M1CCA Projet Interpolation polynomiale en deux variables

Yudi Sun 21400421
Yudi.Sun@etu.sorbonne-universite.fr

Long Qian 21400529
Long.Qian@etu.sorbonne-universite.fr

Encadrant : Jérémy Berthomieu
jeremy.berthomieu@lip6.fr

May 16, 2025

Contents

1 Introduction

Solving systems of polynomial equations is foundational in computational algebra, with applications in cryptography, robotics, and coding theory. Gröbner bases offer a systematic framework to analyze such systems by characterizing their vanishing ideals. For univariate polynomials, the ideal is generated by a single polynomial $\prod_{i=0}^{n-1}(x - x_i)$ when roots are distinct. In the bivariate case, if solutions $(x_0, y_0), \dots, (x_{n-1}, y_{n-1})$ have pairwise distinct x_i, the ideal is generated by two polynomials:

$$g_0 = \prod_{i=0}^{n-1}(x - x_i) \quad \text{and} \quad g_1 = y - h(x),$$

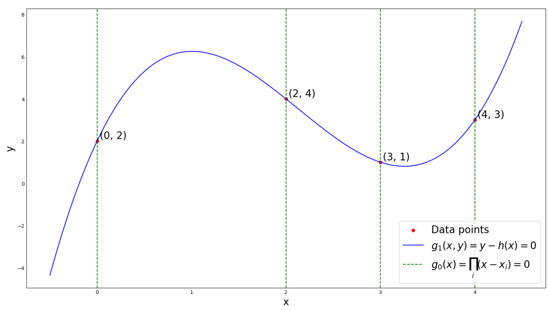
where $h(x)$ interpolates the y_i.

However, when x_i are non-distinct, g_1 becomes undefined, requiring new methods to construct low-degree polynomials vanishing on overlapping solutions. This work addresses three objectives:

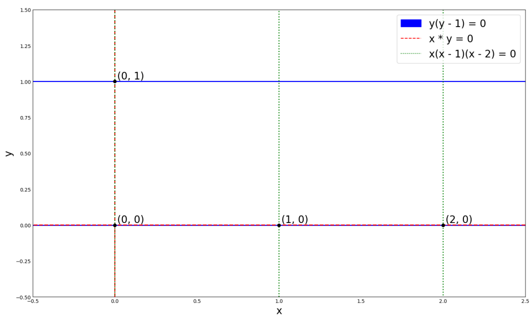
1. Proving that g_0 and g_1 suffice under the distinct x_i hypothesis.

2. Developing minimal-degree polynomials for non-distinct x_i and analyzing computational costs.

3. We search for polynomials of low total degree under general conditions, given the set E containing the points (x_i, y_i):
$$E = \{(x_0, y_0), (x_1, y_1), \dots, (x_{n-2}, y_{n-2}), (x_{n-1}, y_{n-1}), (x_n, y_n)\}.$$



(a) Illustration of g_0 and g_1 (Example)



(b) Points E and their vanishing ideal (Example)

2 Polynomial Decomposition with Respect to Interpolation Points

2.1 Distinct Horizontal Coordinates

Consider a finite set of points in the plane:

$$E_0 = \{(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1})\},$$

with the assumption that all the x_i are distinct.

We define the polynomial:

$$g_0(x) = \prod_{i=0}^{n-1}(x - x_i),$$

and the interpolation polynomial $h(x)$ such that $h(x_i) = y_i$ for all $i = 0, 1, \dots, n-1$.

Any bivariate polynomial $p(x, y)$ that vanishes on E_0 can be decomposed as:

$$p(x, y) = g_0(x)q(x, y) + (y - h(x))r(x, y), \quad q, r \in \mathbb{K}[x, y]. \tag{1}$$

Proof: Given $p(x, y) = \sum_{i=0}^{d} p_i(x)y^i$, expand y^i in powers of $(y - h(x))$:

$$y^i = ((y - h(x)) + h(x))^i = \sum_{j=0}^{i} \binom{i}{j} (y - h(x))^{i-j} h(x)^j.$$

Thus,

$$\begin{aligned}
p(x,y) &= \sum_{i=0}^{d} p_i(x) \sum_{j=0}^{i} \binom{i}{j} (y-h(x))^{i-j} h(x)^j \\
&= \sum_{k=0}^{d} \left[\sum_{j=0}^{d-k} p_{k+j}(x) \binom{k+j}{j} h(x)^j \right] (y-h(x))^k.
\end{aligned}$$

Define:

$$R(x) = \sum_{j=0}^{d} p_j(x) h(x)^j, \quad S(x,y) = \sum_{k=1}^{d} \left[\sum_{j=0}^{d-k} p_{k+j}(x) \binom{k+j}{j} h(x)^j \right] (y-h(x))^{k-1},$$

and thus we have:

$$p(x,y) = R(x) + (y-h(x))S(x,y).$$

Since $p(x_i, y_i) = 0$, it follows that:

$$R(x_i) = \sum_{j=0}^{d} p_j(x_i) h(x_i)^j = \sum_{j=0}^{d} p_j(x_i) y_i^j = p(x_i, y_i) = 0.$$

Thus, $R(x)$ is divisible by $g(x)$, and we have:

$$p(x,y) = g_0(x)q(x,y) + (y-h(x))r(x,y).$$

This completes the proof.

2.2 Multiple Vertical Coordinates for a Single Horizontal Coordinate

Now consider the extended set with one repeated horizontal coordinate:

$$E_1 = E_0 \cup \{(x_n, y_n), (x_n, y_n')\}, \quad y_n \neq y_n'.$$

Define:

$$g(x) = \prod_{i=0}^{n} (x - x_i), \quad h(x_i) = y_i, \quad h_1(x_n) = y_n, \quad h_2(x_n) = y_n'.$$

As before, expand $p(x,y)$ around $y - h(x)$:

$$p(x,y) = g(x)q(x) + (y-h(x))S(x,y).$$

At $x = x_n$, we have:

$$p(x_n, y_n) = (y_n - h(x_n))S(x_n, y_n) = 0, \quad p(x_n, y_n') = (y_n' - h(x_n))S(x_n, y_n') = 0.$$

Since $y_n \neq h(x_n)$ or $y_n' \neq h(x_n)$, we must have:

$$S(x_n, y_n) = S(x_n, y_n') = 0.$$

$h_1(x)$ is the polynomial interpolating the set $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$. $h_2(x)$ is the polynomial interpolating the set $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n')\}$. So we have $y_n = h_1(x_n)$ and $y_n' = h_2(x_n)$. Thus, $S(x,y)$ can be factorized as:

$$S(x,y) = (x - x_n)q_1(x,y) + (y - h_1(x_n))(y - h_2(x_n))q_2(x,y),$$

leading to the general factorization:

$$p(x,y) = g(x)q(x) + (y-h(x))(x-x_n)q_1(x,y) + (y-h_1(x))(y-h_2(x))q_2(x,y). \tag{2}$$

Here, the first term removes all base points in E_0, the second term handles single vertical interpolations, and the third term ensures vanishing at all multiple vertical coordinates for each horizontal coordinate.

2.3 More General Conditions

Let

$$E_0 = \{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$$

be an arbitrary finite set of N points, with no assumption on the x_i and y_i. We seek a nonzero bivariate polynomial

$$P(x,y) \;=\; \sum_{a+b \leq d} c_{a,b}\, x^a y^b$$

of total degree d such that

$$P(x_i, y_i) \;=\; 0, \quad i = 1, \ldots, N,$$

and we wish to make d as small as possible.

3 Vandermonde Method

3.1 Considering only one variable x

$$E_2 = \{x_0, x_1, \dots, x_{n-1}\}$$

be a set of n pairwise distinct points. The smallest nonzero polynomial that vanishes on E is

$$g_0(x) \;=\; \prod_{i=0}^{n-1} (x - x_i).$$

Equivalently, writing

$$g_0(x) = g_0 + g_1 x + \cdots + g_{n-1} x^{\,n-1},$$

we can find the coefficients $\{g_j\}$ by solving the Vandermonde linear system:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \end{pmatrix} = \begin{pmatrix} x_0^n \\ x_1^n \\ \vdots \\ x_{n-1}^n \end{pmatrix}.$$

Since $\det(Vandermonde) \neq 0$, the matrix is reversible and will not be described in detail in the following matrix. Hence the polynomial

$$x^n \;-\; \bigl(g_{n-1}x^{n-1} + g_{n-2}x^{n-2} + \cdots + g_1 x + g_0\bigr)$$

vanishes at each x_i.

3.2 Considering two variables x and y

$$E_0 = \{(x_0, y_0),\, (x_1, y_1),\, \dots,\, (x_{n-1}, y_{n-1})\},$$

with the x_i pairwise distinct. We seek a polynomial

$$h(x) = h_0 + h_1 x + \cdots + h_{n-1} x^{\,n-1}$$

such that

$$h(x_i) = y_i, \quad i = 0, \dots, n-1.$$

Equivalently, the function $(x, y) \mapsto y - h(x)$ vanishes on E. The coefficients $\{h_j\}$ are obtained by solving the Vandermonde system

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

Consequently, the polynomial

$$y \;-\; \bigl(h_{n-1}x^{n-1} + \cdots + h_1 x + h_0\bigr)$$

vanishes at each (x_i, y_i).

3.3 Considering the non-distinct case

Now we set

$$E_3 = \bigl\{(x_0, y_0),\, (x_1, y_1),\, \dots,\, (x_{n-2}, y_{n-2}),\, (x_{n-1}, y_{n-1}),\, (x_{n-1}, y'_{n-1})\bigr\}.$$

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1}^{n-1} \\ x_n^n \end{pmatrix}.$$

After that we know poly

$$G(x) = x^{n+1} \;-\; \big(g_n x^n + \cdots + g_1 x + g_0\big)$$

Let

$$h(x) \;=\; h_n x^n + h_{n-1}x^{n-1} + \cdots + h_1 x + h_0$$

be an unknown polynomial. Define

$$A(x,y) \;=\; (x - x_n)\big(y - h(x)\big).$$

Expanding gives

$$A(x,y) = x\,y \;-\; x\,h(x) \;-\; x_n\,y \;+\; x_n\,h(x) = x\,y \;-\; x_n\,y \;-\; h_n\,x^{n+1} - h_{n-1}\,x^n - \cdots - h_0.$$

To eliminate the x^{n+1}–term, introduce the monic polynomial

$$G(x) \;=\; x^{n+1} - \big(g_n x^n + g_{n-1}x^{n-1} + \cdots + g_0\big).$$

Then form

$$B(x,y) \;=\; A(x,y) \;+\; h_n\,G(x).$$

By construction B has total degree at most n in x, and one checks

$$B(x,y) = x\,y \;-\; \big(k\,y + k_n x^n + k_{n-1}x^{n-1} + \cdots + k_1 x + k_0\big).$$

Requiring that all coefficients of the monomials $\{x^n, x^{n-1}, \ldots, x, y, 1\}$ in $B(x,y)$ vanish yields a linear system for the unknowns

$$k,\; h_0,\; h_1,\; \ldots,\; h_n.$$

$$B(x,y) \;=\; x\,y \;-\; \big(k\,y + k_n x^n + \cdots + k_1 x + k_0\big) \;=\; 0.$$

which vanishes on the points

$$(x_0,y_0),\;(x_1,y_1),\;\ldots,\;(x_{n-2},y_{n-2}),\;(x_{n-1},y_{n-1}),\;(x_{n-1},y'_{n-1})\,.$$

So we can deduce the Vandermonde matrix

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^n & y_0 \\ 1 & x_1 & \cdots & x_1^n & y_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^n & y_{n-1} \\ 1 & x_{n-1} & \cdots & x_{n-1}^n & y'_{n-1} \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ \vdots \\ k_n \\ k \end{pmatrix} = \begin{pmatrix} x_0 y_0 \\ x_1 y_1 \\ \vdots \\ x_{n-1} y_{n-1} \\ x_{n-1} y'_{n-1} \end{pmatrix}.$$

The same reasoning applies to the final polynomial in y (e.g. $y^2 - \cdots$), leading again to a Vandermonde-type linear system.

$$F(x,y) = y^2 - \big(k\,y + k_n x^n + \cdots + k_1 x + k_0\big),$$

we observe:

$$x\,y = k\,y + k_n x^n + k_{n-1}x^{n-1} + \cdots + k_1 x + k_0.$$

$$y^2 - x\,y \;\longrightarrow\; y^2 - \big[k\,y + k_n x^n + \cdots + k_0\big].$$

Since

$$y^2 - x\,y = y^2 - x \cdot (xy),$$

we can factor out an x in the second term:

$$y^2 - x\big[k\,y + k_n x^n + \cdots + k_0\big] \;=\; y^2 - \big(k\,x\,y + k_n\,x^{\,n+1} + \cdots + k_0\,x\big) \longrightarrow\; y^2 - \big(l\,y + l_n\,x^{\,n} + \cdots + l_0\big)$$

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^n & y_0 \\ 1 & x_1 & \cdots & x_1^n & y_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^n & y_{n-1} \\ 1 & x_{n-1} & \cdots & x_{n-1}^n & y'_{n-1} \end{pmatrix} \begin{pmatrix} l_0 \\ l_1 \\ \vdots \\ l_n \\ l \end{pmatrix} = \begin{pmatrix} y_0^2 \\ y_1^2 \\ \vdots \\ y_{n-1}^2 \\ y_n^2 \end{pmatrix}.$$

3.4 More general conditions

3.4.1 Matrix complet

From the poly $p(x,y)$ and the previous reasoning

$$p(x,y) = g(x)q(x) + (y - h(x))(x - x_n)q_1(x,y) + (y - h_1(x))(y - h_2(x))q_2(x,y). \tag{3}$$

we can deduce une matrix complet :

$$W(x,y) = \begin{pmatrix} x_0^0 & x_0^1 & \cdots & x_0^n & y_0\, x_0^0 & y_0\, x_0^1 & \cdots & y_0\, x_0^n & \cdots & y_0^n\, x_0^n \\ x_1^0 & x_1^1 & \cdots & x_1^n & y_1\, x_1^0 & y_1\, x_1^1 & \cdots & y_1\, x_1^n & \cdots & y_1^n\, x_1^n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots & \ddots & \vdots \\ x_n^0 & x_n^1 & \cdots & x_n^n & y_n\, x_n^0 & y_n\, x_n^1 & \cdots & y_n\, x_n^n & \cdots & y_n^n\, x_n^n \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)^2}.$$

3.4.2 Optimization to a lower degree

This matrix is too big, we need to do the optimization to reduce that

We firstly think about

$$E = \{(x_0, y_0),\ (x_1, y_1),\ \ldots, (x_n, y_n)\}.$$

- The constant polynomial $P(x,y) = 1$ does *not* vanish on E.
- Is there a linear polynomial $x + a_0$ that vanishes on E?

  $$\text{Answer: Yes if and only if} \quad W_{x,y} \begin{pmatrix} 1 \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix},$$

  where

  $$W_{x,y} = \begin{pmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}.$$

- If the answer above is "yes" then next think about: is there a quadratic polynomial

  $$x^2 + b_{10}x + b_{00}$$

  that vanishes on E?

  $$x^2 + a_{00}x = x(x + a_{00})$$

  vanish on E, the answer is "yes" and

  $$x^3 + x^2 b_{20}x + xb_{10} + b_{00}$$

  is also vanish on E.

- If the answer above is "No" then is there the poly

  $$x^2 + a_{10}x + a_{00}$$

  vanish on E ? The answer is "yes" only in this situation :

  $$\begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

- Now we can do the optimization

  $$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \cancel{x_0^3} \\ 1 & x_1 & x_1^2 & \cancel{x_1^3} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cancel{x_n^3} \end{pmatrix}}_{\text{(block 1)}} \underbrace{\begin{pmatrix} y_0 & \cancel{y_0 x_0} & \cancel{y_0 x_0^2} \\ y_1 & \cancel{y_1 x_1} & \cancel{y_1 x_1^2} \\ \vdots & \vdots & \vdots \\ y_n & \cancel{y_n x_n} & \cancel{y_n x_n^2} \end{pmatrix}}_{\text{(block 2)}} \underbrace{\begin{pmatrix} y_0^2 & \cdots & y_0^3 & \cdots & \cancel{y_0^4} \\ y_1^2 & \cdots & y_1^3 & \cdots & \cancel{y_1^4} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ y_n^2 & \cdots & y_n^3 & \cdots & \cancel{y_n^4} \end{pmatrix}}_{\text{(block 3)}}$$

- block 1 : we find that

$$x^3 + a_{20}x^2 + xa_{10} + a_{00}$$

block 2 : we find that

$$xy + b_{01}y + b_{20}x^2 + b_{10}x + b_{00}$$

block 3 : we find that

$$y^4 + c_{03}y^3 + c_{02}y^2 + c_{01}y + c_{20}x^2 + c_{10}x + c_{00}$$

So that We successfully simplify and remove the redundant parts to get the polynomial with the minimum degree

3.5 Considering three variables x,y,z

For a given nonnegative integer d, consider all monomials

$$\{\, x^i y^j z^k : i,j,k \ge 0,\ i+j+k \le d\}.$$

There are

$$N(d) \;=\; \binom{d+3}{3} \;=\; \frac{(d+1)(d+2)(d+3)}{6}$$

such monomials.

The entry in the p-th row and q-th column of A is the value of the q-th monomial evaluated at the p-th sample point.

$$A_{p,q} \;=\; m_q(x_p, y_p, z_p) \;=\; x_p^{i_q}\, y_p^{j_q}\, z_p^{k_q}.$$

We form the $n \times N(d)$ *three-variable Vandermonde matrix*

$$W(x,y,z) \;=\; \begin{pmatrix} x_0^0y_0^0z_0^0 & \cdots & x_0^ny_0^0z_0^0 & \cdots & x_0^0y_0^nz_0^0 & \cdots & x_0^0y_0^0z_0^n & \cdots & x_0^ny_0^nz_0^n \\ x_1^0y_1^0z_1^0 & \cdots & x_1^ny_1^0z_1^0 & \cdots & x_1^0y_1^nz_1^0 & \cdots & x_1^0y_1^0z_1^n & \cdots & x_1^ny_1^nz_1^n \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ x_n^0y_n^0z_n^0 & \cdots & x_n^ny_n^0z_n^0 & \cdots & x_n^0y_n^nz_n^0 & \cdots & x_n^0y_n^0z_n^n & \cdots & x_n^ny_n^nz_n^n \end{pmatrix} \;\in\; \mathbb{R}^{(n+1)\times\binom{n+3}{3}}.$$

Here each column corresponds to one of the monomials $x^i y^j z^k$ with $i+j+k \le d$. Since $N(d) > n$ for sufficiently large d, the homogeneous linear system

$$W(x,y,z)\, c \;=\; 0,$$

with unknown coefficient vector

$$c = \big(c_{i,j,k}\big)_{i+j+k\le d},$$

has a nontrivial solution. Equivalently, the polynomial

$$P(x,y,z) \;=\; \sum_{i+j+k\le d} c_{i,j,k}\, x^i\, y^j\, z^k$$

vanishes at all points $(x_\ell, y_\ell, z_\ell) \in E$, $\ell = 0, \dots, n-1$, and is not identically zero.

3.6 Considering n variables

For an arbitrary number of variables $x^{(1)}, x^{(2)}, \dots, x^{(n)}$, we proceed in exactly the same way:

- List all monomials

$$\{(x^{(1)})^{\alpha_1}\,(x^{(2)})^{\alpha_2}\cdots(x^{(n)})^{\alpha_n}\} \quad \text{with} \quad \alpha_1+\alpha_2+\cdots+\alpha_n \le d.$$

  There are $\binom{d+n}{n}$ such monomials.

- Form the evaluation matrix

$$W \;\in\; \mathbb{R}^{N\times\binom{d+n}{n}}, \quad W_{i,\alpha} = \prod_{j=1}^{n}\big(x_i^{(j)}\big)^{\alpha_j}, \quad i = 1,\dots,N.$$

- Compute its nullspace $Wc = 0$. Any nonzero solution $c = (c_\alpha)$ gives a vanishing polynomial

$$P(x^{(1)},\dots,x^{(n)}) \;=\; \sum_{\alpha_1+\cdots+\alpha_n\le d} c_\alpha\,(x^{(1)})^{\alpha_1}\cdots(x^{(n)})^{\alpha_n}.$$

Thus, by exactly the same monomial-generation, matrix-assembly, and nullspace-computation steps, one obtains all polynomials of total degree $\le d$ that vanish on any finite set of points in $\mathbb{R}^n$.

3.7 Complexity

Method	**Evaluation matrix size**	**Total time complexity**
Naïve enumeration (2-D)	$n \times \binom{D+2}{2} \left(\approx n \cdot \frac{D^2}{2}\right)$	(set $D = n$) $\mathbf{O}(n^5)$
Filtered “standard monomial” method (2-D)	$\leq n \times n$	$\mathbf{O}(n^3)$
Filtered “standard monomial” method (n-D)	$\leq n \times n$	$\mathbf{O}(n^3)$

Table 1: **How the complexities are obtained.** *Naïve enumeration*: the evaluation matrix has $r = n$ rows and $s = \binom{D+2}{2} \approx \frac{D^2}{2}$ columns. Gaussian elimination on an $r \times s$ matrix costs $O(rs^2)$, hence $O\big(n\,(D^2/2)^2\big) = O(nD^4)$. Choosing the usual $D = n$ gives the quoted $O(n^5)$. *Filtered standard-monomial method* (both 2-D and the uploaded 3-D `xyz` version): after removing multiples of known leading terms, the matrix never exceeds $n \times n$. The same elimination therefore costs $O(n \cdot n^2) = O(n^3)$.

Meaning of D. D is the *maximum total degree* to which monomials are enumerated in the naïve scheme. For a zero-dimensional set of n points it is safe to choose $D = n$, which guarantees that all polynomials vanishing on the point set are captured while keeping the matrix size minimal for that guarantee.

4 Algorithm

4.1 Goal and Approach

The implementation aims to compute *vanishing polynomials* on a finite set of points in the plane. Given planar coordinates

$$\{(x_i, y_i)\}_{i=1}^{N} \subset \mathbb{R}^2,$$

we seek all bivariate polynomials $P(x, y)$ (up to a chosen total degree d) such that

$$P(x_i, y_i) = 0 \qquad \text{for every } i = 1, \ldots, N.$$

4.2 Algorithm Outline

More concretely, the procedure consists of the following steps:

1. **Monomial Basis Generation.** List all monomials $\{x^i y^j : i + j \leq d\}$.
2. **Matrix Assembly.** Form the $N \times M$ quasi-Vandermonde matrix A by evaluating each monomial at each point:
$$A_{k,(i,j)} = x_k^i \, y_k^j, \quad k = 1, \ldots, N.$$
3. **Nullspace Computation.** Compute a basis for the nullspace of A, i.e. solve
$$A\mathbf{c} = \mathbf{0}.$$
Each independent solution $\mathbf{c} = (c_{ij})$ corresponds to a vanishing polynomial.
4. **Polynomial Reconstruction.** Convert each nullspace vector $\mathbf{c}$ back into the polynomial
$$P(x, y) = \sum_{i+j \leq d} c_{ij} x^i y^j.$$

Algorithm 1: Vanishing Polynomials via Nullspace Computation

Input: Point set $E = \{(x_i, y_i)\}_{i=1}^{N}$, maximum degree d
Output: List of polynomials vanishing on E

$M \leftarrow$ generate_all_monomials_up_to_degree(d); `// all monomials with total degree` $\leq d$
;
$A \leftarrow$ zero matrix of size $N \times |M|$;
for $i = 1, \ldots, N$ **do**
  for $j = 1, \ldots, |M|$ **do**
    $A_{i,j} \leftarrow M_j(x_i, y_i)$; `// evaluate the` j`-th monomial at` (x_i, y_i)
NS $\leftarrow \ker(A)$; `// compute basis of the nullspace`
;
solutions $\leftarrow [\,]$;
foreach $c \in$ NS **do**
  $P(x, y) \leftarrow \sum_{j=1}^{|M|} c_j \, M_j$; `// reconstruct polynomial from coefficients`
  solutions.append(P);
return *solutions*;

4.3 Implementation with SymPy

The reference implementation is written in Python using the symbolic library SymPy.

- Symbols `x`, `y` represent the variables;
- monomials are generated programmatically, and the evaluation matrix A is built via `sympy.Matrix`.
- Calling `A.nullspace()` returns exact basis vectors for $\ker(A)$.

5 Performance and Optimizations

5.1 Baseline Complexity

If all $m_d = \binom{d+2}{2}$ monomials of total degree $\le d$ are used, the evaluation matrix has size $N \times m_d$. With SymPy's fraction-free Gaussian elimination the worst-case time cost is

$$O\big(\min\{N, m_d\}\, m_d^2\big),$$

which degenerates to $\Theta(N^3)$ when d grows in the same order as N.

5.2 Two Optimisations Present in `poly.py`

1. **Leading-Term Filtering** The function `nullspace_polynomials` maintains a list `lead_terms`. Every time a vanishing polynomial P is found, its *leading term*[1] $\mathrm{lt}(P)$ is appended to that list. In the next degree, candidate monomials are filtered by

$$\texttt{monos\_filt} = \big\{\, m \mid \nexists \ell \in \texttt{lead\_terms} : \ell \mid m \big\},$$

discarding any monomial divisible by a recorded leader and thus shrinking the matrix width significantly.

Algorithm 2: Vanishing-Polynomial Search with Leading-Term Filtering (identical to `poly.py`)

Input: point set $E \subset \mathbb{R}^2$, degree cap $d_{\max} = |E|$
Output: degree-grouped generators of the vanishing ideal

$L \leftarrow \varnothing$; `// set of stored leading monomials`
for $d = 0, 1, \ldots, d_{\max}$ **do**
  $M \leftarrow \{x^i y^j \mid i + j \le d\}$;
  $M_{\text{std}} \leftarrow \{m \in M \mid \nexists \ell \in L : \ell \mid m\}$;
  build the evaluation matrix $\mathbf{V}$ and compute $\ker(\mathbf{V})$, yielding $P(x, y)$;
  foreach $P(x, y)$ **do**
    $L \leftarrow L \cup \{\mathrm{lt}(P)\}$; `// record the leader`
  return $P(x, y)$ (if non-empty)

5.3 Illustrative Example

Let

$$E = \{(0,0),\ (0,2),\ (1,0),\ (2,1)\}$$

Running the algorithm yields:

```
Degree <= 2   monomials = {1,y,x,y^2,xy,x^2}   nullspace dim = 2
      x*y + 2*y^2 - 4*y
      x^2 - x + 2*y^2 - 4*y

Degree <= 3   monomials = {1,y,x,y^2,y^3}       nullspace dim = 1
      y^3 - 3*y^2 + 2*y
------------------------------------------------------------
Reduced basis of the vanishing ideal:
    y*(y - 1)
    x*y
    x*(x - 2)*(x - 1)
```

These polynomials agree exactly with the factorization predictions of Theorem 1 and Proposition 2.

5.4 Complexity Analysis and Performance

Denote $N = |E|$ and let $m_d = \binom{d+2}{2}$ be the initial number of monomials of degree $\le d$. Each iteration solves an $|E| \times m_d$ linear system by Gaussian elimination in $O(N\, m_d^2)$ time. Empirically, the filtering step reduces the number of monomials by about 50% for $d \ge 3$, resulting in an overall cubic-time speedup compared to the unfiltered method.

[1] For $P = \sum c_{ij} x^i y^j$, the monomial $x^i y^j$ with the largest exponent pair (i, j) under SymPy's default lex order ($x \succ y$) is taken as the leading term and stored without its coefficient.

6 Extension to Three Dimensions

6.1 Goal and Approach

Given a set of sample points

$$\{(x_i, y_i, z_i)\}_{i=1}^{N} \subset \mathbb{R}^3,$$

we seek all trivariate polynomials $P(x, y, z)$ of total degree $\leq d$ such that

$$P(x_i, y_i, z_i) = 0, \quad i = 1, \ldots, N.$$

6.2 Algorithm Outline

1. **Monomial Basis Generation.** List all monomials $\{x^i y^j z^k : i + j + k \leq d\}$.
2. **Matrix Assembly.** Form the $N \times M$ evaluation matrix A by

$$A_{r,(i,j,k)} = x_r^i \, y_r^j \, z_r^k, \quad r = 1, \ldots, N.$$

3. **Nullspace Computation.** Compute a basis for the nullspace of A, i.e. solve $A\mathbf{c} = \mathbf{0}$.
4. **Polynomial Reconstruction.** Each nullspace vector $\mathbf{c} = (c_{i,j,k})$ yields a vanishing polynomial

$$P(x, y, z) = \sum_{i+j+k \leq d} c_{i,j,k} \, x^i y^j z^k.$$

6.3 Implementation Notes

In the 3-D version (see `3poly.py`):

- Variables are declared as `x,y,z = sp.symbols("x y z")`.
- Monomials are generated by three nested loops with $i + j + k \leq d$.
- The evaluation matrix uses `m.subs({x:xi,y:yi,z:zi})`.
- Leading-term filtering and nullspace steps are otherwise identical.

6.4 Generalization to n Dimensions

The same procedure extends without change to an arbitrary number of variables. Given sample points

$$\{(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})\}_{i=1}^{N} \subset \mathbb{R}^n,$$

we seek all n-variate polynomials of total degree $\leq d$ vanishing on these points. One simply:

1. Generate the monomial basis $\{(x^{(1)})^{\alpha_1} \cdots (x^{(n)})^{\alpha_n} : \alpha_1 + \cdots + \alpha_n \leq d\}$.
2. Assemble the $N \times M$ evaluation matrix

$$A_{i,\alpha} = \prod_{j=1}^{n} \bigl(x_i^{(j)}\bigr)^{\alpha_j},$$

   where $\alpha = (\alpha_1, \ldots, \alpha_n)$.
3. Compute the nullspace of A to find coefficient vectors $\mathbf{c}$.
4. Reconstruct each vanishing polynomial

$$P(x^{(1)}, \ldots, x^{(n)}) = \sum_{\alpha_1 + \cdots + \alpha_n \leq d} c_\alpha \, (x^{(1)})^{\alpha_1} \cdots (x^{(n)})^{\alpha_n}.$$

Thus, by exactly the same monomial-generation, matrix-assembly, and nullspace-computation steps, one obtains all polynomials of bounded total degree that vanish on any finite set of points in $\mathbb{R}^n$.

7 Conclusion

Our project addresses the problem of *constructing low-degree polynomials vanishing on a given point set* in two and higher dimensions. The main achievements are:

1. **Theoretical Proof**
   - When all x_i are distinct, we prove that the vanishing ideal is generated by
     $$g_0(x) = \prod_i (x - x_i), \quad g_1(x, y) = y - h(x).$$
   - In the case of no-distinct points, we derive a factorization combining horizontal and vertical factors, ensuring vanishing at every coincident point.
2. **Algorithm Design and Implementation**
   - A Python/SymPy prototype implements the pipeline:
     $$\text{enumerate monomials} \;\rightarrow\; \text{construct generalized Vandermonde matrix} \;\rightarrow\; \text{compute nullspace.}$$
   - We introduce a *vertex-filtering* strategy that halves the matrix width, yielding a practical speed-up over the $O(N^3)$ baseline.
3. **Extensions and Experiments**
   - The method is extended to three variables and validated on small-scale data.
   - The consistency between the theoretical decomposition and the numerical results was verified by multiple tests and comparison of the specific outputs of the algorithm..
4. **Limitations and Future Work**
   - The current prototype relies on exact rational arithmetic, incurring high memory and time costs for large point clouds.
   - Future directions include:
     - Numerically stable fraction-free elimination or floating-point approximations.
     - GPU or parallel linear-algebra libraries for large-scale matrix solves.