldots, f_d) = \sum_{i=0}^d \theta_i f_i$ where $\theta_i = \theta(f_0, \ldots, f_{i-1}, 1, f_{i+1}, \ldots, f_d)$. To see this, subtract the k^{th} row from the $k+1^{th}$ row bottom up in the determinant for θ, and then expand it along the top row.

Below we abuse the notation by interchangeably using $X_i, \overline{X_i}, Y_i, X, \overline{X}, Y$ to stand for the sets of power products as well as row vectors.

Just like θ, $\theta_i = \overline{X_i} \Theta_i {X_i}^T$. Also, $\theta_i f_i = (\overline{X_i} \Theta_i {X_i}^T) f_i = \overline{X_i}(\Theta_i {X_i}^T f_i) = \overline{X_i}(\Theta_i \times M_{\mathbf{P'}_i}) {Y_i}^T$.

$$\begin{aligned} \theta(f_0, f_1, \ldots, f_d) &= \sum_{i=0}^{d} \theta_i f_i = \sum_{i=0}^{d} \overline{X_i}(\Theta_i \times M_{\mathbf{P'}_i}) Y_i^T \\ &= \overline{X} \left(\Theta_0 : \Theta_1 : \cdots : \Theta_d\right) \begin{pmatrix} M_{\mathbf{P'}_0} \\ M_{\mathbf{P'}_1} \\ \vdots \\ M_{\mathbf{P'}_d} \end{pmatrix} Y^T = \overline{X} \ (T \times M_{\mathbf{P'}}) \ Y^T = \overline{X} \Theta_{\mathbf{P}} Y^T \end{aligned}$$

where $\overline{X} = \bigcup_{i=0}^d \overline{X}_i$, $T = (\Theta_0 : \Theta_1 : \cdots : \Theta_d)$.

We thus have:

$$\Theta_{\mathbf{P}} = T \times M_{\mathbf{P}'}$$

We also have $\overline{X}\Theta X^T = \overline{X}\Theta_{\mathbf{P}}Y^T$. Note that $X \subseteq Y$, and therefore, Θ and $\Theta_{\mathbf{P}}$ are the same matrices except for $\Theta_{\mathbf{P}}$ having extra zero columns. Given $M_{\mathbf{P}'}$, the Dixon matrix Θ can be obtained from it by pre-multiplying by a transformation matrix T, and vice versa.

Proposition 3.2 *If Θ has an independent column q_i, then $M_{\mathbf{P}'}$ has the same independent column.*

Proof: Note that the columns of Θ are of the form $q_i = Tc_i$, where c_i is a column of $M_{\mathbf{P}'}$. By hypothesis, there exists q_i such that

$$q_i \neq \sum_{\substack{j=1\\ j\neq i}}^{k} \lambda_j q_j \qquad \text{for any} \qquad \lambda_j \in \mathbb{C},$$

but if we assume that $M_{\mathbf{P}'}$ has no independent column, then for all columns c_i,

$$c_i = \sum_{\substack{j=1\\ j\neq i}}^{k} \lambda_j c_j,$$

then

$$q_i = Tc_i = \sum_{\substack{j=1\\ j\neq i}}^{k} \lambda_j T c_j = \sum_{\substack{j=1\\ j\neq i}}^{k} \lambda_j q_j,$$

contradicting the hypothesis.□

3.3 Illustration: Univariate case revisited

Let

$$\begin{aligned} f_0 &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{m-1}x^{m-1} + a_m x^m, \\ f_1 &= b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-1}x^{n-1} + b_n x^n. \end{aligned}$$

For the polynomial system $\{f_0, f_1\} \subset \mathbb{C}[x]$, the Dixon formulation is better know as Bezoutian. Here in the expression $\Theta = T \times M$, we will note that M is the well known Sylvester matrix and Θ is the Bezout Matrix. The Sylvester matrix for $\{f_0, f_1\}$ is given by

$$R = \begin{array}{c} n\left\{\begin{array}{c} \\ \\ \\ \\ \end{array}\right. \\ m\left\{\begin{array}{c} \\ \\ \\ \\ \end{array}\right. \end{array} \left|\begin{array}{cccccccc} a_0 & a_1 & \cdots & a_m & & & & \\ & a_0 & a_1 & \cdots & a_m & & & \\ & & & & \ddots & \ddots & & \\ & & & & & a_0 & a_1 \cdots & a_m \\ b_0 & b_1 & \cdots & b_n & & & & \\ & b_0 & b_1 & \cdots & b_n & & & \\ & & & & \ddots & \ddots & & \\ & & & & & b_0 & b_1 \cdots & b_n \end{array}\right|$$

and $T = (\Theta_0 : \Theta_1)$ where $\theta_0 = \theta(1, f_1) = \overline{X}_0^T \Theta_0 X_0$.

$$\theta_0 = \theta(1, f_1) \quad = \quad \frac{1}{\overline{x} - x}\begin{vmatrix} 1 & f_1(x) \\ 1 & f_1(\overline{x}) \end{vmatrix} = \frac{f_1(x) - f_1(\overline{x})}{\overline{x} - x} = b_1 S_1 + b_2 S_2 + \cdots + b_n S_n$$

where $S_k = \sum_{i=1}^{k} \overline{x}^{i-1} x^{k-i}$.

Note that $X_0 = \{1, x, \ldots, x^{n-1}\}$ and $X_1 = \{1, x, \ldots, x^{m-1}\}$ as in the Sylvester matrix construction. From this, it can be seen that Θ_0 matrix is $n \times n$, and hence, similarly Θ_1 will be $m \times m$, and the matrix T is of size $\max\{m,n\} \times (n+m)$. For the univariate case we can write down matrix T as

$$T = \begin{pmatrix} & & & & & & & & a_m \\ & & & b_n & & & & a_m & a_{m-1} \\ & & b_n & b_{n-1} & & & & & \\ & & & \vdots & & & & & \vdots \\ & b_n & \cdots & b_3 & b_2 & & a_m & \cdots & a_3 & a_2 \\ b_n & b_{n-1} & \cdots & b_2 & b_1 & a_m & a_{m-1} & \cdots & a_2 & a_1 \end{pmatrix}$$

It can be seen from the above construction that when $m > n$, $\det \Theta$ will have the extra factor of a_m^{m-n}, which comes from matrix T. This can be easily verified using *Cauchy-Binet*-formula for determinant of a product of non-square matrices. [AW92]

3.4 Relating Sizes of the Dixon Sparse Matrix and Dixon Matrix

Let $Size(\Theta)$ denote either the minimum number of non-zero rows or non-zero columns. For purposes of comparing matrix sizes, assume that $\mathbf{P}$ is **unmixed,** i.e., every polynomial in $\mathbf{P}$ has the same support.

In the derivation, the reader should note that in the unmixed case, $\overline{X}_i = \overline{X}_j = \overline{X}$, which implies that number of non-zero rows of Θ is $\leq Size(\Theta_i)$

Proposition 3.3 *If* $\mathbf{P}$ *is a generic unmixed system where* $\mathcal{A} = S_{\mathbf{x}}(P)$ *and* $\mathbf{0} \in \mathcal{A}$*, then*

$$\#rows(\Theta) = \#rows(\Theta_i).$$

Proof: As discussed above,

$$\theta = \prod_{i=0}^{d} \frac{1}{\overline{x}_i - x_i} \sum_{\substack{\sigma \subseteq \mathcal{A} \\ |\sigma|=d+1}} \begin{vmatrix} c_{0,\sigma_0} & c_{0,\sigma_1} & \cdots & c_{0,\sigma_d} \\ c_{1,\sigma_0} & c_{1,\sigma_1} & \cdots & c_{1,\sigma_d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,\sigma_0} & c_{d,\sigma_1} & \cdots & c_{d,\sigma_d} \end{vmatrix} \begin{vmatrix} \pi_0(\mathbf{x}^{\sigma_0}) & \pi_0(\mathbf{x}^{\sigma_1}) & \cdots & \pi_0(\mathbf{x}^{\sigma_d}) \\ \pi_1(\mathbf{x}^{\sigma_0}) & \pi_1(\mathbf{x}^{\sigma_1}) & \cdots & \pi_1(\mathbf{x}^{\sigma_d}) \\ \vdots & \vdots & \ddots & \vdots \\ \pi_d(\mathbf{x}^{\sigma_0}) & \pi_d(\mathbf{x}^{\sigma_1}) & \cdots & \pi_d(\mathbf{x}^{\sigma_d}) \end{vmatrix}.$$

Consider a $\sigma = \{\sigma_0, \ldots, \sigma_d\}$, that is $\sigma \subseteq \mathcal{A}$. For θ_i, σ is chosen from $\mathcal{A} \cup \{\mathbf{0}\}$, and for θ, it is chosen from $\mathcal{A}$. But since $\mathbf{0} \in \mathcal{A}$, the monomial sets must be equal. The only difference is that for θ_i, $\sigma_i = \mathbf{0}$, and for θ, there is no such restriction. Since we are only concerned with monomials in $\{\overline{x}_1, \ldots, \overline{x}_d\}$, the first row of θ is effectively 1's. Hence any monomial of θ_i is a monomial of θ. □

Note that the condition that $\mathbf{0} \in \mathcal{A}$ will force toric and affine resultants to be the same.

Corollary 3.3.1 *If* $\mathbf{0} \in \mathcal{A}$*,* $\#rows(M_{\mathbf{P}'}) = (d+1)\#rows(\Theta)$*, where* $M_{\mathbf{P}'}$ *is the sparse matrix.*

For the unmixed case, $M_{\mathbf{P}'}$ is bigger by a factor of $(d+1)$. For mixed polynomial systems, $M_{\mathbf{P}'}$, however, is smaller, that is the ratio in the sizes is not as big as $(d+1)$. It is even possible to construct examples where both the matrices have almost the same size. This is so because the size of the Dixon matrix Θ is usually as big as $\max Size(\Theta_i)$.

It has been already shown in [KS96] that the size of Dixon matrix is bounded from above by $Vol(\pi_0(\mathcal{A}) + \pi_1(\mathcal{A}) + \cdots + \pi_{d-1}(\mathcal{A}))$, i.e. the volume of the Minkowski sum of the projections of Newton polytopes, which is much closer to mixed volume i.e. lower bound. This suggests that the proposed Dixon sparse matrix construction has even a better upper bound as compared to the toric sparse matrix constructions.

4 Complexity of Constructing Dixon Sparse Matrices

Proposition 4.1 *Let* $P = \{f_0, \ldots, f_d\}$ *and* $m = |\cup_{i=0}^{d} \mathcal{A}_i|$ *where* $\mathcal{A}_i = S_{\mathbf{x}}(f_i)$*, then the complexity of constructing* $M_{P'}$ *is*

$$T_A = O\left(\frac{m!(d+1)^2}{(m-d)!}\right) = O(d^2 m^d).$$

Proof Assume (in the worst case) that each support has m points, therefore each θ_i has the same structure, except for different permutation of columns. Hence, the total complexity is bounded from above by $(d+1)$ times the complexity of expanding single θ_i, w.l.o.g. assume θ_0.

$$\begin{aligned}\theta_0 &= \theta(1, f_1, \ldots, f_d) = \\ &= \prod_{i=0}^{d} \frac{1}{\overline{x}_i - x_i} \sum_{\sigma \subseteq \mathcal{A}} \begin{vmatrix} 1 & 0 & 0 & \cdots & 0 \\ c_{1,0} & c_{1,\sigma_1} & c_{1,\sigma_2} & \cdots & c_{1,\sigma_d} \\ c_{2,0} & c_{2,\sigma_1} & c_{2,\sigma_2} & \cdots & c_{2,\sigma_d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{d,0} & c_{d,\sigma_1} & c_{d,\sigma_2} & \cdots & c_{d,\sigma_d} \end{vmatrix} \begin{vmatrix} 1 & \pi_0(\mathbf{x}^{\sigma_1}) & \pi_0(\mathbf{x}^{\sigma_2}) & \cdots & \pi_0(\mathbf{x}^{\sigma_d}) \\ 1 & \pi_1(\mathbf{x}^{\sigma_1}) & \pi_1(\mathbf{x}^{\sigma_2}) & \cdots & \pi_1(\mathbf{x}^{\sigma_d}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \pi_d(\mathbf{x}^{\sigma_1}) & \pi_d(\mathbf{x}^{\sigma_2}) & \cdots & \pi_d(\mathbf{x}^{\sigma_d}) \end{vmatrix}\end{aligned}$$

where $\sigma_i \in \mathcal{A}_i$, for $i \in \{1, \ldots, d\}$ (note $\sigma_0 = \mathbf{0}$) and $\sigma_i \neq \sigma_j$ for $i \neq j$. Hence there are $\binom{m}{d}$ terms in this sum, where we need to expand 2 determinants of size $(d+1)$. In the worst case, it will take $2(d+1)!$ time. It is not necessary to carry out division by $\overline{x}_i - x_i$, as the resulting monomials can be easily deduced. This is analogous to the Sylvester matrix construction, where given the degrees of the polynomials, any entry in the matrix can be deduced in constant time. $\square$

Note that the subdivision and incremental methods have, respectively, complexities as

$$T_S = O\left(Size(M) d^{9.5} m^{6.5} \log^2 k \log^2 \frac{1}{\epsilon_l \epsilon_\delta}\right),$$

and

$$T_I = O^*\left(e^{3d} m^{5.5} (\deg R)^3\right) + O^*\left(d^{7.5} m^{2d+5.5}\right),$$

where k is the max degree of polynomials in the system, ϵ_l is the probability of failure to pick generic lifting vector, and ϵ_δ is the probability of perturbation failure [CE96]. In the above formula, $Size(M) = \Omega(\deg R)$ and $\deg R = \Omega(m^d)$. With these crude lower bounds for the size of the resultant matrix and the degree of the resultant itself, we can compare the methods.

$$\frac{T_S}{T_A} = O\left(d^{7.5} m^{6.5} \log^2 k \log \frac{1}{\epsilon_l \epsilon_\delta}\right),$$

and

$$\frac{T_I}{T_A} = O\left(e^{3d} m^{5.5} \deg R^2\right) + O\left(d^{5.5} m^{d+5.5}\right).$$

The experimental results below confirm that the proposed method is an order of magnitude faster than the subdivision and incremental methods. The tables below discuss a diverse set of examples from different application domains. As is clear from the table, the proposed method can easily and quickly construct the Dixon sparse matrices, whereas the subdivision and incremental methods can take considerable time in construction itself.

As the tables below suggest, the proposed method also constructs smaller matrices if coefficients are not generic, implying that it depends not only on the support of a polynomial system but also on the actual coefficients. This can be advantageous for applying the algorithm to problems arising from application domains. Note that this is not true for the subdivision and incremental methods; in fact non-generic coefficients might cause them to construct even bigger matrices.

Table 1: Problems definitions (see `http://www.cs.unm.edu/~artas/ET` for a detailed description of the examples)

#	**Problem**	d	**Bezout Bound**	**Mixed Volume**	$\deg R_{\mathbb{T}}^{\dagger}$	**Mx Vol Affine**	$\deg R_{\mathbb{A}}^{\dagger}$
1.	Emiris Example	2	8,6,12	4,3,4	11	4,3,4	11
2.	Cyclic roots $n = 3$	2	4,2,2	2,2,2	6	2,2,2	6
3.	Cyclic roots $n = 4$	3	18,9,6,6	5,6,5,4	20	5,6,5,5	21
4.	Cyclic roots $n = 5$	4	96,48,32,24,24	16,18,18,16,14	82	16,18,18,16,18	86
5.	Cyclic roots $n = 6$	5	600,300,200, 150,120,120	46,58,56, 58,46,26	290	46,58,56, 58,46,52	316
6.	Sum of squares $n = 2$	2	2,2,4	2,2,4	8	2,2,4	8
7.	Sum of squares $n = 3$	3	4,4,4,8	4,4,4,8	20	4,4,4,8	20
8.	Sum of squares $n = 4$	4	8,8,8,8,16	8,8,8,8,16	48	8,8,8,8,16	48
9.	Sum of squares $n = 5$	5	16,16,16 16,16,32	16,16,16, 16,16,32	112	16,16,16, 16,16,32	112
10.	Sum of squares $n = 6$	6	32,32,32,32, 32,32,64	32,32,32,32, 32,32,64	256	32,32,32,32, 32,32,64	256
11.	Max Volume of Tetrahedron	4	24,24,24,24,16	9,9,9,9,8	44	9,9,9,9,8	44
12.	Distance of intersection of conics from origin	2	4,4,4	4,4,4	12	4,4,4	12
13.	Condition of perpendicular intersection of conic and circle	2	4,4,4	4,4,4	12	4,4,4	12
14.	Implicitization of a sphere	2	16,8,8	8,4,4	16	12,4,4	20
15.	Implicitization of bicubic surface	2	18,18,9	18,18,9	45	18,18,9	45
16.	Implicitization of dense bicubic surface	2	36,36,36	18,18,18	54	18,18,18	54
17.	Cubic surface implicitization	2	9,9,9	9,9,9	27	9,9,9	27
18.	Equilibrium of Lorentz system	3	4,4,8,4	3,4,5,3	15	3,4,5,3	15
19.	Camera motion from point matches	5	32,32,32,32,32,32	6,6,6,6,6,6	36	6,6,6,6,6,6	36
20.	Conformal analysis of cyclic molecules	3	16,16,16,64	12,12,12,16	52	12,12,12,16	52
21.	Stewart platform q_1 is a parameter	6	64,64,64,64, 64,64,64	22,32,32,32, 32,32,32	214	22,32,32,32, 32,32,32	214
22.	Stewart platform x_3 is a parameter	6	64,64,64,64, 64,64,64	42,52,52,52, 52,52,52	354	42,52,52,52, 52,52,52	354

† - an upper bound on the degree, since the exact number of roots in most cases are not known.

The table below contains data of our run on benchmark examples, see table 1. All three methods are implemented in Maple. The subdivision implementation is provided by Canny and Emiris. The incremental and the proposed method are implemented by the authors, where incremental algorithm is of [EC95].[2]

Table 2: Comparison of the three methods on examples. The blanks in the table means that we were unable to obtain the answer with our implementation. The column labeled "Extra" in the above table estimates a lower bound on the degree of the extraneous factor, since the actual number of roots, and hence, the degree of the resultant can be smaller.

#	**Subdivision Matrix**					**Incremental Matrix**					**Dixon Sparse Matrix**				
	Size	Rank	Extra	# Mult	Time	Size	Rank	Extra	# Mult	Time	Size	Rank	Extra	# Mult	Time
1	14×14	14	3	4,4,6	10.29	12×12	12	1	5,3,4	12.41	12×12	12	1	5,3,4	0.08
2	6×6	6	0	2,2,2	3.18	8×7	7	1	3,2,3	4.04	6×6	6	0	2,2,2	0.05
3	25×25	25	5	5,7,8,5	10.90						20×22	18	-2	4,6,4,6	0.01
4	144×144	144	62	15,26,33,32,38	187.35						90×92	90	4	16,18,18,16,22	0.04
5											408×412	402	86	57,68,72,68,57,86	0.75
6	9×9	9	1	2,3,4	3.39	8×8	8	0	2,2,4	6.61	8×8	8	0	2,2,4	0.03
7	20×20	20	0	4,4,4,8	12.08	20×20	20	0	4,4,4,8	72.13	20×20	20	0	4,4,4,8	0.06
8	48×48	48	0	8,8,8,8,16	35.13	48×48	48	0	8,8,8,8,16	762.23	48×48	48	0	8,8,8,8,16	0.18
9	136×136	136	24	16,16,16,40,16,32	103.08						112×112	112	0	16,16,16,16,16,32	0.08
10	280×280	280	24	32,32,48,40,32,32,64	577.11						256×256	256	0	32,32,32,32,32,32,64	0.39
11	102×102	102	58	9,18,20,26,29	166.64						60×63	60	16	11,11,12,16,10	0.42
12	14×14	14	2	4,5,5	7.95	16×15	15	3	6,6,4	17.06	15×14	14	2	5,5,5	0.04
13	21×21	21	9	4,7,10	14.41	16×15	15	3	6,6,4	14.06	13×14	12	0	$5,3,5^*$	0.04
14	24×24	24	8	8,8,8	14.88	16×16	16	0	8,4,4	20.38	20×20	20	0	12,4,4	0.16
15	64×64	64	19	18,25,21	78.93	110×77	77	32	41,41,28	74.89	45×45	45	0	18,18,9	0.28
16	72×72	72	18	18,27,27	154.24						54×54	54	0	18,18,18	0.75
17	38×38	38	11	9,15,14	26.53	39×33	33	6	12,15,12	27.76	28×28	28	1	9,10,9	0.09
18	23×23	23	8	3,5,7,8	19.98	19×17	17	2	4,4,6,5	134.62	16×16	16	1	3,5,5,3	0.07
19											36×36	36	0	$6,6,6,6,6^*$	0.92
20	108×108	108	56	12,16,24,56	112.01						86×84	81	29	15,21,18,32	0.07
21											659×454	437	223	27,105,105,105,105,105,107	9.54
22											1328×924			74,207,207,207,207,207,219	25.75

* - matrix is bigger for generic coefficients.

[2] Emiris in private communication has promised to provide us with an implementation of his incremental algorithm. As soon as that implementation is available, we will update our tables based on his improved implementation.

As the table 2 shows, the incremental algorithm constructs smaller matrices than the subdivision algorithm. However, the incremental algorithm is slower (at least our implementation) than the subdivision algorithm, and on some examples, it fails to construct a sparse matrix. The incremental algorithm, however, behaves very well on multi-homogeneous and dense systems, constructing either exact or almost-exact matrices.

4.1 Relating Sizes of the Dixon and Newton Sparse Matrices

We show that the lower bound on the size of the Dixon sparse matrix is bigger than the lower bound on sparse matrices for toric varieties. This is so because the underlying affine variety can be bigger than the associated toric variety of a polynomial system. Theoretically, one should not expect too much of a difference, and the experimental results confirm this.

Let $\mathbf{P}_i = \{f_0, \ldots, f_{i-1}, f_{i+1}, \ldots, f_d\}$. Given that $\mathbf{P}$ has no solutions in the case of symbolic coefficients, $\mathbf{P}_i$ must have a finite number of solutions for each i, as $\mathbf{P}_i$ defines a zero dimensional ideal.

Theorem 4.1 *[PS93]*

$$R = \delta \prod_{\varepsilon} f_i(\varepsilon) \qquad \textit{where} \qquad \varepsilon \textit{ is solution of } P_i = 0,$$

and δ is rational function in the coefficients of P_i. As a consequence,

$$R = Res(f_0, \ldots, \lambda f_i, \ldots, f_d) = \lambda^k Res(f_0, \ldots, f_i, \ldots, f_d)$$

where k = # of roots of P_i.

The polynomial R is separately homogeneous in the coefficients of f_i with degree equal to number of solutions of P_i. This puts a lower bound on the size of any Newton matrix. Therefore we are interested on some bound on the number of roots of P_i.

Given that the goal is to get the condition for the existence of a root in $(\mathbb{C}^*)^d$, the size of a Newton matrix must be at least

$$\sum_{i=1}^{d} \mu(Q_0, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_d).$$

We also have similar result about the number of solution P has in $\left(\mathbb{C}^d\right)^*$. Let $Q_i' = Q_i \cup \mathbf{0}$, then the number of solutions P has in $\left(\mathbb{C}^d\right)^*$ is equal to $\mu(Q_1', \ldots, Q_d')$ for most choices of the coefficients. See [LW96] and [Roj99]. Therefore, in this case, the matrix size must be at least

$$\sum_{i=1}^{d} \mu(Q_0', \ldots, Q_{i-1}', Q_{i+1}', \ldots, Q_d').$$

When all Newton polytopes Q_i are the same, i.e., P is unmixed, then

$$\mu(Q_1, Q_2, \ldots, Q_d) = d! Vol(Q).$$

Hence, a difference in lower bound on the size of a sparse affine matrix and a sparse toric matrix is at most

$$(d+1)!(Vol(Q \cup \mathbf{0}) - Vol(Q)).$$

The above difference is determined by how close Newton polytopes are lying near the origin. Note that the polytope Q_i is almost cornered at the origin, i.e., it "touches" all axis hyper-planes except possible one, since otherwise the system will not be sufficiently constrained. For this reason, in most practical cases, $Vol(Q \cup \mathbf{0}) - Vol(Q)$ is much smaller than $Vol(Q)$, and hence, the advantage of looking just at the toric matrices is almost insignificant.

5 Conclusion

Sparse matrices have proved to be quite cumbersome and difficult if constructed explicitly from the support by taking advantage of the sparseness of polynomial systems. These constructions suffer from high run-time complexity observed in practice, as well as their theoretical complexity is quite high. We have given a simple and efficient construction for sparse resultant matrices from a classical technique introduced by Cayley and Dixon. It settles an open question in our research that had been lingering since 1997 when we were able to show that the generalized Dixon resultant formulation implicitly exploited the sparse structure of a polynomial system. The proposed method is easier to implement as well as faster in contrast to the incremental as well as the subdivision algorithms.

The usefulness of sparse matrices for computing projection operators for polynomial systems is still overshadowed by the generalized Dixon resultant method which has turned out to be superior in performance, both in theory and practice [KS95]. The Dixon matrix is smaller by a factor of the dimension, and therefore extracting a maximal minor from the Dixon matrices is easier. We, however, believe that sparse matrices are a useful tool for studying the source of extraneous factors in projection operators computed using the generalized Dixon resultant formulations. As a matter of fact, the proposed method for constructing sparse matrices is a by-product of our investigations into a priori identifying extraneous factors in the computation of projection operators based on Dixon matrices.

References

[AW92] W.A. Adkins and S.H. Weintraub. *Algebra, An Approach via Module Theory.* Springer-Verlag, New York, 1992.

[Can90] J. Canny. Generalized characteristic polynomials. *Journal of Symbolic Computation*, 9:241–250, 1990.

[CE93] J. Canny and I. Emiris. An efficient algorithm for the sparse mixed resultant. In G. Cohen, T. Mora, and O. Moreno, editors, *Proc. Intern. Symp. Applied Algebra, Algebraic Algor. and Error-Corr. Codes, Lect. Notes in Comp. Science 263*, pages 89–104, Puerto Rico, May 1993. Springer Verlag.

[CE96] J.F. Canny and I.Z. Emiris. A subdivision based algorithm for the sparse resultant. *submitted to J. ACM*, 1996.

[CLO96] D. Cox, J. Little, and D. O'Shea. *Ideals,Varieties, and Algorithms.* Springer-Verlag, New York, second edition, 1996.

[CLO98] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry.* Springer-Verlag, New York, first edition, 1998.

[Dix08] A.L. Dixon. The eliminant of three quantics in two independent variables. *Proc. London Mathematical Society*, 6:468–478, 1908.

[DKM92] B.R. Donald, D. Kapur, and J.L Mundy, editors. *Symbolic and Numerical Computation for Artificial Inteligence.* Academic Press, 1992.

[EC95] I.Z. Emiris and J.F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symbolic Computation*, 20(2):117–149, August 1995.

[Emi94] I. Emiris. *Sparse Elimination and Applications in Kinematics.* PhD thesis, Department of Computer Science, Univeristy of Calif., Berkeley, 1994.

[GKZ94] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants.* Birkhauser, Boston, first edition, 1994.

[Hof89] C.M Hoffman. *Geometric and Solid modeling*. Morgan Kaufmann Publishers, Inc., San Mateo, California 94403, 1989.

[KS95] D. Kapur and T. Saxena. Comparison of various multivariate resultants. In *ACM ISSAC 95*, Montreal, Canada, July 1995.

[KS96] D. Kapur and T. Saxena. Sparsity considerations in the Dixon resultant formulation. In *Proc. ACM Symposium on Theory of Computing*, Philadelphia, May 1996.

[KSY94] D. Kapur, T. Saxena, and L. Yang. Algebraic and geometric reasoning using Dixon resultants. In *ACM ISSAC 94*, pages 99–107, Oxford, England, July 1994.

[LW96] T.Y. Li and X. Wang. The BKK root count in $\mathbb{C}^n$. *Mathematics of computation*, Oct. 1996.

[Mac16] F.S. Macaulay. The algebric theory of modular systems. *Cambridge Tracts in Math. and Math. Phys.*, 19, 1916.

[PS93] P.Pedersen and B. Sturmfels. Product formulas for resultants and chow forms. *Math. Zeitschrift*, 214:377–396, 1993.

[Roj99] J.M. Rojas. Toric intersection theory for affine root counting. *Journal of Pure and Applied algebra*, June 1999.

[Sax97] T. Saxena. *Efficient variable elimination using resultants*. PhD thesis, Department of Computer Science, State Univeristy of New York, Albany, NY, 1997.

[Stu91] B. Sturmfels. Sparse elimination theory. In D. Eisenbud and eds. L. Robbiano, editors, *Proc. Computat. Algebraic Geom. and Commut. Algebra*, Cortona,Italy, June 1991. Cambridge Univ. Press.