Solving Parametric Polynomial Systems using Dixon resultants

Ayoola Jinadu

Department of Mathematics, Simon Fraser University
November 2, 2022

This is joint work with Michael Monagan

Motivation

1. We have some parametric polynomial systems that failed badly in Maple and Magma when Groebner basis and Triangular sets are used to solve them.

   Many of these parametric polynomial systems can be found in Lewis papers [2015, 2018, 2019]

2. These methods can take a very long time to execute or run out of memory after several hours.

Let $X = \{x_1, x_2, \cdots, x_n\}$ be the set of variables and $Y = \{y_1, y_2, \cdots, y_m\}$ be the set of parameters, we define $\mathcal{F} = \{f_1, \cdots, f_n\} \subset \mathbb{Q}[Y][X]$ to be a parametric polynomial system with $n \geq 2$ and $m \geq 1$.

For us, solving parametric systems means : we want to obtain a polynomial with x_1 as the only remaining variable including the m parameters by eliminating the $n-1$ variables $\{x_2, x_3, x_4, \cdots, x_n\}$.

Dixon resultant formulation

1. Construct Cancellation matrix C
2. Compute Dixon polynomial $\Delta = \det(C)$.
3. Build the Dixon matrix D
   - The entries of D are in $\mathbb{Q}[x_1, Y]$
4. Select a maximal minor M, an $r \times r$ sub-matrix of D with $r = \operatorname{rank}(D) = \operatorname{rank}(M)$.
   - Choose random integers $\alpha \in \mathbb{Z}_p^{m+1}$ for $x_1, y_1, \cdots, y_m$, then do **Gaussian elimination** on $D(\alpha)$ over $\mathbb{Z}_p$.
5. Compute the Dixon resultant $R = \det(M)$.
6. The Dixon resultant R is in the elimination ideal $J = \mathbb{Q}[Y][x_1] \cap \langle f_1, f_2, \cdots, f_n \rangle$
   - When R is factored over $\mathbb{Q}$, R often has many repeated factors and a large polynomial content
   - As R is a multiple of the unique generator of J, one may need to identify the extraneous factors

The Dixon matrix D can be rectangular and if square, $R = \det(D)$ is often 0, hence step 4.

An Example (Heron2d system)

Let

$$\mathcal{F} = \{x_2^2 + x_3^2 - y_3^2,\ (x_2 - y_1)^2 + x_3^2 - y_2^2,\ -x_3 y_1 + 2x_1\} \tag{1}$$

with variables $X = \{x_1, x_2, x_3\}$ and parameters $Y = \{y_1, y_2, y_3\}$.

Goal: Eliminate $\{x_2, x_3\}$

The Dixon matrix D for $\mathcal{F}$ is

$$D = \begin{bmatrix} -2y_1^2 & 0 & y_1^3 - y_1 y_2^2 + y_1 y_3^2 \\ 0 & -2y_1^2 & 4x_1 y_1 \\ y_1^3 - y_1 y_2^2 + y_1 y_3^2 & 4x_1 y_1 & -2y_1^2 y_3^2 \end{bmatrix} \tag{2}$$

and its determinant which is the Dixon resultant

$$R = \det(D) = 2y_1^4(16x_1^2 + y_1^4 - 2y_1^2 y_2^2 - 2y_1^2 y_3^2 + y_2^4 - 2y_2^2 y_3^2 + y_3^4) \in \mathbb{Q}[y_1, y_2, y_3][x_1]. \tag{3}$$

Problem Formulation

Let $d = \deg(R, x_1)$ and let

$$R = \sum_{k=0}^{d} r_k(y_1, \cdots, y_m) x_1^k \in \mathbb{Q}[Y][x_1] \tag{4}$$

be the Dixon resultant of $\mathcal{F}$ in x_1 and let $C = \gcd(r_0, r_1, \cdots, r_d)$ be the polynomial content of 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

$$R_j = x_1^{d_{T_j}} + \sum_{k=0}^{d_{T_j - 1}} \frac{f_{jk}(y_1, y_2, \cdots, y_m)}{g_{jk}(y_1, y_2, \cdots, y_m)} x_1^{d_{j,k}} \tag{5}$$

where

- $\hat{r} = C/L$ for some $L \in \mathbb{Q}[Y]$,
- each R_j is monic and square-free in $\mathbb{Q}(Y)[x_1]$, i.e., $\gcd(R_j, R_j^{'}) = 1$, and
- $\gcd(R_i, R_j) = 1$ for $i \neq j$.
- $\gcd(f_{jk}, g_{jk}) = 1$ and $d_{T_j} \leq \deg(R, x_1)$.

Our Goal : Interpolate all the monic square-free factors R_j and **NOT** R in expanded form.

An Example (Robot arms system)

Variables $X = \{t_1, t_2, b_1, b_2\}$ and Parameters $Y = \{aa, al, l_1, l_2, l_3, x, y\}$. Eliminate $\{t_2, b_1, b_2\}$.

The Dixon resultant R in factored form can be expressed as $CA_1^{24}A_2^4A_3^2A_4^2$ and it has **6,924,715** terms when expanded

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

$$A_1 = \left({t_1}^2 + 1\right)$$

$$A_2 = \left(al^2{l_1}^2 + 2al^2l_1x - al^2{l_2}^2 - 2al^2l_2l_3 - al^2{l_3}^2 + al^2x^2 + al^2y^2 + {l_1}^2 + 2l_1x - {l_2}^2 + 2l_2l_3 - {l_3}^2 + x^2 + y^2\right){t_1}^2 + \left(-4al^2l_1y - 4l_1y\right)t_1 + al^2{l_1}^2 - 2al^2l_1x - al^2{l_2}^2 - 2al^2l_2l_3 - al^2{l_3}^2 + al^2x^2 + al^2y^2 + {l_1}^2 - 2l_1x - {l_2}^2 + 2l_2l_3 - {l_3}^2 + x^2 + y^2$$

$$A_3 = \left(aa^2 + 2aal_2\right){t_1}^2 + aa^2 - 4aal_1 + 2aal_2 + 4{l_1}^2 - 4l_1l_2$$

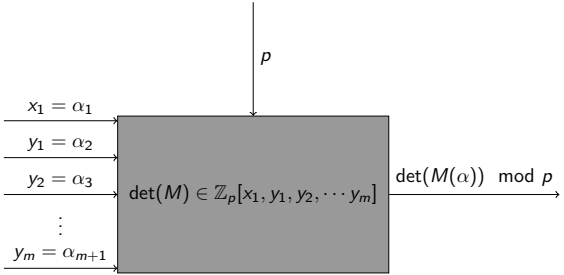
$$A_4 = \left(aa^2 - 2aal_2\right){t_1}^2 + aa^2 - 4aal_1 - 2aal_2 + 4{l_1}^2 + 4l_1l_2$$

We compute $R_1 = A_1, R_2 = \mathrm{monic}(A_2, t_1)$ and $R_3 = \mathrm{monic}(A_3A_4, t_1)$.
The largest coefficient to be interpolated is $\mathrm{LC}(A_2, t_1)$ and it has 14 terms.
The product $R_1R_2R_3$ has 450 terms when the fractions are cleared.

Our Dixon resultant algorithm

Let M be a maximal minor of a Dixon matrix D with polynomial entries in $x_1, y_1, y_2, \cdots, y_m$
To Interpolate the $R_j's$:

1. We use a black box representation for $\det M$.
   - A black box **BB** : $\mathbb{Z}_p \times \mathbb{Z}_p^m \to \mathbb{Z}_p$ is a computer program that stores a Dixon matrix M and takes as inputs $\alpha \in \mathbb{Z}_p^{m+1}$ and prime p to output $\det(M(\alpha)) \in \mathbb{Z}_p$.



2. Compute all the degree bounds with high probability
   - We need $\deg(R, x_1)$, maximum partial degrees $\max_{1 \le j \le l} \{_{0 \le k \le T_j - 1} \{(\deg f_{jk} y_i), \deg(g_{jk}, y_i)\}\}$
   - The total degrees $\{(\deg f_{jk} y_i) + \deg(g_{jk}, y_i)\}$ for each coefficient of all the monic square free factors R_j.

Our Dixon resultant algorithm

3. Feeding some input points $(\beta_0, \alpha^j), (\beta_1, \alpha^j), \cdots (\beta_{\deg(R,x_1)}, \alpha^j)$ to the black box for some $\alpha \in \mathbb{Z}_p^m$ to output $\theta_{i,j}$ and we densely interpolate polynomial univariate images

$$\bar{h}_1(x_1) = a_{d,1}x_1^d + a_{d-1,1}x_1^{d-1} + \cdots a_{1,1}x_1 + a_{0,1}$$

$$\vdots$$

$$\bar{h}_t(x_1) = a_{d,t}x_1^d + a_{d-1,t}x_1^{d-1} + \cdots a_{1,t}x_1 + a_{0,t}$$

using points $\{(\beta_i, \theta_{i,j} : 0 \leq i \leq d) : 1 \leq j \leq t\}$ and $d = \deg(R, x_1)$.

4. Make these images monic + square free factorization in x_1:

$$\bar{h}_1(x_1) \Rightarrow h_{t,1}(x_1), h_{1,2}(x_1) \cdots h_{1,l}(x_1)$$

$$\vdots$$

$$\bar{h}_t(x_1) \Rightarrow h_{t,1}(x_1), h_{t,2}(x_1), \cdots h_{t,l}(x_1)$$

5. Do sparse multivariate rational function interpolation on the coefficients of l monic univariate factors $\{h_{i,1} : 1 \leq i \leq t\}$ in x_1 to obtain the R_j's.
6. If 1 prime is not enough to get the $R_j's$ then use the support from the first image and get more images using a new prime. Then do **Chinese remaindering + Rational number reconstruction.**

Benchmark: DixonRes versus Minor Expansion and Zippel's Interpolation

System names	$\#Eq$	n/m	dim D/**Rank**	$\#S$	$t_{\max}$	$\#R$	**DRes**	**Minor**	**Zippel**
Robot-t_1	4	4/7	$(32 \times 48)/20$	450	14	6924715	7.34s	2562.6s	$> 10^5$s
Robot-t_2	4	4/7	$(32 \times 48)/20$	13016	691	16963876	316.99s	!	$> 10^5$s
Robot-b_1	4	4/7	$(32 \times 48)/20$	334	85	6385205	27.78s	182.4s	$> 10^5$s
Robot-b_2	4	4/7	$(32 \times 48)/20$	11737	624	16801877	241.61s	!	$> 10^5$s
Pendulum	3	2/3	$(40 \times 40)/33$	4667	243	19899	45.46s	1721.50s	2105.321s
Heron5d	15	14/16	$(707 \times 514)/399$	823	822	12167689	23.12s	!	$> 10^5$s
Tot	4	4/5	$(85 \times 94)/56$	8930	348	52982	82.11s	!	17370.07s
Flex-v1	3	3/15	$(8 \times 8)/8$	5685	2481	45773	201s	5.09s	308684.76s
Flex-v2	3	3/15	$(8 \times 8)/8$	12101	2517	45773	461.4s	5.02s	308684.76s
Perimeter	6	6/4	$(16 \times 16)/16$	1980	303	9698	49.97s	18.23	2360.27s
Storti	6	5/2	$(24 \times 113)/20$	12	4	32	0.177s	229.945s	0.053s
Pose	4	4/8	$(13 \times 13)/12$	24068	8800	24068	461.4s	4.48s	21996.25s
Image3d	10	10/9	$(178 \times 152)/130$	130	84	1456	2.34s	1.04s	53.68s
Heron3d	6	5/7	$(16 \times 14)/13$	23	22	90	0.411s	0.014s	0.738s
Nachtwey	6	6/5	$(11 \times 18)/11$	244	106	244	7.23s	0.424s	5.36s

! = ran out of memory

$t_{\max}$ = the number terms in the largest polynomial to be interpolated (Compare $t_{\max}$ to $\#R$!)

Dres = timings for our Dixon resultant algorithm implemented in **Maple** with some parts coded in **C**

Zippel = timings for Zippel's sparse interpolation of R implemented in **Maple** with some parts coded in **C**

Minor = Gentleman and Johnson Minor expansion implemented in **Maple**

References

Jinadu, A., and Monagan, M.:
An Interpolation Algorithm for computing Dixon Resultants.
Proceedings of CASC '2022, LNCS **13366**: pp 185-205, Springer, 2022.

Cuyt, A., and Lee, W.-S.
Sparse Interpolation of Multivariate Rational Functions.
Theoretical Computer Science **412**, 16 (2011), pp. 1445–1456.

Lewis, R. H.
Dixon-EDF: The Premier Method for Solution of Parametric Polynomial Systems.
In Special Sessions in Applications of Computer Algebra (2015), Springer, pp. 237–256.

Lewis, R. H.
Resultants, Implicit Parameterizations, and Intersections of Surfaces.
In International Congress on Mathematical Software (2018), Springer, pp. 310–318.

Lewis, R. H.
New Heuristics and Extensions of the Dixon Resultant for Solving Polynomial Systems.
Applications of Computer Algebra, Montreal, Canada (2019), pp. 16–20.

Ben-Or, M., and Tiwari, P.
A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation.
Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (1988), pp. 301–309.