A new interpolation algorithm for computing Dixon resultants

Ayoola Jinadu & Michael Monagan
Department of Mathematics, Simon Fraser University
{ajinadu,mmonagan}@sfu.ca

Introduction

We study **Dixon resultants**[2], a determinant approach to eliminate $n-1$ variables from a parametric polynomial system with n variables by taking the determinant of the Dixon matrix with polynomial entries. To the best of our knowledge, the Dixon resultant method is the most efficient and practical method of all known resultant methods.

Gröbner basis and Triangular set methods in Maple and Magma can be used to solve polynomial systems but experiments have shown that these methods may fail. In particular, these methods fail on the polynomial systems involving many parameters listed in [6, 7]. The failure of these methods on parametric polynomial systems is due to the intermediate expression swell caused by the parameters.

We have designed and implemented a new interpolation algorithm [3] for computing Dixon resultants which performed significantly better than Zippel's sparse interpolation when we tried our code on polynomial systems from [6, 7]. The only interpolation method that has been applied to Dixon resultants that we are aware of was done by Kapur and Saxena in [5]. They used Zippel's sparse interpolation [8] to interpolate the Dixon resultant R.

What are we computing?

Let $X=\{x_1,x_2,\cdots,x_n\}$ denote the set of variables and let $Y=\{y_1,y_2,\cdots,y_m\}$ be the set of parameters with $n\geq 2$ and $m\geq 0$. Let $\mathcal{F}=\{f_1,f_2,\cdots,f_n\}\subset\mathbb{Q}[X,Y]$ be a parametric polynomial system where f_i is a polynomial in variables X with coefficients in $\mathbb{Q}[Y]$. Let $I=\langle f_1,f_2,\cdots,f_n\rangle$ be the ideal generated by $\mathcal{F}$. The Dixon resultant R of $\mathcal{F}$ in x_1 is the determinant of the Dixon matrix and it is a polynomial in the elimination ideal $I\cap\mathbb{Q}[Y][x_1]$.

Example

Let $\mathcal{F}=\{x_2^2+x_3^2-y_3^2,\ (x_2-y_1)^2+x_3^2-y_2^2,\ -x_3y_1+2x_1\}$ with variables $X=\{x_1,x_2,x_3\}$ and parameters $Y=\{y_1,y_2,y_3\}$. The Dixon matrix D for the above polynomial system is

$$D=\begin{bmatrix}-2y_1^2 & 0 & y_1^3-y_1y_2^2+y_1y_3^2\\ 0 & -2y_1^2 & 4x_1y_1\\ y_1^3-y_1y_2^2+y_1y_3^2 & 4x_1y_1 & -2y_1^2y_3^2\end{bmatrix}$$

and its determinant which is the Dixon resultant

$$R=\det(D)=2y_1^4(16x_1^2+y_1^4-2y_1^2y_2^2-2y_1^2y_3^2+y_2^4-2y_2^2y_3^2+y_3^4).$$

Let $R=\sum_{k=0}^{d} r_k(y_1,\cdots,y_m)x_1^k\in\mathbb{Q}[Y][x_1]$ be the Dixon resultant of $\mathcal{F}$ in x_1 where $d=\deg(R,x_1)>0$. Let $C=\gcd(r_0,\cdots,r_d)$ be the polynomial content of R. In our paper [3], we compute **the monic square-free factors of R** and **NOT** R.

The **monic square-free factorization of** R is a factorization of the form $\hat{r}\prod_{j=1}^{l}R_j^j$ such that

1. $\hat{r}=C/L$ for some $L\in\mathbb{Q}[Y]$,
2. each R_j is monic and square-free in $\mathbb{Q}(Y)[x_1]$, i.e., $\gcd(R_j,R_j')=1$, and
3. $\gcd(R_i,R_j)=1$ for $i\neq j$.

This monic square-free factorization exists and it is unique. We view

$$R_j=x_1^{d_{T_j}}+\sum_{k=0}^{T_j-1}\frac{f_{jk}(y_1,y_2,\cdots,y_m)}{g_{jk}(y_1,y_2,\cdots,y_m)}x_1^{d_{jk}}\in\mathbb{Q}(y_1,y_2,\cdots,y_m)[x_1]$$

where $\gcd(f_{jk},g_{jk})=1$, $f_{jk},g_{jk}\in\mathbb{Q}[y_1,y_2,\cdots,y_m]$ and $d_{T_j}\leq\deg(R,x_1)$. Note, the factors R_j are not necessarily irreducible over $\mathbb{Q}$. We give the following real example to illustrate what we are computing.

Example

Let

$$C=\underbrace{-65536\left(al^2+1\right)^8 l_2^8\left(al^2l_2^2+2al^2l_2l_3+al^2l_3^2+l_2^2-2l_2l_3+l_3^2\right)^4}_{\textbf{polynomial content}}$$

$$\begin{aligned}A_1&=t_1^2+1\\ A_2&=(al^2l_1^2+2al^2l_1x-al^2l_2^2-2al^2l_2l_3-al^2l_3^2+al^2x^2+al^2y^2+l_1^2+2l_1x-l_2^2\\ &\quad+2l_2l_3-l_3^2+x^2+y^2)t_1^2+\left(-4al^2l_1y-4l_1y\right)t_1+al^2l_1^2-2al^2l_1x-al^2l_2^2\\ &\quad-2al^2l_2l_3-al^2l_3^2+al^2x^2+al^2y^2+l_1^2-2l_1x-l_2^2+2l_2l_3-l_3^2+x^2+y^2\\ A_3&=(aa^2+2aal_2)t_1^2+aa^2-4aal_1+2aal_2+4l_1^2-4l_1l_2\\ A_4&=(aa^2-2aal_2)t_1^2+aa^2-4aal_1-2aal_2+4l_1^2+4l_1l_2\end{aligned}$$

where $X=\{t_1,t_2,b_1,b_2\}$ are the **variables**, t_1 is the main variable and $Y=\{aa,al,l_1,l_2,l_3,x,y\}$ are the **parameters**. The Dixon resultant R of the robot arms system in t_1 has **6,924,715** terms and it factors as $CA_1^{24}A_2^4A_3^2A_4^2$.

Our **new Dixon resultant algorithm** computes $R_1=A_1$, $R_2=\mathrm{monic}(A_2,t_1)$ and $R_3=\mathrm{monic}(A_3A_4,t_1)$. The largest coefficient of R_1, R_2 and R_3 is the leading coefficient of A_2 which has only **14** terms! Notice that R_1 and R_2 are irreducible over $\mathbb{Q}$ but R_3 is not.

We note that the constructed Dixon matrix D of a polynomial system $\mathcal{F}$ can be **singular or rectangular** which means that we have no information about the solutions of the system. If this happens, one simply needs to extract a maximal minor M of D and then compute $R=\det(M)$. This idea is due to Kapur, Saxena and Yang [4].

Our new algorithm

Let M be a maximal minor of rank t of a Dixon matrix D. In this work, we use a **black box** to represent $\det(M)$. This black box representation assumes that $\det(M)$ is unknown. To be explicit, our black box is a C code which takes as input a prime p and $\alpha\in\mathbb{Z}_p^t$ and outputs $\det(M(\alpha))\in\mathbb{Z}_p$.

Our **new Dixon resultant algorithm** probes the black box to compute the monic square-free factors R_j of R from monic univariate images in x_1 using sparse multivariate rational function interpolation to interpolate the coefficients of R_j in $\mathbb{Q}(Y)$ modulo primes and uses Chinese remaindering and rational number reconstruction to recover the rational coefficients of R_j.

We use the sparse rational function interpolation algorithm of Cuyt and Lee [1] with the Ben-Or/Tiwari polynomial algorithm for this purpose. In order to avoid unlucky evaluation points with high probability and reduce the size of our primes for machine arithmetic use, we have modified the Cuyt and Lee's algorithm to use **Kronecker substitution**. Thus, we interpolate the mapped function

$$K_r(R_j)=x_1^{d_{T_j}}+\sum_{k=0}^{T_j-1}\frac{f_{jk}(y,y^{r_1},y^{r_1r_2},\cdots,y^{r_1r_2\cdots r_{m-1}})}{g_{jk}(y,y^{r_1},y^{r_1r_2},\cdots,y^{r_1r_2\cdots r_{m-1}})}x_1^{d_{jk}}\in\mathbb{Q}(y)[x_1]$$

with a Kronecker substitution $K_r:\mathbb{Q}(y_1,\cdots,y_m)[x_1]\rightarrow\mathbb{Q}(y)[x_1]$ such that for $1\leq i\leq m-1$, each $r_i>\max_{j=1}^{l}(\max_{k=0}^{T_j-1}(\deg(f_{jk},y_i),\deg(g_{jk},y_i)))$. Inverting the Kronecker map K_r yields the R_j's.

Although the degree of the mapped rational function $K_r(R_j)$ is exponential in y, the degree of many univariate rational functions in a new variable z with Kronecker substitution K_r through which $K_r(R_j)$ is interpolated remains the same. Consequently, the number of terms and the number of black box probes needed to interpolate R_j does not change. To recover the exponents in y we require prime $p>\prod_{i=1}^{m}r_i$.

Interpolating the R_j's instead of R results in a huge gain because all **unwanted repeated factors and the polynomial content are removed.** The **advantage of our new Dixon resultant algorithm** over other known polynomial interpolation algorithms is that the number of polynomial terms in R_j to be interpolated is much less than in R and the number of primes used by our algorithm in the sparse interpolation step when we apply the Chinese remainder theorem is reduced.

Experiments

We have implemented our new Dixon resultant algorithm in **Maple** with some parts coded in **C** for efficiency. In Table 1, we present basic information about the real polynomial systems we tried our code on. The $\dim(D)$ and the rank of its maximal minor M are in column 4. The number of terms in the product of all the monic square-free factors in expanded form when the denominators are cleared is denoted by $\#S$ and $t_{\max}=\max(\#f_{jk},\#g_{jk})$. In column 8 named as DRes, we report the timings of our Dixon resultant algorithm. The timings of a hybrid implementation of Zippel's sparse algorithm in Maple + C for interpolating R in expanded form are given in column 9. DRes-Probe denotes the number of black box probes needed by our Dixon resultant algorithm and Zippel-Probe represents the number of black box probes needed by Zippel's algorithm to interpolate R in **expanded form.**

Table: Timings for our new algorithm labelled DixonRes versus Zippel's Interpolation

Systems	**#Equations**	n/m	dim D/**Rank**	$\#S$	$t_{\max}$	$\#R$	**DRes**	**Zippel**	DRes-Probe	Zippel-Probe
Robot-t_1	4	4/7	$(32\times 48)/20$	450	14	6924715	7.34s	$>10^5$s	16641	-
Robot-t_2	4	4/7	$(32\times 48)/20$	13016	691	16963876	316.99s	$>10^5$s	711481	-
Robot-b_1	4	4/7	$(32\times 48)/20$	334	85	6385205	27.78s	$>10^5$s	94901	-
Robot-b_2	4	4/7	$(32\times 48)/20$	11737	624	16801877	241.61s	$>10^5$s	535473	-
Heron5d	15	14/16	$(707\times 514)/399$	823	822	12167689	23.12s	$>10^5$s	63235	-
Flex-v1	3	3/15	$(8\times 8)/8$	5685	2481	45773	201s	308684.76s	589753	3310871
Flex-v2	3	3/15	$(8\times 8)/8$	12101	2517	45773	461.4s	308684.76s	2669965	3310871
Perimeter	6	6/4	$(16\times 16)/16$	1980	303	9698	49.97s	2360.27s	227071	230773
Pose	4	4/8	$(13\times 13)/12$	24068	8800	24068	461.4s	21996.25s	526708	569513
Pendulum	3	2/3	$(40\times 40)/33$	4667	243	19899	45.46s	2105.321s	123891	128322
Tot	4	4/5	$(85\times 94)/56$	8930	348	52982	82.11s	17370.07s	424261	742099
Image3d	10	10/9	$(178\times 152)/130$	130	84	1456	2.34s	53.68s	12721	29415
Heron3d	6	5/7	$(16\times 14)/13$	23	22	90	0.411s	0.738s	1525	3071
Nachtwey	6	6/5	$(11\times 18)/11$	244	106	244	7.23s	5.36s	58376	12983
Storti	6	5/2	$(24\times 113)/20$	12	4	32	0.177s	0.053s	1089	343

At the moment, we are currently working on the analysis of the failure probability of our new Dixon resultant algorithm.

References

- Cuyt, A., and Lee, W.-S.: Sparse Interpolation of Multivariate Rational Functions. *J. Theoretical Comp. Sci.* **412**, pp. 1445–1456, Elsevier, 2011.
- Dixon, A. : On a form of the Eliminant of Two Quantics. *Proceedings of the London Mathematical Society* **2**, (1908), pp. 468–478.
- Jinadu, A., and Monagan, M. : An Interpolation Algorithm for computing Dixon Resultants. *Accepted for CASC '2022.*
- Kapur, D., Saxena, T., and Yang, L.: Algebraic and Geometric Reasoning using Dixon Resultants. *Proceedings of ISSAC '94* pp. 99–107, 1994.
- Kapur, D., and Saxena, T.: Comparison of Various Multivariate Resultant formulations. *Proceedings of ISSAC '95* , pp. 187–194, ACM, 1995.
- Lewis, R.: Dixon-EDF: The Premier Method for Solution of Parametric Polynomial Systems. *Special Sessions ACA (2015), Springer, pp. 237–256.*
- Lewis, R.: *Resultants, Implicit Parameterizations, and Intersections of Surfaces.* Proceedings of ICMS *(2018), Springer, pp. 310–318.*
- *Zippel, R.: Probabilistic Algorithms for Sparse Polynomials.* Proceedings of EUROSAM '79 *, pp. 216–226, Springer-Verlag, 1979.*