Abstract: We introduce a novel computational framework, Principal Variety Analysis (PVA), for primarily nonlinear data modeling. PVA accommodates algebraic sets as the target subspace through which limitations of other existing approaches is dealt with. PVA power is demonstrated in this paper for learning kinematics of objects, as an important application. PVA takes recorded coordinates of some pre-specified features on the objects as input and outputs a lowest dimensional variety on which the feature coordinates jointly lie. Unlike existing object modeling methods, which require entire trajectories of objects, PVA requires much less information and provides more flexible and generalizable models, namely an analytical algebraic kinematic model of the objects, even in unstructured, uncertain environments. Moreover, it is not restricted to predetermined model templates and is capable of extracting much more general types of models. Besides finding the kinematic model of objects, PVA can be a powerful tool to estimate their corresponding degrees of freedom. PVA computational success depends on exploiting sparsity, in particular algebraic dimension minimization through replacement of intractable $\ell_0$ norm (rank) with tractable $\ell_1$ norm (nuclear norm). Complete characterization of the assumptions under which $\ell_0$ and $\ell_1$ norm minimizations yield virtually the same outcome is introduced as an important open problem in this paper.

Keywords: Nonlinear Dimensionality Reduction, Nonlinear Data Modeling, Analytical Algebraic Modeling, Norm Minimization, Sparsity, Kinematics Learning

1 Introduction

With the advent of many advances in technology, such as low-cost high throughput sensors, human genome sequencing, physics and astronomy, social media, and etc., available data from real world phenomena has reached an unprecedented level. Captured high quality data are in fact growing at an astronomical rate. That necessitates advancement of algorithms and methods to extract models, patterns, and knowledge from data. For instance, methods for mostly linear dimensionality reduction [1], capturing topology of the space from which data points are sampled [2], and studying various aspects of data geometry including manifold learning have been proposed [3]. However in spite of that progress, there are still many important open problems and unexplored areas in data modeling, particularly for nonlinear spaces and systems. In this paper, we address that gap with applications in robotics as robots are one of the best examples of ubiquitous nonlinear systems.

Principal Component Analysis (PCA), in which data points are modeled over a linear or affine space, is the most well-known unsupervised data analysis method for dealing with high-dimensional data. However, over decades, its intrinsic deficiencies particularly in nonlinear regimes has motivated research towards alleviating the corresponding shortcomings. Geometrically, the developed methods can be classified into two rudimentary categories: distance preservation and topology preservation (see [4] for an inclusive review). Distance-preserving approaches, depending on the selected metric, are based on pairwise distances of the given data points. Technically, all of them strive to solve a maximization problem of finding top eigenvectors, which construct the embedding space. Even though metric multidimensional scaling (MDS) [5] was proposed as the pioneering distance-preserving method, it is still linear like PCA. The first well-known works, in which the transformation is nonlinear, are nonmetric MDS [6] and Sammon’s nonlinear mapping (NAM) [7]. Thereafter, other
researchers attempted to amend those approaches by either stochastic techniques, such as curvilinear component analysis (CCA) [8], or even replacing Euclidean distance with other distance metrics, such as geodesic distance in [9]. Nevertheless, computing the geodesic distance is mathematically complex and, consequently, computationally intractable. Therefore, some other methods, like Isomap [10], and curvilinear distance analysis (CDA) [11], utilized graph distance metric instead to approximate it. Despite outperforming the Euclidean metric based approaches, these techniques are still incapable of capturing all possible manifolds. There are some other methods that cannot be classified into the aforementioned categories. Kernel PCA (KPCA) [12], semidefinite embedding (SDE), also known as maximum variance unfolding (MVU) [13], and generalized PCA [14] can be enumerated as some outstanding representatives of such approaches. Kernel PCA tries to overcome the nonlinearity of data using some predetermined kernel function which transforms the data into a higher-dimensional space. However, selecting the best-suited kernel function has always been problematic. To mitigate this issue MVU attempts to learn the best kernel from the given data. Vidal et al. [14] fit a union of linear subspaces into the data points that suffices to solve the motion segmentation problem in hand.

On the other hand, topology-preserving methods work with some kind of similarity measure rather than explicit use of pairwise distances. In contrast to distance-preserving methods, they are sturdy but arduous to implement. Kohonen’s self-organizing map (SOM) [15] can be addressed as the first prominent method in this category. Other subsequent works, such as generative topographic mapping (GTM) [16] (in which SOM is reformulated within probabilistic framework), locally linear embedding (LLE) [17], and Isotop [18], are among various endeavors to remedy the SOM inadequacies. Regardless of which category those methods belong to, almost none of them succeeds when the intrinsic dimensionality of the given data exceeds four [4]. Moreover, because most of them struggle to embed the data into a new space, loss of information seems inevitable applying these approaches. In this paper, we address these issues by introducing a new methodology, called Principal Variety Analysis (PVA), as a general framework for analytical modeling of given data points over an algebraic space with arbitrary dimension. PVA eschews embedding the data into a new space thus causes no information loss. These extracted kinematic models of the surrounding objects can be utilized by motion planners, for instance, for either better roadmap approximation [19] or coping with dynamicity of the environment [20, 21].

Considering the problem as variety learning, Heldt et al. [22] introduce a numerically stable approximate vanishing ideal algorithm. Livni et al. [23] focus on compact representation of the vanishing ideals. Király and Tomioka [24], Király et al. [25] outline ideal-kernel-duality to extract discriminative and generative components from the data and address some other alternatives for rank minimization problem. Comparing to PVA, none of these works concern about the dimensionality optimization of the resulting variety of the vanishing ideals.

The dimensionality reduction problem can be cast as providing a low-dimensional representation of the given data as well. Technically, multivariate polynomial interpolation (MPI) [26] can be contemplated as a formidable formulation to do so. Interpolation theory for univariate functions has been well-developed over a long period of time but in the case of multivariate functions it is still immature. The conventional way to sort out the univariate version of the problem goes through solving a Vandermonde linear system [27] or LU factorization of it [28]. Basically, the most conspicuous strategy to tackle the MPI problem is to generalize existing tools for univariate case, such as Vandermonde linear system and LU factorization, to the multivariate one. However, in most of the works that embrace this tactic the number of points determine the dimension of the interpolating manifold [28]. On the other hand especially in the polynomial case, the problem gets more complicated because, considering the Haar-Mairhubert-Curtis theorem [29], multivariate polynomials are incapable of forming a Haar space. The Padua points [30] and the corresponding interpolation method is a technique to handle this problem. Nevertheless, it gets unstable while computing the basic Lagrange polynomials. To resolve that, Chebyshev polynomials of second kind are attained for interpolation [31]. Interestingly, the closest approach to PVA is the Kronecker’s pioneering work [32]. There, it is assumed that the number of polynomials equals the number of variables, so a square nonlinear system of equations is solved symbolically to find the corresponding variety. First, comparing to PVA, the Kronecker’s method cannot be practically applied to high-dimensional problems because it is computationally intractable due to symbolic computations. Second, PVA does not put any restriction on the number of polynomials.

The data driven approach matches the robotics spirit as robots essentially interact with the real world while constantly sensing and modeling it. In the past, operating articulated objects has been
well-studied in many works [33] in which most of them are either model-free or have considerable prior knowledge about the model and its related parameters. However as robots are gradually introduced into more complex environments, kinematics of different kinds of objects which robots interact with and/or manipulate cannot be assumed to be necessarily known in advance. In general, a robot needs both dynamic and kinematic models of its surrounding objects to have a comprehensive understanding of its environment. The introduced framework in [34] struggles to extract a specific class of kinematic models, planar models, through graphical representation. In [35], models and architectures for controlling robot body and external controllable objects have been investigated. A recent work on dynamically modeling objects can be found in [36] in which it is applied to predict rigid body behavior from recorded trajectories of objects. The proposed approach employs kinematic optimization to find a feasible pose that satisfies all of the kinematic constraints. Probabilistic model learning is another way to model articulated objects kinematically in unstructured environments [37]. Nonetheless, it suffers from both maintaining a kinematic graph and performing an additional step to reduce the dimension. Furthermore, it is based on a limited set of pre-coded candidate models: rigid, prismatic, revolute, and Gaussian. Extracting kinematic model of objects’ structure is considered to be a potential intriguing application of PVA. However, PVA is a general purpose method which is expected to have many other important applications.

2 Technical Background

We are given \( D = \{p_1, p_2, \ldots, p_k\} \subseteq \mathbb{R}^n \) a set of data points that are assumed to lie on an unknown, low-dimensional, real algebraic variety \( W \). We assume there are no outliers in \( D \). The problem is to find \( V \), an approximation of \( W \) the variety from which data points are captured, through computing the polynomials \( f_1, f_2, \ldots, f_m \in \mathbb{R}[X] \) that define
\[
V = Z(f) = \{x \in \mathbb{R}^n \mid f_i(x) = 0, \ i = 1, 2, \ldots, m\},
\]
in which \( Z \) denotes the real zero set. Throughout this paper, we denote variables by \( X = (X_1, X_2, \ldots, X_n) \) and values by \( x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \).

3 Problem Formulation

It is clear that if we put no upper bound on the degrees of \( f_i \)'s, then \( V \) simply degenerates into a zero-dimensional point set on every \( p_j \). Hence, we assume that \( \deg(f_i) \leq d \) for \( i = 1, 2, \ldots, m \), and \( d \in \mathbb{N} \).

The points in \( D \) have to lie on \( V \), i.e. \( D \subseteq V \) if \( D \) is noiseless. For brevity and without loss of generality, we assume the data points are noiseless in the rest of this paper. Should it be necessary, we mention important changes in the presence of noise.

**Problem 1 (Principal Variety Analysis)** Given input data set \( D \subset \mathbb{R}^n \), the problem is formulated as the following optimization:
\[
\begin{align*}
\minimize_{V} & \quad \dim(V) \\
\text{subject to} & \quad V = \{x \in \mathbb{R}^n \mid f_i(x) = 0, \ i = 1, 2, \ldots, m\}, \\
& \quad D \subseteq V, \\
& \quad \deg(f_i) \leq d, \ 1 \leq i \leq m,
\end{align*}
\]
in which \( \dim \) represents the dimension of a variety and \( \deg \) is the degree of a polynomial.

Above, we need a practically powerful definition of variety dimension that can also be conveniently and efficiently computed. Below, we present our definition of variety dimension for which we will give an efficient algorithm in the following sections.

3.1 Dimension of a Real Algebraic Variety

In this paper, we use a geometric notion of the dimension of an algebraic variety, which serves our quest the best. Mathematically, the real dimension of a real algebraic set is the dimension of its Zariski closure. We use a slightly different definition here that can be computed more easily.
Definition 1 (Dimension of Variety) The dimension of a real algebraic variety \( V \), which is the zero set of \( f = (f_1, f_2, \ldots, f_m) \) real polynomials, is

\[
\dim(V) = \max_{p \in V} \left( n - \text{rank} \left. \frac{\partial f}{\partial x} \right|_p \right) = n - \min_{p \in V} \text{rank} \left. \frac{\partial f}{\partial x} \right|_p.
\]

Above, \( \frac{\partial f}{\partial x} \) is the Jacobian of \( f \). Therefore, \( \text{rank} \left. \frac{\partial f}{\partial x} \right|_p \) is the dimension of the orthogonal (co-tangent) space at point \( p \) and \( n - \text{rank} \left. \frac{\partial f}{\partial x} \right|_p \) is the dimension of the tangent space at point \( p \). Maximum tangent space dimension over all points of the variety gives the dimension of variety.

With this definition, Problem 1 is rewritten as

\[
\begin{align*}
\text{maximize} & \quad \min_{p \in V} \text{rank} \left. \frac{\partial f}{\partial x} \right|_p \\
\text{subject to} & \quad V = \{ x \in \mathbb{R}^n \ | \ f_i(x) = 0, \ i = 1, 2, \ldots, m \}, \\
& \quad D \subseteq V, \\
& \quad \text{deg}(f_i) \leq d, \ 1 \leq i \leq m.
\end{align*}
\]

However, solving the optimization (4) is not easy since rank minimization is essentially an \( \ell_0 \) norm minimization. The vast majority of rank (\( \ell_0 \) norm) minimization problems are NP-Hard, even in affine subspaces, and for most practical problems there is no efficient algorithm that yields an exact solution. This is predominantly because of the non-convex nature of the rank optimization problem [38]. Hence, we seek a proxy for the rank function that is guaranteed to yield the same optimal solution under reasonable sparsity assumptions and whose minimization is computationally tractable.

4 Sparsity Comes to Rescue

4.1 Nuclear Norm Minimization

Fazel et al. [39] proposed a popular heuristic algorithm that replaces the rank function with the nuclear norm in the matrix completion framework [40]. Nuclear norm of a matrix, in this case the Jacobian, is the sum of the matrix singular values. More precisely, let for the Jacobian matrix \( \frac{\partial f}{\partial x} \in \mathbb{R}^{m \times n} \) of \( f \), \( \sigma_i(\cdot) \) be the \( i \)th largest singular value. The nuclear norm of the matrix \( \frac{\partial f}{\partial x} \) is defined as

\[
\| \frac{\partial f}{\partial x} \|_* = \sum_{i=1}^{n} \sigma_i(\frac{\partial f}{\partial x}).
\]

Recht et al. [40] provided a necessary and sufficient condition that determines when this heuristic successfully finds the minimum rank solution of a linear constraint set [38].

4.2 Dimension of a Sparse Real Algebraic Variety

Recht et al. [40] showed that when the linear map defining the constraint set in rank minimization problem is restricted isometric, the nuclear norm is the best convex lower approximation of the rank function.

Although there are currently no proofs for our case of nonlinear constraints, our experimental results in this paper demonstrate that substitution of rank in (4) with nuclear norm is still legitimate, under some conditions, for nonlinear constraint sets. Replacing rank with the nuclear norm, Problem 1 is then rewritten as

\[
\begin{align*}
\text{maximize} & \quad \text{rank} \left. \frac{\partial f}{\partial x} \right|_p \\
\text{subject to} & \quad V = \{ x \in \mathbb{R}^n \ | \ f_i(x) = 0, \ i = 1, 2, \ldots, m \}, \\
& \quad D \subseteq V, \\
& \quad \text{deg}(f_i) \leq d, \ 1 \leq i \leq m.
\end{align*}
\]
\[ p^* = \arg\min_{p \in V} \| \frac{\partial f}{\partial x} \|_p. \] (7)

5 Algorithm

5.1 Inner Polynomial Optimization

The inner nuclear norm minimization in (7) can be written as a polynomial optimization on an algebraic (semialgebraic in the case of noisy data) set. Recall that the nuclear norm of a matrix \( J \) is
\[ \| J \|_* = \text{trace} \left( \sqrt{J^T J} \right), \] (8)
in which \( \sqrt{B} = C \) if and only if \( C^2 = B \). Hence for a given \( f = (f_1, f_2, \ldots, f_m) \) such that \( D \subseteq \mathcal{Z}(f) = V \), the nuclear norm minimization problem in (7) becomes finding the point for which the following optimization problem attains its solution:
\[ \begin{align*}
\text{minimize} & \quad \text{trace}(C) \\
\text{subject to} & \quad C^2 = \frac{\partial f}{\partial x} \bigg|_q \frac{\partial f}{\partial x} \bigg|_q, \\
f(q) &= 0, \\
q &\in \mathbb{R}^n. 
\end{align*} \] (9)

Note that this is the crucial phase of the algorithm for detecting possible singularities that would affect the dimensionality of the resulting variety.

5.2 Outer Optimization

Since the optimization problem formulated in (6) does not currently have a known closed form solution, we exploited a breadth-first technique to solve the outer optimization. To do so, all possible candidates can be formulated as null space of a homogeneous system of linear equations below.

For \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n \), we define
\[ \Lambda(d) := \{ \alpha \in \mathbb{N}^n | |\alpha| \leq d \}, \] (10)
in which \( |\alpha| := \alpha_1 + \alpha_2 + \ldots + \alpha_n \). Hence, a degree-constrained polynomial, \( f_i \), can be written as
\[ f_i = \sum_{\alpha \in \Lambda(d)} c_{\alpha,i} X^\alpha, \] (11)
where \( c_{\alpha,i} \)'s are coefficients of corresponding monomials in \( f_i \). Given the set \( D \), by substituting the \( D \) data points in (11) the homogeneous system of linear equations is constructed as
\[ Ac = 0, \] (12)
in which matrix \( A \) contains the numbers obtained by evaluating the \( D \) data points in \( (X^\alpha)_{\alpha \in \Lambda(d)} \) and \( c \) is the corresponding coefficients variable vector in (11). It is required to keep the number of degree-constrained polynomials less than or equal to the number of elements in \( \Lambda(d) \) that have nonzero corresponding coefficient (which is usually the case in practice). Thus, the system of linear equations (12) is guaranteed to be never over-determined. The general solution for (12) is written as
\[ c = \lambda_1 n_1 + \lambda_2 n_2 + \ldots + \lambda_k n_k, \] (13)
where \( n_1, n_2, \ldots, n_k \) is a basis for the kernel of \( A \) and \( \lambda_i \)'s are picked randomly. The resulting coefficient vector, \( c \), is utilized in (11) to produce a particular polynomial out of the possible candidates. PVA commences exploring for a lowest dimensional variety by selecting \( m = n - 1 \). If it fails to find a solution it will seek for a higher dimensional variety incrementally (See Algorithm 1 in Appendix for more details).
Figure 1: The PVA constituent polynomials for the location of (a) the microwave door corner vs a fixed feature on the body (LED display) and (b) the drawer corner vs a fixed feature on the counter. The PVA variety of (c) the microwave and (d) the drawer.

6 Experimental Results

We developed a prototype implementation of PVA in MATLAB as a proof of concept. Faster implementations are conceivable using more advanced optimization libraries written in C. PVA was run on a machine with 64 AMD Opteron 6386 SE 2.8 GHz cores and 512 GB RAM. The resulting optimization problems were solved using YALMIP toolbox [41]. Experiments were performed in two groups.

First, we selected some well-known complex test cases and fed them into the inner (polynomial) optimization step of the PVA to examine its performance. We scrutinized the inner nuclear norm minimization on three major scenarios: (i) curves embedded in $\mathbb{R}^3$, (ii) surfaces in $\mathbb{R}^3$, and (iii) spheres in n-dimensional spaces. The experiments show that PVA correctly computes the dimension of input varieties within a few seconds (see Table 2 in Appendix for more details).

Second, we performed experiments on three different objects to illustrate performance of the entire PVA including the outer optimization:

1. Microwave Door: We put a marker on the corner of a home microwave door and recorded ten 3D coordinates of the marker against a fixed feature on the microwave body, the LED display. The data points in $D$ were $(p_{\text{door}}, p_{\text{led}}) \in \mathbb{R}^6$ in this case. Fig. 1 (a) and (c) illustrate the PVA variety with the constituent polynomials.

2. Drawer: We put a marker on the corner of a drawer and recorded ten 3D coordinates of the marker against a fixed feature on the counter. The data points in $D$ were $(p_{\text{drawer}}, p_{\text{counter}}) \in \mathbb{R}^6$ in this case. Fig. 1 (b) and (d) illustrate the PVA variety with the constituent polynomials.

3. Puma 560: We simulated trajectory of a 6-DOF Puma 560 robot using Robotics Toolbox [42]. In each experiment, ten configurations were sampled from the trajectory and the 3D coordinates of the joints were used as data points. The data points in $D$ were $(p_1, p_2, p_3, p_4, p_5, p_6) \in \mathbb{R}^{18}$ where $p_i$ is the 3D coordinates of the $i^{th}$ joint. First, two completely random experiments were executed. In one of them, eight coordinates of the 18 coordinates in $D$ were constant and in the other, all of the 18 coordinates were varying. Therefore, we performed PVA both on all 18 coordinates and the 10 varying coordinates respectively in $\mathbb{R}^{18}$ and $\mathbb{R}^{10}$. Figures 2 and 3 (a) illustrate the ten sampled configurations on the robot trajectory and the end effector location computed from the one dimensional PVA variety on which the features (joint coordinates) lie, respectively. Second, we conducted ten random independent experiments and collected ten samples of each experiment. Figure 3 (b) demonstrates the two dimensional PVA variety on which the end effector in all ten random experiments lies.

In all experiments, the resulted variety and the corresponding polynomials exactly pass through the given data points. Table 1 shows the running time, dimension of output variety, and the number of polynomials for each of the experiments above. Performance of the inner optimization problem solver was verified over hundreds of ersatz and well-known varieties. The algorithm was impeccable in determining the dimension of all the test cases in palatable amount of time.
7 Conclusion and Future Work

Principal Variety Analysis (PVA) was introduced in this paper as a general framework for primarily nonlinear data modeling. Its performance was demonstrated for modeling objects kinematics. Unlike the other object modeling approaches, which are based on recorded trajectories of objects, PVA just exploits the recorded coordinates of some pre-specified features on the objects. Therefore, it requires much less information to provide an analytical kinematic model of the objects, even in unstructured, uncertain environments. Furthermore, in contrast to local interpolation approaches, PVA can potentially reduce the effect of noise and non-smooth data. Our framework accommodates two methods to handle noise:
Table 1: Running Times, Dimensions, and Number of Polynomials for the PVA Experiments (the maximum polynomial degrees were set to four in all experiments).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>No. of Polys</th>
<th>Output Dim</th>
<th>Running Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microwave</td>
<td>2</td>
<td>1</td>
<td>0.65</td>
</tr>
<tr>
<td>Drawer</td>
<td>2</td>
<td>1</td>
<td>1.15</td>
</tr>
<tr>
<td>Puma 560 (10D)</td>
<td>9</td>
<td>1</td>
<td>169.51</td>
</tr>
<tr>
<td>Puma 560 (18D)</td>
<td>17</td>
<td>1</td>
<td>18861.17</td>
</tr>
<tr>
<td>Puma 560 (18D)</td>
<td>16</td>
<td>2</td>
<td>13401.72</td>
</tr>
<tr>
<td>Puma 560 (18D)</td>
<td>15</td>
<td>3</td>
<td>13342.30</td>
</tr>
<tr>
<td>Puma 560 (18D)</td>
<td>14</td>
<td>4</td>
<td>13512.64</td>
</tr>
<tr>
<td>Puma 560 (18D)</td>
<td>13</td>
<td>5</td>
<td>13891.89</td>
</tr>
<tr>
<td>Puma 560 (18D)</td>
<td>12</td>
<td>6</td>
<td>13231.88</td>
</tr>
<tr>
<td>Puma 560 (18D, 10 random experiments)</td>
<td>16</td>
<td>2</td>
<td>14358.87</td>
</tr>
</tbody>
</table>

- **Implicit**: Replace $D \subseteq V$ with adding a Lagrange multiplier for the constraints,

$$\lambda \sum_{i=1}^{m} \sum_{j=1}^{k} f_{i}^{2}(p_{j}),$$

to the objective function, i.e. minimizing distance from the fitted variety.

- **Explicit**: Allow the data points to be off of $V$ by a small distance, i.e. $f_{i}^{2}(p_{j}) \leq \epsilon$ for $i = 1, 2, \ldots, m, j = 1, 2, \ldots, k$, and a small $\epsilon \geq 0$.

Both methods above would work well with the PVA framework and left as an interesting research for future work.

Besides, PVA is not restricted to just some predetermined model templates of objects and is capable of extracting any kind of model the object may possess. In addition to finding kinematic model of objects, PVA can be considered as a powerful tool to estimate their corresponding Degrees of Freedom (DOFs). Its computational success depends on dimension minimization through replacement of intractable $\ell_{0}$ norm (rank) with tractable $\ell_{1}$ norm (nuclear norm). Previous results prove validity of that replacement for linear spaces. Our simulation results demonstrate that substitution still holds for nonlinear spaces. A formal proof is beyond the scope of this paper and remains as future work.

References


Algorithm 1 PVA algorithm

1: procedure PVA(dataPoints, spaceDim, maxDegree, range)
2:   pointsNo ← size(dataPoints, 2)
3:   n ← spaceDim + spaceDim * spaceDim
4:   \( \vec{x} \leftarrow \text{sdpvar}(1, n) \)
5:   monomials ← monolist(\( \vec{x}(1 : \text{spaceDim}) \), maxDegree)
6:   B ← \emptyset
7:   for i ← 1, pointsNo do
8:       assign(\( \vec{x}(1 : \text{spaceDim}) \), dataPoints(i))
9:       B ← B \cup \text{new row value(monomials)}^T
10:   end for
11:   basis ← nullSpace(B)
12:   coeffNo ← size(basis, 1)
13:   basisNo ← size(basis, 2)
14:   polyNo ← spaceDim
15:   solved ← False
16:   repeat
17:       polyNo ← polyNo − 1
18:       for i ← 1, polyNo do
19:           \( \vec{\lambda} \leftarrow \text{rand}(1, \text{range}, \text{basisNo}) \)
20:          \( \vec{f} \leftarrow \text{rand}(1, \text{range}, \text{basisNo}) \)
21:          for j ← 1, basisNo do
22:              \( \text{coeffs} \leftarrow \text{coeffs} \cup \vec{\lambda}(j) * \text{basis}(\cdot, j) \)
23:          end for
24:          \( f(i) \leftarrow \text{coeffs}^T * \text{monomials} \)
25:       end for
26:   C ← reshape(\( \vec{x}(\text{spaceDim} + 1 : n) \), [spaceDim, spaceDim])^T
27:   obj ← trace(C)
28:   F ← \emptyset
29:   for i ← 1, polyNo do
30:       F ← F \cup (f(i) \geq 0)
31:       F ← F \cup (-f(i) \geq 0)
32:   end for
33:   J ← jacobiann([f(1 : polyNo)]^T, \( \vec{x}(1 : \text{spaceDim}) \))
34:   constraints = C^2 − J^T * J
35:   for i ← 1, spaceDim do
36:       for j ← 1, spaceDim do
37:           F ← F \cup (constraints(i, j) \geq 0)
38:           F ← F \cup (-constraints(i, j) \geq 0)
39:       end for
40:   end for
41:   solved ← optimize(F, obj)
42:   until solved = True
43:   eval ← \text{replace}(J, \( \vec{x}(1 : \text{spaceDim}) \), value(\( \vec{x}(1 : \text{spaceDim}) \)))
44:   varietyDim ← spaceDim − \text{rank(full(eval))}
45:   return varietyDim, f
46: end procedure
Table 2: Determination of Dimension for Test Case Varieties in the Inner Polynomial Optimization Step of the PVA. For numerical considerations, all varieties are centered at \((1, \ldots, 1)\).

<table>
<thead>
<tr>
<th>Case</th>
<th>Algebraic Formula</th>
<th>Dim Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clover</td>
<td>((x - 1)^4 + 2(x - 1)^2(y - 1)^2 + (y - 1)^4 - (x - 1)^3 + 3(x - 1)(y - 1)^2)</td>
<td>9.4529</td>
</tr>
<tr>
<td>Ampersand</td>
<td>([(y - 1)^2 - (x - 1)^2][x - 2][2(x - 1) - 3])</td>
<td>– 1</td>
</tr>
<tr>
<td>Bicuspids in (\mathbb{R}^3)</td>
<td>[4[(x - 1)^2 + (y - 1)^2 - 2(x - 1)^2][(x - 1)^2 - 1] + [(y - 1)^2 - 1] ]</td>
<td>1 0.1332</td>
</tr>
<tr>
<td>Bow</td>
<td>((x - 1)^4 + 4(1 - x)^2 + (y - 1)^3)</td>
<td>1 6.2361</td>
</tr>
<tr>
<td>Elliptic</td>
<td>((y - 1)^2 - (x - 1)^3 + x - 2)</td>
<td>1 1.0470</td>
</tr>
<tr>
<td>Watt’s Curve</td>
<td>[4(y - 1)^2 + (y - 1)^3] [(x - 1)^2 + (y - 1)^2 - 1]</td>
<td>1 0.3598</td>
</tr>
<tr>
<td>Calypso</td>
<td>((x - 1)^2 + (y - 1)^2(z - 1) - (z - 1)^2)</td>
<td>2 1.2318</td>
</tr>
<tr>
<td>Dingdong</td>
<td>((x - 1)^2 + (y - 1)^2(z - 1)^2)</td>
<td>2 6.3090</td>
</tr>
<tr>
<td>Distel</td>
<td>[1000[(x - 1)^2 + (y - 1)^2][x - 1][x - 1][z - 1]^2]</td>
<td>2 8.6266</td>
</tr>
<tr>
<td>Durchblick</td>
<td>[2(x - 1)(y - 1) + (x - 1)(z - 1)^3]</td>
<td>2 7.4589</td>
</tr>
<tr>
<td>Eistüte</td>
<td>[(x - 1)^2 + (y - 1)^2]^3 [4(x - 1)^2(y - 1)^2]</td>
<td>2 8.5421</td>
</tr>
<tr>
<td>Eve</td>
<td>[5(x - 1)^2 + 2(1 - x)(1 - y)^2 + 5(y - 1)^6]</td>
<td>2 5.5867</td>
</tr>
<tr>
<td>Flirt</td>
<td>[15(y - 1)^2 + 5(z - 1)^2 - 15(y - 1)^3 - 5(y - 1)^2]</td>
<td>2 18.1745</td>
</tr>
<tr>
<td>Geisha</td>
<td>[(x - 1)^2(y - 1)(z - 1) + (x - 1)^2(z - 1)^2]</td>
<td>2 3.9184</td>
</tr>
<tr>
<td>Harlein</td>
<td>[(x - 1)^2(z - 1) + 10(x - 1)^2(y - 1)]</td>
<td>2 7.4701</td>
</tr>
<tr>
<td>Helix</td>
<td>[6(x - 1)^2 - 2(x - 1)^2 - (y - 1)^2]</td>
<td>2 7.1149</td>
</tr>
<tr>
<td>Herz</td>
<td>[6(y - 1)^2 + (z - 1)^2 - (z - 1)^2]</td>
<td>2 6.5586</td>
</tr>
<tr>
<td>Kolibri</td>
<td>[10(y - 1)^2 + (z - 1)^2 - (y - 1)^2]</td>
<td>2 27.0135</td>
</tr>
<tr>
<td>Leopard</td>
<td>[10(y - 1)^2 + (z - 1)^2 + (z - 1)^2]</td>
<td>2 8.5942</td>
</tr>
<tr>
<td>Octdong</td>
<td>[(x - 1)^2(y - 1)^2 + (z - 1)^4 - (z - 1)^2]</td>
<td>2 2.0404</td>
</tr>
<tr>
<td>Plop</td>
<td>[(x - 1)^2 + [(z - 1) - (y - 1)]] [3]</td>
<td>2 0.3425</td>
</tr>
<tr>
<td>Sofa</td>
<td>[(x - 1)^2 + (y - 1)^3 + (z - 1)^3]</td>
<td>2 6.6046</td>
</tr>
<tr>
<td>Torus</td>
<td>[16[(x - 1)^2 + (y - 1)^2]]</td>
<td>2 7.1438</td>
</tr>
</tbody>
</table>

\(^1\) Curves in \(\mathbb{R}^3\) have been intersected by \(z\) plane.