Lie Derivative Inclusion with Polynomial Output Feedback*

Tsuyoshi Yuno† and Toshiyuki Ohtsuka‡

In this paper, we deal with two problems of static output feedback for input-affine polynomial dynamical systems. One is to design a static output-feedback controller so as to render a prescribed algebraic set invariant for the resulting closed-loop system. The other is to design a static output-feedback controller so as to realize a prescribed vector field on a prescribed algebraic set. It is shown that the two problems can be represented by a particular inclusion of polynomials, and the inclusion can be solved. As a result, all the static output-feedback controllers required in the problems can be exactly represented by using free polynomial parameters.

1. Introduction

In this paper, we deal with two problems of static output feedback. One aims to control a dynamical system so as to render a prescribed subset of the state space invariant for the resulting closed-loop system, and the other aims to control a dynamical system so as to realize a prescribed vector field on a prescribed subset of the state space. It will be shown that the two problems can be represented by a particular inclusion of polynomials, which is called the Lie derivative inclusion, and the inclusion can be solved. By using this representation, all the controllers required in the problems can be exactly computed in a unified manner, even though the problems are totally different from each other.

A special class of the Lie derivative inclusion has been formulated and solved in [2] within the context of polynomial-type static state feedback for polynomial dynamical systems. The literature showed that the Lie derivative inclusion is an important concept in control theory because the above two problems have several control applications. For example, when the state of a controlled system must belong to a given constraint set, this constraint is always satisfied by a controller rendering the constraint set invariant. Such a controller can also solve the model matching problem. A controller realizing a prescribed vector field on a prescribed subset can be used to realize a prescribed periodic orbit and to solve a path-following control problem. See [2] for details.

This paper is an extension of [2] to the case of static output feedback. However, the solution method presented in [2] cannot be applied to solving the Lie derivative inclusion within the context of static output feedback because a solution of a linear equation over a subalgebra is needed to solve the inclusion unlike the case of state feedback (see Subsection 3.4 for details). Although an algorithm for partially solving such an equation has been proposed as a SINGULAR [3,4] package, it requires very strict assumptions regarding the cardinality of generators of the subalgebra and the algebraic independence of the generators. To the best of the authors’ knowledge, any algorithm for fully solving a linear equation over a subalgebra has never been proposed. Hence, in this paper, we will give an algorithm for solving the Lie derivative inclusion with static output feedback through more involved techniques than [2]. The algorithm implicitly gives a procedure for solving a linear equation over a subalgebra. As a result, all the static output-feedback controllers solving the Lie derivative inclusion can be exactly computed. We also show that the realization of a prescribed vector field via static output feedback can be applied to the replacement of a state feedback controller by a static output-feedback controller. This application does not appear in the case of state feedback.

As is done in [2], we assume that the given system is a polynomial dynamical one and that the controller is of polynomial-type static output feedback. Moreover, we assume that the given subset of the state space is algebraic. As is well known, polynomial dynamical systems can represent a variety of objects [5].

* Manuscript Received Date: April 30, 2014
† Graduate School of Engineering Science, Osaka University; 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, JAPAN
‡ Graduate School of Informatics, Kyoto University; Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, JAPAN
Key Words: polynomial dynamical systems, nonlinear systems, invariant set, vector field, commutative algebra.
In particular, many non-polynomial dynamical systems can be described as polynomial dynamical systems via immersion [6]. Moreover, algebraic sets can represent a variety of typical subsets, e.g., lines, hyperplanes, ellipses, and spheres. Therefore, the above assumptions are reasonable as a first step in tackling the problems.

This paper is organized as follows. First, the above two problems associated with the Lie derivative inclusion are mathematically formulated in Section 2. The notation used in this paper is also described. Section 3 explains that the problems can be represented as the Lie derivative inclusion. Moreover, the difficulty in solving the Lie derivative inclusion with static output feedback is explained. We give the algorithm (Algorithm 1) for solving the inclusion and prove Theorem 2 to show that the algorithm indeed solves the inclusion. As a result, all the static output-feedback controllers solving the inclusion can be exactly represented by using free polynomial parameters. An example given in Section 4 demonstrates the implementation of Algorithm 1. Appendix A.1 gives the algorithm for computing the set of the polynomial vectors required in Algorithm 1 and the lemma required for proving Theorem 2. In Appendix A.2, several well-known algorithms employed for Algorithm 1 are shown for the sake of completeness.

This paper is written in the language of commutative algebra [7,8] and Gröbner bases [9–11]. Understanding it requires some knowledge of these topics.

2. Mathematical Preliminaries and Motivating Problems

In this section, we define the notation used in this paper and describe the system examined herein. After that, we mathematically formulate the problems mentioned in the introduction. The first problem is to obtain a static output-feedback controller such that a prescribed algebraic set in the state space is invariant for the resulting closed-loop system. The second problem is to obtain a static output-feedback controller such that the resulting closed-loop system has a prescribed vector field on a prescribed algebraic set in the state space.

2.1 Notation and System Description

Throughout the paper, we use the following notation. Let $R$ denote the field of real numbers. Let $R = R[X_1, \ldots, X_n]$ be a polynomial ring over $R$ with indeterminates $X_1, \ldots, X_n$. Let $S = R[Y_1, \ldots, Y_p]$ be a polynomial ring over $R$ with indeterminates $Y_1, \ldots, Y_p$. For a given positive integer $s$, let $R^s$ denote the free $R$-module of all the column vectors with $s$ components in $R$. We will consider only the standard operations in $R^s$. For given positive integers $s_1$ and $s_2$, let $R^{s_1 \times s_2}$ denote the set of all the $s_1 \times s_2$ matrices with entries in $R$. For given polynomial vectors $m_1, \ldots, m_r \in R^n$ (resp. polynomials $m_1, \ldots, m_r \in R$), let $\langle m_1, \ldots, m_r \rangle_R$ denote the $R$-submodule of $R^n$ (resp. the ideal of $R$) generated by the set of generators $\{m_1, \ldots, m_r\}$. That is, let $\langle m_1, \ldots, m_r \rangle_R := \{c_1 m_1 + \cdots + c_r m_r : c_i \in R \text{ for } i = 1, \ldots, r\}$. For a given polynomial vector $v \in R^n$, we distinguish between a polynomial vector $v$ and a polynomial map $v(\cdot)$ according to these notations. We further distinguish between the indeterminates $X_1, \ldots, X_n$ of $v$ and the variables $x = [x_1, \ldots, x_n]^T \in R^n$ of $v(\cdot)$. The above notations are also defined for the ring $S$ as well as $R$. For a given polynomial $\Phi \in R$, the derivation of $\Phi$ w.r.t. $X_i$ is denoted by $\partial \Phi / \partial X_i$. Let $\partial \Phi / \partial X$ be the row vector $[\partial \Phi / \partial X_1, \ldots, \partial \Phi / \partial X_n]$. For a given polynomial $\Phi \in R$ and a polynomial vector $f \in R^n$, let $L_f \Phi := (\partial \Phi / \partial X)f$.

The following notations are commonly used in (real) algebraic geometry [12].

**Definition 1** Let $\mathcal{A}$ be a given algebraic set in $R^n$. The vanishing ideal of $\mathcal{A}$ is the ideal $I$ of $R$ consisting of all the polynomials vanishing on $\mathcal{A}$, i.e.,

$$I := \{\rho \in R : \rho(x) = 0 \forall x \in \mathcal{A}\}.$$ 

**Definition 2** Let $J$ be a given ideal of $R$. The real radical of $J$ is the ideal

$$\sqrt{J} := \left\{\rho \in R : \exists N \in \mathbb{N}, \exists \lambda_1, \ldots, \lambda_r \in R, \rho^{2N} + \lambda_1^2 + \cdots + \lambda_r^2 \in J\right\},$$

where $\mathbb{N}$ denotes the natural numbers.

Let $J$ be an ideal of $R$ representing a given algebraic set $\mathcal{A} \subset R^n$, i.e., an ideal such that $\mathcal{A} = \{x \in R^n : \rho(x) = 0 \forall \rho \in J\}$. It is well known [12] that the vanishing ideal $I$ of $\mathcal{A}$ is the real radical of $J$, i.e., $I = \sqrt{J}$. Several algorithms for computing the real radical of a given ideal are given in [13–16]. We henceforth assume that the vanishing ideal $I$ of a given algebraic set $\mathcal{A}$ can be computed.

We will deal with a system of the form

$$\begin{align*}
\dot{x} &= f^o(x, u) = f(x) + G(x)u, \\
y &= h(x)
\end{align*}$$

with $f \in R^n$, $G \in R^{n \times m}$, and $h = [h_1, \ldots, h_p]^T$, $h_i \in R$ ($i = 1, \ldots, p$). We write $f^o(\cdot, \cdot) = [f_1^o(\cdot, \cdot), \ldots, f_n^o(\cdot, \cdot)]^T$.

2.2 Controlled Invariance of a Prescribed Algebraic Set

Consider an autonomous system of the form

$$\dot{x} = F(x)$$

with $F \in R^n$. Let $\phi(t, x_0)$ denote the solution of eq. (2) at time $t$ with the initial condition $x(0) = x_0 \in R^n$. Denote by $E(x_0)$ the maximal existence interval of $\phi(\cdot, x_0)$. We define the notion of the invariance of a set as follows.

**Definition 3** A set $\mathcal{A} \subset R^n$ is said to be invariant for system (2) if $x_0 \in \mathcal{A}$ implies $\phi(t, x_0) \in \mathcal{A}$ for all $t \in E(x_0)$. Now, let us return to system (1) and define the notion of controlled invariance by static output feed-
A set $\mathcal{A} \subset \mathbb{R}^n$ is said to be controlled invariant by static output feedback for system (1) if there exists a static output-feedback controller $u \in S^m$ such that $\mathcal{A}$ is invariant for the resulting closed-loop system of eq. (1).

We shall consider the following problem.

**Problem 1** Let $I = \langle p_1, \ldots, p_k \rangle \subset \mathbb{R}$ be the vanishing ideal of a given algebraic set $\mathcal{A} \subset \mathbb{R}^n$. Then, given a system of form eq. (1), determine whether $\mathcal{A}$ is controlled invariant by static output feedback for system (1). Moreover, if $\mathcal{A}$ is controlled invariant by static output feedback, obtain a static output-feedback controller $u \in S^m$ for system (1) such that $\mathcal{A}$ is invariant for the resulting closed-loop system.

**Definition 5** A solution of Problem 1 is a static output-feedback controller $u \in S^m$ such that $\mathcal{A}$ is invariant for the resulting closed-loop system of eq. (1). Problem 1 is said to be solvable if a solution to it exists.

As stated in [2], the problem of state constraint and that of model matching can be represented by Problem 1.

### 2.3 Realization of a Prescribed Vector Field

The second problem is the realization of a prescribed vector field. It is described as follows.

**Problem 2** Let $I = \langle p_1, \ldots, p_k \rangle \subset \mathbb{R}$ be the vanishing ideal of a given algebraic set $\mathcal{A} \subset \mathbb{R}^n$. Given a system of form eq. (1) and a polynomial vector $f^i = [f_1^i, \ldots, f_n^i] \in \mathbb{R}^n$ with a given positive integer $n' \leq n$, determine whether there exists a static output-feedback controller $u \in S^m$ such that

$$f_1^i(x) = f_1^i(x, u(x)), \quad \vdots \quad f_n^i(x) = f_n^i(x, u(x)),$$

for all $x \in \mathcal{A}$. Moreover, if such a controller exists, it should be able to be computed explicitly.

**Definition 6** A solution of Problem 2 is a static output-feedback controller $u \in S^m$ satisfying eq. (3). Problem 2 is said to be solvable if a solution to it exists.

As stated in [2], the problem of realizing a prescribed periodic orbit and that of a path-following control can be represented by Problem 2. We give another problem represented by Problem 2 in the following example, which is the replacement of a state feedback controller by a static output-feedback controller.

**Example 1** (Replacement of a state feedback controller by a static output-feedback controller) Consider system (1) and suppose that a state feedback controller $u_X \in \mathbb{R}^m$ for the system has been given. Put

$$f^i := f + Gu_X \quad \text{and} \quad I := \{0\}.$$

Then, Problem 2 represents the problem of obtaining a static output-feedback controller $u \in S^m$ such that the vector field of the resulting closed-loop system coincides with that of the closed-loop system with $u_X$.

### 3. Lie Derivative Inclusion and Its Solution

In this section, we show that Problems 1 and 2 can be represented by the Lie derivative inclusion with static output feedback. Moreover, the difficulty in solving the inclusion is explained. Algorithm 1 is the procedure for solving the inclusion. Theorem 2 shows that Algorithm 1 indeed solves the inclusion. As a result, all the static output-feedback controllers required in the problems can be represented by using free polynomial parameters.

#### 3.1 Lie Derivative Inclusion

In this subsection, we define the inclusion to be solved. It will be shown in later subsections that the two problems described in the previous section can be represented by the inclusion.

For a given positive integer $r$, let $\text{Sbs}_{Y \mapsto h} : \mathbb{R}^r \rightarrow \mathbb{R}^r$ denote the map defined by the substitution $Y_i \mapsto h_i (i = 1, \ldots, p)$. We define the Lie derivative inclusion as follows.

**Definition 7** Let $\Phi_i, \Psi_i (i = 1, \ldots, \ell)$ be given polynomials in $R$ and $I$ be a given ideal of $R$. Then, consider the system of inclusions

$$L_{f + \text{Sbs}_{Y \mapsto h}^m(u)} \Phi_i \in \Psi_i + I,$$

$$L_{f + \text{Sbs}_{Y \mapsto h}^m(u)} \Phi \in \Psi + I$$

with $u \in S^m$ to be found. We call the above system the Lie derivative inclusion. A solution of the Lie derivative inclusion is a static output-feedback controller $u \in S^m$ satisfying eq. (4). The Lie derivative inclusion is said to be solvable if a solution to it exists.

(Remark 1) In [2], ideals are given independently for each inclusion in eq. (4). However, to solve Problems 1 and 2, it suffices to use a single ideal $I$ as inclusion (4). This treatment allows eq. (4) to be calculated in the factor ring $\mathbb{R} = R/I$. See the proof of Theorem 2 in Subsection 3.5.

#### 3.2 Representation of Problem 1

This subsection demonstrates how Problem 1 is represented by the Lie derivative inclusion. Let $\mathcal{A} \subset \mathbb{R}^n$ be a given algebraic set and $I = \langle p_1, \ldots, p_k \rangle \subset \mathbb{R}$ be the vanishing ideal of $\mathcal{A}$. A necessary and sufficient condition for the invariance of an algebraic set is given by the following theorem [17–19].

**Theorem 1** A given algebraic set $\mathcal{A}$ is invariant for system (2) if and only if

$$L_{FP_i} \in I.$$
holds for $i = 1, \ldots, k$.

The following corollary is a straightforward consequence of the above theorem.

**Corollary 1** A given algebraic set $\mathcal{A}$ is invariant for the resulting closed-loop system of eq. (1) with a controller $u \in S^{m_i}$ if and only if $u$ satisfies the inclusion

$$L_f + GS_{y \rightarrow h}^m(u) \rho_i \in I.$$  

From this corollary, we have that the solutions of **Problem 1** coincide with the solutions of inclusion (4) with $\Phi_i = \rho_i$ and $\Psi_i = 0$ for $i = 1, \ldots, k$.

### 3.3 Representation of **Problem 2**

This subsection demonstrates how **Problem 2** is represented by the Lie derivative inclusion. Let $\{e_1, \ldots, e_n\}$ be the canonical basis of $R^n$ and rewrite eq. (3) as follows:

$$f_i^1(x) = e_i^T (f(x) + Gx(u(h(x)))),$$

$$f_i^m(x) = e_i^T (f(x) + Gx(u(h(x))))$$

**Problem 2** requires the above set of equations to be satisfied for all $x \in \mathcal{A}$. This is equivalent to the following inclusions of polynomials:

$$e_i^T (f + GS_{y \rightarrow h}^m(u)) \in f_i + I,$$

$$e_i^T (f + GS_{y \rightarrow h}^m(u)) \in f_i + I.$$  

The above inclusions can be derived from inclusion (4) by setting $\Phi_i = X_i$ and $\Psi_i = f_i^d$ for $i = 1, \ldots, n'$. That is, the solutions of **Problem 2** coincide with the solutions of inclusion (4) with $\Phi_i = X_i$ and $\Psi_i = f_i^d$ for $i = 1, \ldots, n'$.

### 3.4 Difficulty in Solving the Lie Derivative Inclusion with Static Output Feedback

The Lie derivative inclusion with static output feedback is more difficult to solve than that with state feedback. We will explain this in detail here.

Given positive integers $s$ and $r$, consider the linear equation

$$\gamma_0 = z_1 \gamma_1 + \cdots + z_r \gamma_r$$

with given polynomial vectors $\gamma_i \in R^n$ ($i = 0, \ldots, r$), where $z := [z_1, \ldots, z_r]^T \in R^r$ is to be found. The literature [2] showed that solving the Lie derivative inclusion with state feedback reduces to solving the above equation. Moreover, the above equation can be solved by using the technique of Gröbner bases (see [9–11]). On the other hand, in the case of static output feedback, the $z_i$'s are taken to be elements of $S$ rather than $R$. Although an algorithm for partially solving such an equation has been proposed as a SINGULAR package (see [3]), it requires very strict assumptions that $p = n$ holds and $\{h_1, \ldots, h_p\}$ is algebraically independent. To the best of the authors' knowledge, any algorithm for fully solving such an equation has never been proposed. Hence, in this paper, we will give an algorithm for solving the Lie derivative inclusion with static output feedback through more involved techniques than [2]. The algorithm implicitly gives a procedure for solving such equations where the $z_i$'s are taken to be elements in $S$.

### 3.5 Solution of the Lie Derivative Inclusion

Here, we will describe an algorithm for solving the Lie derivative inclusion with static output feedback. The algorithm can also be used to solve **Problems 1** and 2, as explained in the previous subsections.

Before we describe the algorithm, let us define the following notation. Let $\mathcal{N} = R/I$ be the factor ring of $R$ modulo $I$, and $H = R[h_1, \ldots, h_p]$ be the subalgebra of $R$ generated by $\{h_1, \ldots, h_p\}$. For any element $a \in R$, let $\pi$ denote the equivalence class $[a] \in \mathcal{N}$. Similarly, for any element $\xi = [\xi_1, \ldots, \xi_s] \in R^s$ with some integer $s \geq 1$, we use the same notation $\xi$ to describe the element $[\xi_1, \ldots, \xi_s] \in \mathcal{N}$. Let $R/I$ be the subalgebra $R[h_1, \ldots, h_p]$ of $R$ generated by $\{h_1, \ldots, h_p\}$. Furthermore, let $\text{Fct} : R^{m+1} \rightarrow R^{m+1}$ be the map defined by $R^{m+1} \ni \xi \mapsto \xi \in R^{m+1}$. For a given positive integer $r$, let $\text{Sbs}_{y \rightarrow h} : S^r \rightarrow R^r$ denote the map defined by the substitution $Y_i \mapsto h_i$ ($i = 1, \ldots, p$), as was defined before.

Let $P$ be either of the rings $R$, $S$, or $\mathcal{N}$. We further define the following notation. For a given ideal $J \subseteq P$ and a given integer $s \geq 1$, let $J^{\leq s}$ be the direct sum of $s$ copies of $J$, i.e.,

$$J^{\leq s} = \{ \xi = [\xi_1, \ldots, \xi_s] \in P^s : \xi_i \in J \ (i = 1, \ldots, s) \}.$$  

Note that, when $J$ is described by $J = \langle \mu_1, \ldots, \mu_k \rangle_P$ with generators $\{\mu_1, \ldots, \mu_k\}$, we can describe $J^{\leq s}$ in the explicit form

$$J^{\leq s} = \langle \mu_1 e_1, \ldots, \mu_k e_1 \rangle_P + \cdots + \langle \mu_1 e_s, \ldots, \mu_k e_s \rangle_P,$$

where $\{e_1, \ldots, e_s\}$ is the canonical basis of $P^s$. In particular, $J^{\leq s}$ is a $P$-submodule of $P^s$. For given polynomial vectors $v_i \in P^s$ ($i = 1, \ldots, r$) with some $r \geq 1$, let $\text{Syz}^P(v_1, \ldots, v_r) \subset P^r$ be the syzygy module of the $r$-tuple $(v_1, \ldots, v_r)$, i.e.,

$$\text{Syz}^P(v_1, \ldots, v_r) := \{ \zeta = [\zeta_1, \ldots, \zeta_r] \in P^r : \zeta_1 v_1 + \cdots + \zeta_r v_r = 0 \}.$$  

For given integers $s_1$ and $s_2$ such that $s_1 \geq s_2 \geq 1$, let $\text{Pr}_{s_1}^{s_2} : P^{s_1} \rightarrow P^{s_2}$ be the projection map defined by $P^{s_1} \ni [\omega_1, \ldots, \omega_{s_1}] \mapsto [\omega_1, \ldots, \omega_{s_2}] \in P^{s_2}$.

We will give the algorithm for solving inclusion (4) in what follows. As we will see in **Theorem 2**, the algorithm checks the solvability of eq. (4) and computes the set of all solutions of eq. (4). The proof of **Theorem 2** indicates that the solutions $u = [u_1, \ldots, u_m]^T \in S^m$ of eq. (4) coincide with the solutions of the inclu-
where \( \{e_1, \ldots, e_{m+1}\} \) is the canonical basis of \( S^{m+1} \), \( \eta_i \in S^{m+1} (i = 1, \ldots, \nu) \) are the polynomial vectors computed in Step 1 in Algorithm 1, and \( \kappa_i \in S (i = 1, \ldots, \nu') \) are the polynomials computed in Steps 2 and 3 in the algorithm. Hence, the algorithm for solving eq. (4) is based on solving the above inclusion. The algorithm is as follows.

[Algorithm 1] (For solving the Lie derivative inclusion)

**Given:** the polynomial vector \( f \in R^n \), the polynomial matrix \( G \in R^{n \times m} \), and the polynomials \( h_i \in R (i = 1, \ldots, p) \) of system (1). Polynomials \( \Phi_i \in R \) and \( \Psi_i \in R \) with \( i = 1, \ldots, \ell \), and an ideal \( I = (p_1, \ldots, p_k) \subset R \).

**Obtain:** a polynomial vector \( u^* \in S^m \) and a set \( \{m_1^u, \ldots, m_{\nu'}^u\} \) of polynomial vectors \( m_i^u \in S^m (i = 1, \ldots, \nu') \), where \( u^* + \langle m_1^u, \ldots, m_{\nu'}^u \rangle_S \) is the set of all solutions of eq. (4).

**Step 1** Compute a set \( \{\eta_{1,1}, \ldots, \eta_{i,m+1}\} \) of polynomial vectors \( \eta_i \in S^{m+1} (i = 1, \ldots, \nu) \) by using Algorithm 2 in Appendix A.1. Write \( \eta_i = [\eta_{i,1}, \ldots, \eta_{i,m+1}]^T \) \( (i = 1, \ldots, \nu) \).

**Step 2** Define the \( R \)-algebra homomorphism \( \pi : S \to R^m \) by

\[
S \ni \alpha(Y_1, \ldots, Y_{p}) \mapsto \alpha(h_1, \ldots, h_p) \in R^m.
\]  

**Step 3** Compute a set of generators \( \{\kappa_1, \ldots, \kappa_{\nu'}\} \) of \( \ker \pi \) by using Algorithm 5 in Appendix A.2.

**Step 4** Check if

\[
e_1 \in u_1 e_2 + \cdots + u_{m} e_{m+1} + \langle \eta_{1,1}, \ldots, \eta_{1,m+1} \rangle_S + \langle \kappa_1, \ldots, \kappa_{\nu'} \rangle_S.
\]

If the above inclusion holds, proceed to the next step because eq. (4) is solvable; otherwise, terminate the algorithm because eq. (4) is not solvable.

**Step 5** Let \( \{e_1, \ldots, e_{m+1}\} \) be the canonical basis in \( S^{m+1} \).

**Step 6** Set \( d = \nu + (m+1)\nu' \), and define the equation

\[
e_1 = z_1 e_2 + \cdots + z_m e_{m+1} + \langle z_{m+1} \eta_1, \ldots, z_{m+\nu} \eta_{\nu} \rangle + \langle z_{m+\nu+1} \kappa_1 e_1, \ldots, z_{m+\nu+\nu'} \kappa_{\nu'} e_1 \rangle + \cdots + \langle z_{m+d} \kappa_{\nu'} e_{m+1} \rangle\]

with the variables \( z := [z_1, \ldots, z_{m+d}]^T \in S^{m+d} \).

**Step 7** By solving the above equation using Algorithm 6 in Appendix A.2, obtain polynomial vectors \( z^* \in S^{m+d} \) and \( m_i^u \in S^{m+d} (i = 1, \ldots, \nu) \), where \( z^* + \langle m_1^u, \ldots, m_{\nu'}^u \rangle_S \) is the set of all solutions \( z \in S^{m+d} \) of the equation.

**Step 8** Put \( u^* := \text{Prj}_{S}^{m+d}(z^*) \in S^m \) and \( m_i^u := \text{Prj}_{S}^{m+d}(m_i^u) \in S^m \) \( (i = 1, \ldots, \nu) \).

(Remark 2) Checking if inclusion (6) holds is the ideal membership problem, which can be exactly solved \([9,20]\).

(Remark 3) All the calculations in the above algorithm are not approximate but exact.

We shall prove that the above algorithm indeed solves the Lie derivative inclusion.

[Theorem 2] Inclusion (4) is solvable if and only if eq. (6) holds. Moreover, \( u^* + \langle m_1^u, \ldots, m_{\nu'}^u \rangle_S \) is the set of all solutions of eq. (4).

(Proof) Let \( \pi : S \to R^m \) be the \( R \)-algebra homomorphism defined by eq. (5) as is in Algorithm 1. For a given integer \( s \geq 1 \), the free \( R \)-module \( \overline{S} \) can be regarded as an \( S \)-module with the law of action \( S \times \overline{S} \ni (\alpha, \zeta) \mapsto \pi(\alpha) \zeta \in \overline{S} \). We henceforth regard \( \overline{S} \) as an \( S \)-module in such a way. The \( R \)-algebra homomorphism \( \pi \) induces the \( S \)-module homomorphism \( \pi : S^{m+1} \to \overline{S}^{m+1} \) defined by

\[
[k_1, \ldots, k_{m+1}]^T \mapsto [\pi(k_1), \ldots, \pi(k_{m+1})]^T.
\]

Note that \( \overline{S} = \text{Fct}^t \circ S^{m+1} \otimes \text{Sbs}_{Y^\rightarrow h}^{m} \). Inclusion (4) can be transformed into

\[
\psi_1 - \frac{\partial \psi_1}{\partial X} f \in \left( \frac{\partial \psi_1}{\partial G} \right) S^{m+1} \otimes \text{Sbs}_{Y^\rightarrow h}^{m} + I,
\]

\[
\vdots
\]

\[
\psi_{\ell} - \frac{\partial \psi_{\ell}}{\partial X} f \in \left( \frac{\partial \psi_{\ell}}{\partial G} \right) S^{m+1} \otimes \text{Sbs}_{Y^\rightarrow h}^{m} + I.
\]

This is equivalent to the equation

\[
\psi_1 - \frac{\partial \psi_1}{\partial X} f = \left( \frac{\partial \psi_1}{\partial G} \right) S^{m+1} \otimes \text{Sbs}_{Y^\rightarrow h}^{m} u,
\]

\[
\vdots
\]

\[
\psi_{\ell} - \frac{\partial \psi_{\ell}}{\partial X} f = \left( \frac{\partial \psi_{\ell}}{\partial G} \right) S^{m+1} \otimes \text{Sbs}_{Y^\rightarrow h}^{m} u.
\]

Now, put

\[
\tilde{v} := \left[ \begin{array}{c} \psi_1 - \frac{\partial \psi_1}{\partial X} f \\ \vdots \\ \psi_{\ell} - \frac{\partial \psi_{\ell}}{\partial X} f \end{array} \right] \in \overline{R}^m,
\]

\[
\tilde{G} := [\tilde{g}_1, \ldots, \tilde{g}_m] := \left[ \begin{array}{c} \frac{\partial \psi_1}{\partial X} G \\ \vdots \\ \frac{\partial \psi_{\ell}}{\partial X} G \end{array} \right] \in \overline{R}^{m \times m},
\]

\[
u = \left[ \begin{array}{c} u_1 \\ \vdots \\ u_m \end{array} \right] \in S^m.
\]

Then, eq. (7) is transformed into
Hence, the solutions of eq. (4) are identical to the solutions \( u = [u_1, \ldots, u_m]^T \in S^m \) satisfying the equation

\[
\tilde{\psi} = u_1 \tilde{g}_1 + \cdots + u_m \tilde{g}_m
\]

with the law of action of the \( S \)-module \( \overline{R} \). Now, consider the \( S \)-submodule \( M := \langle \tilde{\psi}, \tilde{g}_1, \ldots, \tilde{g}_m \rangle_S \subset \overline{R} \).

Let \( \{e_1, \ldots, e_{m+1}\} \) be the canonical basis of \( S^{m+1} \), and define the surjective \( S \)-module homomorphism \( \chi : S^{m+1} \to M \) by

\[
e_1 \mapsto \tilde{\psi}, \quad e_2 \mapsto \tilde{g}_1, \quad \text{and} \quad e_{m+1} \mapsto \tilde{g}_m.
\]

By the homomorphism theorem, we have \( S^{m+1}/N \simeq M \), where \( N = \ker \chi \). Using this isomorphism, we have that eq. (9) is equivalent to the equation

\[
e_1 \in u_1 e_2 + \cdots + u_m e_{m+1} \mod N.
\]

That is, the solutions of eq. (4) are identical to the solutions \( u = [u_1, \ldots, u_m]^T \in S^m \) satisfying the inclusion

\[
e_1 \in u_1 e_2 + \cdots + u_m e_{m+1} + N.
\]

Now, let \( \{\eta_1, \eta_2, \ldots, \eta_d\} \) and \( \{\kappa_1, \kappa_2, \ldots, \kappa_{\nu'}\} \) be the sets computed in Step 1 and Steps 2–3 in Algorithm 1, respectively. Note that \( \langle \kappa_1, \ldots, \kappa_{\nu'} \rangle_S = \ker \pi \). Then, it can be verified that

\[
N = \langle \eta_1, \ldots, \eta_d \rangle_S + \langle \langle \kappa_1, \ldots, \kappa_{\nu'} \rangle_S \rangle_S^{m+1}
\]

as follows. By definition, we have

\[
N = \ker \chi
\]

\[
= \{ \zeta = [\zeta_1, \ldots, \zeta_{m+1}]^T \in S^{m+1} : \zeta_1 \tilde{\psi} + \zeta_2 \tilde{g}_1 + \cdots + \zeta_{m+1} \tilde{g}_m = 0 \}
\]

\[
= \{ \zeta = [\zeta_1, \ldots, \zeta_{m+1}]^T \in S^{m+1} : \pi(\zeta_1 \tilde{\psi} + \pi(\zeta_2 \tilde{g}_1 + \cdots + \pi(\zeta_{m+1}) \tilde{g}_m = 0 \}
\]

\[
= \{ \zeta \in S^{m+1} : \Pi(\zeta) \in \text{Syz} \tilde{\psi}, \tilde{g}_1, \ldots, \tilde{g}_m \cap \overline{R}^{m+1} \}
\]

\[
= \Pi^{-1}(\text{Syz} \tilde{\psi}, \tilde{g}_1, \ldots, \tilde{g}_m \cap \overline{R}^{m+1}).
\]

On the other hand, by Lemma 1 in Appendix A.1, we have

\[
\Pi(\langle \eta_1, \ldots, \eta_d \rangle_S) = \text{Syz} \tilde{\psi}, \tilde{g}_1, \ldots, \tilde{g}_m \cap \overline{R}^{m+1}.
\]

Hence, we have

\[
N = \langle \eta_1, \ldots, \eta_d \rangle_S + \text{ker} \Pi
\]

\[
= \langle \eta_1, \ldots, \eta_d \rangle_S + \langle \ker \pi \rangle_S^{m+1},
\]

which implies eq. (11). Now, let us consider eq. (10). The above expression for \( N \) implies that there exists a solution \( u \) satisfying eq. (10) if and only if the following inclusion holds:

\[
e_1 \in \langle e_2, \ldots, e_{m+1} \rangle_S + \langle \eta_1, \ldots, \eta_d \rangle_S
\]

\[
+ \langle \kappa_1 e_1, \ldots, \kappa_{\nu'} e_1 \rangle_S + \cdots
\]

\[
+ \langle \kappa_1 e_{m+1}, \ldots, \kappa_{\nu'} e_{m+1} \rangle_S.
\]

Noting that

\[
\langle e_2, \ldots, e_{m+1} \rangle_S \supset \langle \kappa_1 e_2, \ldots, \kappa_{\nu'} e_2 \rangle_S + \cdots + \langle \kappa_1 e_{m+1}, \ldots, \kappa_{\nu'} e_{m+1} \rangle_S,
\]

we have that eq. (12) is equivalent to

\[
e_1 \in \langle e_2, \ldots, e_{m+1} \rangle_S + \langle \eta_1, \ldots, \eta_d \rangle_S + \langle \kappa_1 e_1, \ldots, \kappa_{\nu'} e_1 \rangle_S.
\]

Write \( u_1 = [\eta_{i(1)}, \ldots, \eta_{i(m+1)}]^T \). Then, by comparing the positions of components, we have that the above inclusion is equivalent to the inclusion

\[
1 \in \langle \eta_{i(1)}, \ldots, \eta_{i(\nu)} \rangle_S + \langle \kappa_1, \ldots, \kappa_{\nu'} \rangle_S.
\]

Therefore, the first statement of the theorem follows. In what follows, we assume that eq. (4) is solvable. Put \( d := \nu + (m+1)\nu' \), and consider the equation

\[
e_1 = z_1 e_2 + \ldots + z_{m+1} e_{m+1}
\]

\[
+ (z_{m+1} \eta_1 + \cdots + z_{m+1} \eta_2)
\]

\[
+ (z_{m+1+1} \kappa_1 e_1 + \cdots + z_{m+1+\nu'} \kappa_{\nu'} e_1)
\]

\[
+ \cdots
\]

\[
+ (z_{m+1+\nu} \kappa_1 e_{m+1} + \cdots + z_{m+1+\nu} \kappa_{\nu'} e_{m+1})
\]

with the variables \( z := [z_1, \ldots, z_{m+\nu}]+ \in S^{m+\nu} \). A solution of eq. (10) is nothing but the image of the projection \( \Pi_m : S^{m+\nu} \to S^m \) of a solution \( z \) of eq. (13). Since eq. (13) is a linear equation with coefficients in the polynomial ring \( S \), we can exactly compute the set of all solutions of eq. (13) by Algorithm 6 in Appendix A.2 as

\[
z^* + \langle m_1^u, \ldots, m_{\nu'}^u \rangle_S
\]

where \( z^* \in S^{m+\nu} \) and \( m_1^u \in S^{m+\nu} \). Putting \( u^* := \Pi_m(z^*) \in S^m \) and \( m_1^u := \Pi_m(m_1^u) \in S^m \), we obtain the set \( u^* + \langle m_1^u, \ldots, m_{\nu'}^u \rangle_S \) of all solutions of eq. (10), i.e., the set of all solutions of eq. (4).

(Remark 4) Theorem 2 can be interpreted as follows: let \( u_1, \ldots, u_1 \) be as above. Then, any static output-feedback controller \( u \in S^m \) satisfying eq. (4) can be parametrized in the form

\[
u = u^* + c_1 m_1^u + \ldots + c_r m_r^u
\]

with arbitrary polynomial parameters \( c_i \in S \). Conversely, any controller \( u \) of the above form satisfies inclusion (4).

4. Example

We give an example that demonstrates the implementation of Algorithm 1 as follows.

[Example 2] Suppose we have the system of form (1)
Problem 2

In Step 4, we can verify that the following inclusion, which indicates the present problem is solvable. In Step 8, we obtain the polynomial vectors $z^* \in S^8$, $m_1^* \in S^8$, ..., and $m_6^* \in S^8$ as

$$z^* = [-Y_1, -Y_2 + Y_2, 1, 0, 0, 0, 0, 0, 0]^T,$$

$$m_1^* = [0, 0, 0, Y_1, -Y_2, -1, -1, 0, 0]^T,$$

$$m_2^* = [0, 0, 0, 0, 0, 0, 0, 0, 0]^T,$$

$$m_3^* = [0, 0, 0, 0, 0, 0, 0, 0, 0]^T,$$

$$m_4^* = [0, 0, 0, 0, 0, 0, 0, 0, 0]^T,$$

$$m_5^* = [0, 0, 0, 0, 0, 0, 0, 0, 0]^T.$$

Finally, in Step 8, we obtain by projection the polynomial vectors $u^* \in S^2$, $m_1^* \in S^2$, and $m_2^* \in S^2$ as

$$u^* = [-Y_1, -Y_2^2 + Y_2],$$

$$m_1^* = [0, Y_1^4 - Y_2^2 + Y_2^2, 0, 0, 0, 0, 0, 0, 0],$$

$$m_2^* = [0, Y_1^4 - Y_2^2 + Y_2^2, 0, 0, 0, 0, 0, 0, 0].$$

According to Remark 4, any static output-feedback controller $u$ satisfying eq. (3) is parametrized as

$$u(y) = \begin{bmatrix} -y_1 \\ -y_2 \\ + c_1(y) \\ + c_2(y) \end{bmatrix} \begin{bmatrix} y_1^2 - y_2^2 + y_2^2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

with arbitrary polynomial functions $c_i(\cdot)$ ($i = 1, 2$). Conversely, any controller $u$ of the above form satisfies eq. (3).

5. Conclusion

In this paper, we dealt with two problems of static output feedback for input-affine polynomial dynamical systems. One is to obtain a static output-feedback controller so as to render a prescribed algebraic set invariant for the resulting closed-loop system, and the other is to obtain a static output-feedback controller so as to realize a prescribed vector field on a prescribed algebraic set. We showed that the two problems can be represented by a particular inclusion of polynomials. A special class of the inclusion has been formulated and solved in [2] within the context of polynomial-type static state feedback for polynomial dynamical systems, and it was named the Lie derivative inclusion. We extended the Lie derivative inclusion to the case of static output feedback. Although there has been a certain difficulty in solving the inclusion with static output feedback, we derived an algorithm for solving the inclusion by removing the difficulty. By using the algorithm, all the static output-feedback controllers solving the inclusion can be exactly computed.

The Lie derivative inclusion can be formulated in more general rings, e.g., the rings of analytic functions, rational functions, and meromorphic functions. One direction for future study is to obtain an algorithm for solving the inclusion in these rings.
Acknowledgements

This work was partly supported by a Grant-in-Aid for Exploratory Research (No. 24656263) and a Grant-in-Aid for JSPS Fellows (No. 261539) from the Japan Society for the Promotion of Science.

References

[1] V. L. Syrmos, C. T. Abdallah, P. Dorato and K. Grigoriadis: Static output feedback—a survey; Automatica, Vol. 33, No. 2, pp. 125–137 (1997)
[2] T. Yuno and T. Ohtsuka: Lie derivative inclusion for a class of polynomial state feedback controls; Transactions of the Institute of Systems, Control and Information Engineers, Vol. 27, No. 11, pp. 423–433 (2014)
[3] W. Decker, G.-M. Greuel, G. Pfister and H. Schoenemann: Singular; http://www.singular.uni-kl.de/
[4] G.-M. Greuel and G. Pfister: A Singular Introduction to Commutative Algebra; 2nd ed., Springer-Verlag (2002)
[5] C. Ebenbauer and F. Allgöwer: Analysis and design of polynomial control systems using dissipation inequalities and sum of squares; Computers & Chemical Engineering, Vol. 30, No. 10, pp. 1590–1602 (2006)
[6] T. Ohtsuka: Model structure simplification of nonlinear systems via immersion; IEEE Transactions on Automatic Control, Vol. 50, No. 5, pp. 607–618 (2005)
[7] N. Bourbaki: Elements of Mathematics: Algebra I, Springer-Verlag (1989)
[8] M. F. Atiyah and I. G. Macdonald: Introduction to Commutative Algebra, Addison-Wesley (1969)
[9] D. A. Cox, J. B. Little and D. O'Shea: Using Algebraic Geometry, Springer-Verlag (1998)
[10] ———: Ideals, Varieties, and Algorithms, Springer-Verlag (1997)
[11] Z. Lin, L. Xu and N. K. Bose: A tutorial on Gröbner bases with applications in signals and systems; IEEE Transactions on Circuits and Systems I, Vol. 55, No. 1, pp. 445–461 (2008)
[12] J. Bochnak, M. Coste and M.-F. Roy: Real Algebraic Geometry, Springer-Verlag (1998)
[13] E. Becker and R. Neuhaus: Computation of real radicals of polynomial ideals; Computational Algebraic Geometry, F. Eyssette and A. Galligo, eds., pp. 1–20 (1993)
[14] A. Galligo and N. Vorobjov: Complexity of finding irreducible components of a semialgebraic set; Journal of Complexity, Vol. 11, No. 1, pp. 174–193 (1995)
[15] E. Becker and J. Schmider: On the real nullstellensatz; Algorithmic Algebra and Number Theory, B. H. Matzat, G.-M. Greuel and G. Hiss, eds., ch. B, Springer-Verlag, pp. 173–185 (1999)
[16] S. J. Spang: A zero-dimensional approach to compute real radicals; Computer Science Journal of Moldova, Vol. 16, pp. 64–92 (2008)
[17] E. Zerz and S. Walcher: Controlled invariant hypersurfaces of polynomial control systems; Qualitative Theory of Dynamical Systems, Vol. 11, No. 1, pp. 145–158 (2012)
[18] E. Zerz, S. Walcher and F. Güçlü: Controlled invariant varieties of polynomial control systems; Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems, pp. 1663–1667 (2010)
[19] N. Motese, B. Bamieh and M. Khammash: Model reduction of polynomial dynamical systems using differential algebras; 49th IEEE Conference on Decision and Control, pp. 6195–6200 (2010)
[20] M. Kreuzer and L. Robbiano: Computational Commutative Algebra I, Springer-Verlag (2000)
[21] G. Kemper: Calculating invariant rings of finite groups over arbitrary fields; Journal of Symbolic Computation, Vol. 21, pp. 351–366 (1996)
[22] A. Furukawa, T. Sasaki and H. Kobayashi: Gröbner basis of a module over K[x_1,...,x_n] and polynomial solutions of a system of linear equations; The 5th ACM Symposium on Symbolic and Algebraic Computation, pp. 222–224 (1986)

Appendices

A1. Computing the Set \{η_1,...,η_r\} in Algorithm 1

We use the notation defined so far. The following algorithm, which is a modification of Algorithms 3 and 4 in Appendix A.2, is used to derive Algorithm 1 in Section 3.

[Algorithm 2]

Given: the polynomial vector \( f \in \mathbb{R}^n \), the polynomial matrix \( G \in \mathbb{R}^{n \times m} \), and the polynomials \( \Phi_i \in \mathbb{R} \) and \( \Psi_i \in \mathbb{R} \) with \( i = 1,...,\ell \), and an ideal \( I = (\rho_1,\ldots,\rho_k) \subset \mathbb{R} \).

Obtain: a set \( \{\eta_1,\ldots,\eta_r\} \) of polynomial vectors \( \eta_i \in S^{m+1} \) (\( i = 1,\ldots,\nu \)).

Step 1. Put

\[
\Psi := \begin{bmatrix}
\Psi_1 - \frac{\partial \Phi_1}{\partial X} f \\
\vdots \\
\Psi_\ell - \frac{\partial \Phi_\ell}{\partial X} f
\end{bmatrix} \in \mathbb{R}^{\ell},
\]

\[
G := [g_1',\ldots,g_m'] := \begin{bmatrix}
\frac{\partial \Phi_1}{\partial X} G \\
\vdots \\
\frac{\partial \Phi_\ell}{\partial X} G
\end{bmatrix} \in \mathbb{R}^{\ell \times m}.
\]

Step 2. Let \( \{e_1,\ldots,e_\ell\} \) be the canonical basis of \( \mathbb{R}^\ell \), and compute a set of generators \( \{b'_1,\ldots,b'_\ell\} \) of the syzygy module

\[
\text{Syz}^R(\Psi,g_1',\ldots,g_m',\rho_1 e_1,\ldots,\rho_k e_\ell,\rho_1 e_1,\ldots,\rho_k e_\ell),
\]

using the algorithm given in [9].

Step 3. Put \( b_i := \text{Pr}^{m+1+k\ell} b_i' \in \mathbb{R}^{m+1} \) (\( i = 1,\ldots,\ell' \)).

Step 4. Let \( \{e_1,\ldots,e_{m+1}\} \) be the canonical basis of \( \mathbb{R}^{m+1} \), and define the \( R \)-submodule

\[
L = \langle b_1,\ldots,b_{\ell'} \rangle_R + \langle e_1,\ldots,e_{m+1} \rangle.
\]


\[ \ldots \rho_1 \epsilon_{m+1}, \ldots, \rho_k \epsilon_{m+1} \right] \}_{R'} \]

**Step 5** Set \( D = R[X_1, \ldots, X_n, Y_1, \ldots, Y_p] \).

**Step 6** Let \( \{ \epsilon_1, \ldots, \epsilon_{m+1} \} \) be the canonical basis of \( D^{m+1} \), and define the \( D \)-submodule \[
\mathcal{L}' = \langle b_1, \ldots, b_{l'}, p_1 \epsilon_1, \ldots, p_k \epsilon_{m+1} \rangle_{D'} + \langle (Y_1 - h_1) \epsilon_1, \ldots, (Y_p - h_p) \epsilon_1 \rangle_{D'} + \ldots + \langle (Y_1 - h_1) \epsilon_{m+1}, \ldots, (Y_p - h_p) \epsilon_{m+1} \rangle_{D'}.
\]

**Step 7** Compute a Gröbner basis \( L' \) w.r.t. an elimination ordering for \( \{ X_1, \ldots, X_n \} \).

**Lemma 1** Let \( \{ \eta_1, \ldots, \eta_k \} \) be the set computed by Algorithm 2. Then,

\[
\Pi(\{ \eta_1, \ldots, \eta_k \}) = \text{Syz} \overline{\Pi} \langle \eta_1, \ldots, \eta_k \rangle \cap \overline{\Pi}^{m+1} \quad (A1)
\]

**(Proof)** Note that \( \Pi = \text{Fct}^I \circ \text{Sbs}^{m+1} \). **Algorithm 2** can be divided into two parts: the first half (Steps 1–3) corresponds to **Algorithm 3** and the latter half (Steps 3–7) corresponds to **Algorithm 4**. Consider the set \( \{ b_1, \ldots, b_{l'} \} \) in **Algorithm 2**. Since the construction of \( \{ b_1, \ldots, b_{l'} \} \) in **Algorithm 2** corresponds to **Algorithm 3** with \( (v_1, \ldots, v_{m+1}) := (\psi', \gamma_1, \ldots, \gamma_m) \), we have

\[
\langle b_1, \ldots, b_{l'} \rangle_{\overline{\Pi}} = \text{Syz} \langle \psi', \gamma_1, \ldots, \gamma_m \rangle = \text{Syz} \langle \psi', \gamma_1, \ldots, \gamma_m \rangle.
\]

On the other hand, since the construction of \( \{ \eta_1, \ldots, \eta_k \} \) in **Algorithm 2** corresponds to **Algorithm 4** with \( L := (b_1, \ldots, b_{l'}) + I^{m+1} \), we have

\[
\begin{align*}
\text{Sbs}^{m+1}_{\psi'}(\{ \eta_1, \ldots, \eta_k \}) &= \left( \text{Sbs}^{m+1}_{\psi'}(\{ \eta_1, \ldots, \eta_k \}) \right)_{H} = L \cap H^{m+1} = \left( (b_1, \ldots, b_{l'}) + I^{m+1} \right) \cap H^{m+1}. 
\end{align*}
\]

Now, we prove that

\[
\text{Fct}^I \left( L \cap H^{m+1} \right) = \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \cap \overline{\Pi}^{m+1} \quad (A4)
\]

as follows. Take an arbitrary \( \xi \in R^{m+1} \) such that \( \xi \in L \cap H^{m+1} \). Since \( \xi \in H^{m+1} \), it is obvious that \( \xi = \text{Fct}^I (\xi) = \xi \in \overline{\Pi}^{m+1} \). Moreover, since \( \xi \in L = (b_1, \ldots, b_{l'}) + I^{m+1} \), there exist \( a_i \in R \) \( (i = 1, \ldots, l') \) and \( \theta \in I^{m+1} \) such that

\[
\xi = a_1 b_1 + \ldots + a_{l'} b_{l'} + \theta.
\]

Hence, we have

\[
\text{Fct}^I (\xi) = \xi = \overline{\pi}_1 b_1 + \ldots + \overline{\pi}_{l'} b_{l'} \in \langle b_1, \ldots, b_{l'} \rangle_{\overline{\Pi}}.
\]

which implies \( \text{Fct}^I (\xi) \in \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \). Thus, we have the inclusion

\[
\text{Fct}^I \left( L \cap H^{m+1} \right) \subset \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \cap \overline{\Pi}^{m+1}.
\]

Conversely, take an arbitrary \( \xi \in \overline{\Pi}^{m+1} \) such that

\[
\xi \in \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \cap \overline{\Pi}^{m+1}.
\]

Since eq.(A2) holds, there exist \( a_i \in R \) \( (i = 1, \ldots, l') \) such that

\[
\xi = \overline{\pi}_1 b_1 + \ldots + \overline{\pi}_{l'} b_{l'}.
\]

Moreover, since \( \xi \in \overline{\Pi}^{m+1} \), there exists \( \theta \in I^{m+1} \) such that

\[
a_1 b_1 + \ldots + a_{l'} b_{l'} + \theta \in H^{m+1}.
\]

Setting \( \zeta := a_1 b_1 + \ldots + a_{l'} b_{l'} + \theta \), we have

\[
\zeta \in \left( (b_1, \ldots, b_{l'}) + I^{m+1} \right) \cap H^{m+1} = L \cap H^{m+1}
\]

and Fct\((\zeta) = \zeta\), which imply the inclusion

\[
\text{Fct}^I \left( L \cap H^{m+1} \right) \supset \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \cap \overline{\Pi}^{m+1}.
\]

Therefore, we have that eq.(A4) holds. Thus, from eq.(A3) and eq.(A4), we have

\[
\text{Fct}^I \circ \text{Sbs}^{m+1}_{\psi'}(\{ \eta_1, \ldots, \eta_k \}) = \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \cap \overline{\Pi}^{m+1},
\]

which implies eq.(A1).

**A2. The Algorithms Employed for Solving the Lie Derivative Inclusion**

In the following, the indexes of the symbols are associated with those in the foregoing sections.

**A2.1 Computing a Syzygy Module over a Factor Ring**

Consider the following problem.

**Problem 3** Given an \((m+1)\)-tuple \((v_1, \ldots, v_{m+1})\) of polynomial vectors \(v_i \in R^l \) \( (i = 1, \ldots, m+1) \), compute a set of generators \( \{ \overline{b}_1, \ldots, \overline{b}_{l'} \} \) of the syzygy module \( \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \).

The algorithm for solving the above problem is given below.

**Algorithm 3** (Computing a syzygy module over a factor ring) [4, Remark 2.5.6]

**Given:** an \((m+1)\)-tuple \((v_1, \ldots, v_{m+1})\) of polynomial vectors \(v_i \in R^l \) \( (i = 1, \ldots, m+1) \).

**Obtain:** a set of generators \( \{ \overline{b}_1, \ldots, \overline{b}_{l'} \} \) of the syzygy module \( \text{Syz} \overline{\Pi} (\psi', \gamma_1, \ldots, \gamma_m) \).

**Step 1** Let \( \{ e_1, \ldots, e_l \} \) be the canonical basis of \( R^l \), and compute a set of generators \( \{ \overline{b}_1, \ldots, \overline{b}_{l'} \} \) of the syzygy module

\[
\text{Syz}^R (v_1, \ldots, v_{m+1}, \rho_1 e_1, \ldots, \rho_k e_k, \\
\rho_1 e_2, \ldots, \rho_k e_2, \\
\ldots, \\
\rho_1 e_l, \ldots, \rho_k e_l),
\]

using the algorithm given in [9].

**Step 2** Put \( b_i := \text{Fct} \overline{\Pi}^{m+1} (\psi', \gamma_1, \ldots, \gamma_m) (\overline{b}_i) \) \( (i = 1, \ldots, l') \).

**Step 3** Put \( \overline{b}_i := \text{Fct}^I (b_i) \in \overline{\Pi}^{m+1} \) \( (i = 1, \ldots, l') \).
A2.2 Intersecting a Module with a Subalgebra

Consider the following problem.

**Problem 4** Given an $R$-submodule $L = \langle w_1, \ldots, \ w_{r+k(m+1)} \rangle_R \subset R^{m+1}$ and a subring $H = R[\ h_1, \ldots, \ h_p \ ] \subset R$, obtain a set of generators $\{ \ eta_1, \ldots, \ eta'_p \}$ of the $H$-module $L \cap H^{m+1}$.

The algorithm for solving the above problem is given below.

**Algorithm 4** (Intersecting a module with a subalgebra) [21]

Given: a set of generators $\{ w_1, \ldots, \ w_{r+k(m+1)} \}$ of an $R$-submodule $L \subset R^{m+1}$, and a set of generators $\{ h_1, \ldots, \ h_p \}$ of a subring $H = R[\ h_1, \ldots, \ h_p \ ] \subset R$.

Obtain: a set of generators $\{ \eta_1, \ldots, \ \eta'_p \}$ of the $H$-module $L \cap H^{m+1}$.

Step 1 Set $D := R[X_1, \ldots, X_n, Y_1, \ldots, Y_p]$.

Step 2 Let $\{ e_1, \ldots, e_m \}$ be the canonical basis of $R^{m+1}$, and form the $D$-submodule

$L' = \langle w_1, \ldots, \ w_{r+k(m+1)} \rangle_D + \langle (Y_1 - h_1) e_1, \ldots, (Y_p - h_p) e_1 \rangle_D + \langle \ldots \rangle_D$.

Step 3 Compute a Gröbner basis $\mathcal{L}$ of $L'$ w.r.t. an elimination ordering for $\{ X_1, \ldots, X_n \}$.

Step 4 Put $\{ \eta_1, \ldots, \eta_m \} := \mathcal{L} \cap S^{m+1}$.

Step 5 Put $\eta'_i := \text{Sls}\mathcal{L}^{m+1}(\eta_i)$ $(i = 1, \ldots, m)$.

A2.3 Computing the Kernel of an Algebra Homomorphism

Consider the following problem.

**Problem 5** Given an $R$-algebra homomorphism $\pi: S \rightarrow \overline{R}$ defined by $Y_i \mapsto \overline{h}_i$, compute a set of generators $\{ k_1, \ldots, k_{\nu} \}$ of $\ker \pi$.

The algorithm for solving the above problem is given below.

**Algorithm 5** (Computing the kernel of an algebra homomorphism) [4]

Given: an $R$-algebra homomorphism $\pi: S \rightarrow \overline{R}$ defined by $Y_i \mapsto \overline{h}_i$.

Obtain: a Gröbner basis $\{ k_1, \ldots, k_{\nu} \}$ of $\ker \pi$.

Step 1 Set $D := R[X_1, \ldots, X_n, Y_1, \ldots, Y_p]$.

Step 2 Put $\beta_i := Y_i - h_i \in D$ $(i = 1, \ldots, p)$.

Step 3 Compute a Gröbner basis $B$ of the ideal

$\langle \rho_1, \ldots, \rho_k, \beta_1, \ldots, \beta_p \rangle_D \subset D$

w.r.t. an elimination ordering for $\{ X_1, \ldots, X_n \}$.

Step 4 Put $\{ k_1, \ldots, k_{\nu} \} := B \cap S$.

A2.4 Solving a Linear Equation with Polynomial Coefficients

Consider the following problem.

**Problem 6** Given $\gamma_0 \in S^{m+1}$ and $\gamma_i \in S^{m+1}$ $(i = 1, \ldots, m+d)$, obtain a solution $z \in [z_1, \ldots, z_{m+d}]^T \in S^{m+d}$ of the equation

$\gamma_0 = z_1 \gamma_1 + \cdots + z_{m+d} \gamma_{m+d}$.

(A5)

The algorithm for solving the above problem is given below.

**Algorithm 6** (Computing the set of all solutions of a linear equation with polynomial coefficients) [9, 11, 22]

Given: polynomial vectors $\gamma_i \in S^{m+1}$ $(i = 0, \ldots, m + d)$.

Obtain: a polynomial vector $z^* \in S^{m+d}$ and a set $\{ m_1, \ldots, m_\nu \}$ of polynomial vectors $m_i \in S^{m+d}$ $(i = 1, \ldots, r)$ such that $z^* + \{ m_1, \ldots, m_\nu \} \in S^{m+d}$ is the set of all solutions of eq.(A5).

Step 1 Compute a Gröbner basis $\{ \gamma_1, \ldots, \gamma_\nu \}$ of $S^{m+d} \subset S^{m+d}$ by Buchberger's algorithm [9, 20].

Step 2 Compute a transformation matrix $Q \in S^{(m+d) \times d}$ such that $[\gamma_1, \ldots, \gamma_\nu] = [\gamma_1, \ldots, \gamma_\nu]Q$. This matrix can be obtained by tracing Buchberger's algorithm in Step 1.

Step 3 Divide $\gamma_0$ by $\{ \gamma_1, \ldots, \gamma_\nu \}$, and let $z' \in S^q$ and $z'' \in S^{m+d}$ be the quotient and the remainder of $\gamma_0$, respectively.

Step 4 If $z'' \neq 0$, then terminate the algorithm because eq.(A5) does not have a solution; otherwise, proceed to the next step because eq.(A5) has a solution.

Step 5 Compute a set of generators $\{ m_1, \ldots, m_\nu \}$ of $\text{Syz}(\gamma_1, \ldots, \gamma_\nu)$ by the algorithm given in [9].

Step 6 Put $z^* := Qz'$ and $\{ m_1, \ldots, m_\nu \} := \{ m_1, \ldots, m_\nu \}$.

Authors

Tsunoshi Yuno

He received his B.E. degree from Kumamoto University, Japan, in 2010. He received his M.E. degree from Osaka University, Japan, in 2012. He is currently a Ph.D. student in Engineering Science at Osaka University. He is also a JSPS research fellow (DC2). His research interests include nonlinear control theory.

Toshiyuki Ohtsuka (Member)

He received his B.E., M.E., and Ph.D. degrees from the Tokyo Metropolitan Institute of Technology, Japan, in 1990, 1992, and 1995, respectively. From 1995 to 1999, he was an Assistant Professor at the University of Tsukuba. In 1999, he joined Osaka University, and he was a Professor in the Graduate School of Engineering Science from 2007 to 2013. In 2013, he joined Kyoto University as a Professor in the Department of Systems Science at the Graduate School of Informatics. His research interests include nonlinear control theory and real-time optimization with applications to mechanical systems and environmental systems. He is a member of IEEE, SICE, JSME, JSASS, and AIAA.