Divide and Conquer for Stability in Control Systems

Paul Lutkus

Introduction: *Modern Challenges in Control*

In science, engineering, and society, the control of nonlinear systems of various types and degrees is paramount. From automated manufacturing, to the dynamics of power grids and communication networks, and even to biochemical processes, the systems we encounter in real-world application areas are largely nonlinear and often complex, but they share the key property of being actuated or disturbed through various channels that can often be quantified analytically and then approximated practically.

A key open problem in the field of nonlinear dynamics is the control of systems with high-dimensional state or output. A humanoid robot could have upwards of twenty degrees of freedom (e.g. joints, motors, etc.) and therefore be described by forty-dimensional dynamics when accounting for the positions and velocities of each component. Even more problematic, the output of a grayscale megapixel camera is an output function with one million dimensions (one for each pixel). Applying rigorous, constructive methodologies from classical control, such as designing Lyapunov and Barrier functions for stability and safety [1], solving Hamilton-Jacobi PDEs to obtain reach-avoid value functions (e.g. for pursuit and evasion) [2], and solving nonlinear model predictive control problems [4] to obtain optimal system trajectories (e.g. gait-generation for walking robots), are all challenging or even nonstarters for the real-time, practical control for these high-dimensional systems, simply due to the well-known "curse of dimensionality", i.e. exponential computational explosion with respect to dimension.

The main thrust of research to solve these problems, over the last half of a decade, has primarily been in machine learning and robotics, leveraging deep neural networks to uncover patterns or "features", "representations", and "embeddings" that solve control problems when passed through a decision-making module, usually another neural network [9]. For complex robotic systems, it can be argued that this approach dominates classical control strategies in performance, especially in regard to control from vision, in controlled environments where large datasets of demonstrations can be collected. However, deep neural network control strategies are not without their limitations. Evidence suggests that these approaches can fail to generalize to similar but non-identical tasks to those found in the training data [10], and therefore do not provide the robustness critical for high-stakes applications. Furthermore, there is also little hope of computationally certifying the behavior of a these neural network control systems (due to the same curse of dimensionality).

The angle presented here is as follows: *rather than using neural networks to directly make control decisions for complex systems, we might instead use neural networks to decompose complex systems into simple components, and then use classical control strategies to make decisions.* This is a **divide and conquer** approach to control design, and it can made precise using the differential-geometric concept of diffeomorphisms between manifolds. In their simplest case, diffeomorphisms are differentiable, bijective functions between two copies of a Euclidean space. In the following, we propose partitioned diffeomorphims as principled and learnable reductions that take high-dimensional state and input to simplified representations from which classical control methodologies can be used to solve the original, high-dimensional control problem. These simplified representations generate low-dimensional, surrogate control problems whose solutions can be rigorously recombined and lifted back to the original system.

Mathematical Background

Definition 1 (Differentiable and Smooth Manifolds) *Consider a topological manifold: informally, a topological space M that is locally homeomorphic to Euclidean space. For every $p \in M$, there exists an open set $U \subseteq M$ (a neighborhood) for which there exists a homeomorphism (a continuous function with continuous inverse) $\varphi : U \to \varphi(U) \subseteq \mathbf{R}^n$. This φ is called a chart. A collection of charts whose neighborhoods cover M is called an atlas, which we denote $\mathcal{A}$. Consider an index set $I \subseteq \mathbf{R}$. We require $\mathcal{A} = \{(U_i, \varphi_i)\}_{i \in I}$ to satisfy the following properties for M to be differentiable:*

1. $\bigcup_{i \in I} U_i = M$
2. *For any charts-neighborhood tuples $(U, \varphi), (V, \psi) \in \mathcal{A}$ such that $U \cap V \neq \emptyset$, $\varphi \circ \psi^{-1} : \psi(U \cap V) \to \varphi(U \cap V)$ is a homeomorphism between open subsets of $\mathbf{R}^n$ (this follows from the assumption that φ and ψ are homeomorphisms). We call $\varphi \circ \psi^{-1}$ a transition function. The manifold M is called differentiable if all transition functions are also continuously differentiable and have continuously differentiable inverses (i.e., are diffeomorphims).*

If the transition functions are infinitely-differentiable, than M is called a smooth manifold.

Definition 2 (Diffeomorphism) *Let M and N be differentiable manifolds. A continuously differentiable function $\Phi : M \to N$ is called a diffeomorphism if it is a bijection and Φ^{-1} is continuously differentiable. The differential of a diffeomorphism from M to N is a map from the tangent bundle TM to the tangent bundle TN called the differential, denoted $d\Phi_p : T_pM \to T_{\Phi(p)}N$. The differential $d\Phi_p$ is expected to be a linear isomorphism between tangent spaces. When N and M are Euclidean spaces, $d\Phi_p$ is just the Jacobian of Φ evaluated at p, i.e., $\frac{\partial \Phi}{\partial p}(p)$.*

Definition 3 (Autonomous ODEs on Manifolds, and Solutions) *Let M be smooth manifold and TM be its tangent bundle. An autonomous ODE on M is a smooth vector field X, i.e., a smooth map $X : M \to TM$ such that $X(p) \in T_pM$ for all $p \in M$. The solution, also called an integral curve, to an ODE on a M is a smooth map $\gamma : I \subseteq R \to M$ that satisfies the following differential equation, $\dot{\gamma}(t) = X(\gamma(t)) \quad \forall t \in I$. Two ODEs (M, X) and (N, Y) are diffeomorphically related if there is a diffeomorphism $\Phi : M \to N$ that bijectively transforms the trajectories of X on M to the trajectories of Y on N. This can equivalently be expressed in terms of the differential via the relation $d\Phi_p(X(p)) = Y(\Phi(p)) \quad \forall p \in M$. We will see this relation take a simpler form when M and N are restricted to be Euclidean spaces.*

Assumption 1 (Euclidean Domains) *In the remainder of this document, we will assume all differential equations are defined on Euclidean spaces, i.e., $\mathbf{R}^n$. As a consequence, the differential of a diffeomorphism Φ between domains will reduce to the Jacobian of Φ. Furthermore, $\Phi : \mathbf{R}^n \to \mathbf{R}^n$ will be a global diffeomorphism as long as Φ is a proper map ($M \subset \mathbf{R}^n$ being compact implies $\Phi^{-1}(M)$ is compact) and $\operatorname{rank}\left(\frac{\partial \Phi}{\partial x}(x)\right) = n$ for all $x \in \mathbf{R}^n$. The purpose of this assumption is to allow us to work with standard techniques from multivariable calculus.*

The purpose of introducing the topic from the abstract perspective is to illustrate that the central ideas discussed here, diffeomorphisms between the domains of differential equations, can be generalized to a broader class of systems – differential equations whose domains are nontrivial manifolds – than those directly discussed in this document. Furthermore it identifies specifically those techniques which break down in the more complex setting (where multivariable calculus must be generalized to calculus on manifolds).

Definition 4 (Euclidean ODEs with Control Parameter) *Now, define a vector field $f : \mathbf{R}^n \to \mathbf{R}^n$, which is also referred to as the RHS of the ODE $\dot{x}(t) = f(x(t))$. Observe that this is a special case of the definition of ODEs on manifolds (cf. Definition 3). Next consider a diffeomorphism $\Phi : \mathbf{R}^n \to \mathbf{R}^n$. We can transform the trajectories by the diffeomorphism to obtain $\frac{\partial \Phi}{\partial x}(x) f(x) = g(\Phi(x))$, for some vector field $g : \mathbf{R}^n \to \mathbf{R}^n$. Also observe that this relationship is a special case of diffeomorphically-related ODEs on manifolds (cf. Definition 3). Next, we introduce a second parameter $u \in \mathbf{R}^p$ to the vector fields $f(x, u)$ and $g(x, u)$. We will assume that we are able to freely choose $u \in \mathbf{R}^p$, for each $t \in \mathbf{R}_{\geq 0}$, to attempt to elicit certain behavior from the vector fields $\dot{x} = f(x, u)$ and $\dot{y} = g(y, u)$, where $y = \Phi(x)$. The map $u(t) : \mathbf{R}_{\geq 0} \to \mathbf{R}^p$ will be referred to as the control signal.*

Definition 5 (Stability) *In the remainder of this document, the behavior that we will attempt to elicit from the solutions $x(t)$ of the ODE $\dot{x}(t) = f(x, u)$ will be stability. Informally, stability refers to the simple property that $\lim_{t\to\infty} x(t) = 0$, where 0 is referred to as the equilibrium point, but we will provide further refinements to this informal definition in what follows. The following stronger definition can be provided using epsilon-delta reasoning, and patches a significant hole in the simple limit conception.*

> *(ε, δ)-**Stability:** Let 0 be an equilibrium point of the system f, i.e. $f(0, 0) = 0$. A system is (ε, δ)-stable if, for every $\varepsilon > 0$, there exists a $\delta > 0$ such that $\|x(0)\| < \delta \implies \|x(t)\| < \epsilon$ for all $t \geq 0$.*

Observe that $\lim_{t\to\infty} x(t) = 0$ allows the trajectory $x(t)$ to travel arbitrarily far from the equilibrium point and stay far for an arbitrarily long time. The ε, δ definition of stability allows us to pick an arbitrary small ϵ-radius and provides an initial radius for which, if the trajectory begins within this initial radius, it will remain within the ε-radius for all time. Observe however, that this (ε, δ)-stability does not guarantee that the trajectory $\lim_{t\to\infty} x(t) = 0$. The combination of both notions yields the next definition of stability:

> ***Asymptotic-Stability:** Let 0 be the equilibrium point of the system f, i.e. $f(0, 0) = 0$. A system is asymptotically stable if it is (ε, δ)-stable and there exists some $\delta > 0$ such that $\|x(0)\| < \delta \implies \lim_{t\to\infty} x(t) = 0$ If $\lim_{t\to\infty} x(t) = 0$ for all $x(0)$, then we refer to the system as globally asymptotically stable.*

Next, we present the concept of Lyapunov stability and Lyapunov functions, which allow to certify the stability of a system f by construction a positive-definite scalar function, with a unique minimizer, which is decreasing on the system's trajectories $x(t)$.

Definition 6 (Lyapunov Function) *A Lyapunov $V : D \subset \mathbf{R}^n \to \mathbf{R}$ is a scalar function that satisfies the following properties*

1. *$\dot{V}(x) = \nabla V(x) \cdot f(x) \leq -\alpha(V(x)) \; \forall x \in D$, for some $\alpha : \mathbf{R}_{\geq 0} \to \mathbf{R}$ such that $\alpha(0) = 0$ and α is strictly increasing.*
2. *$V(0) = 0$ and $V(x) > 0$ for all $x \in D \setminus \{0\}$.*

Property (1) corresponds to the trajectories of the system decreasing in V with a rate proportional to $V(x)$, and property (2) says that V must be positive definite with a unique global mininum. If such a function V exists, then the system f is asymptotically stable on the domain $D \subset \mathbf{R}^n$. If V is defined and satisfies properties (1) and (2) on all $\mathbf{R}^n$, along with being radially unbounded, i.e., $V(x) \to \infty$ as $\|x\| \to \infty$, then the system f is called globally asymptotically stable and trajectories converge to the equilibrium point 0 for all $x(0) \in \mathbf{R}^n$.

Example 1 (Quadratic Lyapunov Functions & Linear Systems) *Let $\dot{x} = f(x) = Ax$ and $P \in \mathbf{R}^{n\times n}$ be a positive definite matrix (i.e., all its eigenvalues are positive). Then $V(x) = x^T Px$ is a Lyapunov function for f if there exists a positive definite Q such that $A^TP + PA = -Q$. Observe that $\dot{V}(x) = \frac{d}{dt}[x^T]Px + x^TP\frac{d}{dt}[x] = x^T(A^TP + PA)x$, and so $\dot{V}(x) < 0$ if the matrix $A^TP + PA$ is negative definite.*

The previous definitions of stability contain no mention of a control input u, which means we must design a control law $\pi : \mathbf{R}^n \to \mathbf{R}^p$ which specifies a unique control input for each state x, rendering the system $f(x, u)$ autonomous, i.e., $\dot{x} = f(x, \pi(x)) = \bar{f}(x)$. In the following section, we will discuss our primary tool for designing $\pi(x)$, called *partial feedback linearization*, which will setting the stage for our main result.

Control-Design: *Partial Feedback Linearization*

As previously discussed, we seek to reduce a control system $\dot{x} = f(x, u)$ to an autonomous system of the form $\dot{x} = \bar{f}(x) = f(x, \pi(x))$ by the creation of a state-feedback control law $\pi(x) = u$. A well-studied technique in this regard is that of partial feedback linearization [5], in which an output function $y = h(x)$ is designed for which the dynamics $\dot{y}$ are made to be linear via an associated control law $\pi(x, z)$.

Definition 7 (Relative Degree) *Consider a smooth, scalar output function $y = h(x)$, for the system $\dot{x} = f(x) + g(x)u$ that has a single scalar input $u \in \mathbf{R}$. Because the system described by (f, g, h) has a single input and single output, we call it a SISO system. Since the input acts on the system as a linear function of the vector-valued function $g : \mathbf{R}^n \to \mathbf{R}^n$, we call the resulting system control-affine. Define the Lie derivative $L_fh(x)$ to be $\nabla h(x) \cdot f(x) \in \mathbf{R}$, which corresponds to the change in h along the vector field $f(x)$. Similarly, $L_gh(x) = \nabla h(x) \cdot g(x) \in \mathbf{R}$. Observe that $\dot{y} = L_fh(x) + L_gh(x)u$. If $L_gh(x) = 0$, then the control input does not appear in $\dot{y}$. The relative degree of a system $f(x) + g(x)u$ for the output $y = h(x)$ is the number of time derivatives of y that must be taken before the control input appears. A SISO system is said to have relative-degree r if the following properties hold for all x.*

$$(1): \ L_gL_f^kh(x) = 0 \ \forall k < r-1, \quad (2): \ L_gL_f^{r-1}h(x) \neq 0$$

If (1) and (2) hold, then $y^{(r)} = L_f^rh(x) + L_gL_f^{r-1}h(x)u$, which can be proven by induction.

Definition 8 (Partial Feedback Linearization) *Let $\dot{x} = f(x) + g(x)u$ be a relative degree r, SISO system with respect to the output function $y = h(x)$. We define $z = (y, \dot{y}, \dots y^{(r-1)}) \in \mathbf{R}^r$ and obtain a set of $n - r$ unactuated (i.e., $L_g\eta_i(x) = 0 \ \forall i \in [1, n-r]$) coordinates $(\eta_1, \dots, \eta_{n-r})$ in order to complete the diffeomorphism $\Phi(x) = (\Phi_z(x), \Phi_\eta(x)) = (z, \eta)$. The decomposition of the state x into actuated z coordinats and unactuated η coordinates is known as the Isidori normal form. Let $\Phi_r(x)$ be the r'th coordinate of the diffeomorphism Φ. Using our previous notation for diffeomorphically-related ODEs, denote the dynamics of the actuated coordinates z to be $\frac{\partial\Phi_r}{\partial x}(x)[f(x) + g(x)u] = b(\Phi_z(x), \Phi_\eta(x)) + a(\Phi_z(x), \Phi_\eta(x))u$ for the r'th coordinate, and $\frac{\partial\Phi_i}{\partial x}(x)[f(x) + g(x)u] = \Phi_{i+1}(x)$ all the remaining actuated coordinates indexed by $i \in [1, r-1]$. The relative degree assumption on (f, g, h) ensures that the remaining coordinates that are not directly actuated by u act as a chain of integrators for which $\dot{z}_i = z_{i+1}$. This is akin to the pedal on a car controlling the acceleration and thus the velocity and position through two degrees of integration. The control law which acts on $\Phi_r(x)$ is of the form*

$$\pi(x) = \frac{1}{a(\Phi_z(x), \Phi_\eta(x))}(-b(\Phi_z(x), \Phi_\eta(x)) + v) \tag{1}$$

The effect of $\pi(x)$ is to cancel the nonlinearities in $\frac{\partial\Phi_r}{\partial x}(x)[f(x) + g(x)u]$ so that $\dot{z}_r = \frac{\partial\Phi_r}{\partial x}(x)[f(x) + g(x)u] = v$. Therefore, the z-dynamics become the linear system

$$\dot{z} = \begin{bmatrix} 0 & I_{r-1\times r-1} \\ \mathbf{0}_{1\times r-1} & 0 \end{bmatrix} z + \begin{bmatrix} \mathbf{0}_{r-1\times 1} \\ 1 \end{bmatrix} v \ .$$

The remaining dynamics are those of the η coordinates which have the form $\frac{\partial\Phi_\eta}{\partial x}(x)[f(x) + g(x)u] = q(\Phi_\eta(x), \Phi_z(x))$. Because these coordinates are unactuated, their autonomous stability is necessary for stability of the original system $f(x) + g(x)u$. We call the stabiliy of the η-system the minimum phase property, and the system is globally minimum phase if the η dynamics are stable for all $\eta \in \mathbf{R}^{n-r}$ when $z = 0$.

If the system is globally minimum phase, the full state $x(t)$ will always be stable as long as the system $\dot{\eta} = q(\eta, z)$ is forward complete, meaning that its trajectories do not go to infinity in finite time. Because the linear control design (which can guarantee an exponential convergence rate) for the z-subsystem ensures that the $z(t)$ gets arbitrarily close to zero in finite time, the minimum phase property and continuity of $q(\eta, z)$ ensures that it will become stable after finite time, and therefore that the full state $x(t)$, which is equivalent to $(z(t), \eta(t))$ under diffeomorphism (which preserves stability properties), will also become stable. Whether or not a system is minimum phase is purely a function of the tuple (f, g, h), and does not depend on the chosen feedback control law for the z-system.

In the following section, we will take inspiration from feedback linearization to design state and control diffeomorphims that decompose an ODE $f(x, u)$ into a tree of smaller ODEs for which simpler stabilization problems can be solved. Lyapunov function certificates for the subsystems can then be combined back up the tree in order to certify stability of the original system.

Main Result: *Tree Decomposition of ODEs*

Consider a control-affine ODE $\dot{x} = f(x) + g(x)u$. In the previous section, we considered scalar output functions $y = h(x)$ with relative degree r, and stacked their time derivatives in order to build a diffeomorphism. Inspired by partial feedback linearization, we will consider vector-valued, potentially relative degree one outputs $\Phi(x) = (E(x), R(x)) = (z, \xi)$ where $E(x) \in \mathbf{R}^m$ and $R(x) \in \mathbf{R}^{n-m}$. Furthermore, we also consider control diffeomorphisms $\Psi(u) = (G(u), S(u)) = (v, w)$ where $G(u) \in \mathbf{R}^l$ and $S(u) \in \mathbf{R}^{p-l}$. Using the Lie derivative notation from feedback linearization, we can write out

$$\dot{z} = \frac{\partial E}{\partial x}(x)\left[f(x) + g(x)\Psi^{-1}(v, w)\right] = L_f E(x) + L_g E(x)\Psi^{-1}(v, w)\ ,$$
$$\dot{\xi} = \frac{\partial R}{\partial x}(x)\left[f(x) + g(x)\Psi^{-1}(v, w)\right] = L_f R(x) + L_g R(x)\Psi^{-1}(v, w)\ .$$

Without loss of generality, we can consider $\dot{x} = f(x) + g(x)u$ to be an arbitrary parent node, and $\dot{z}$ and $\dot{\xi}$ its two child nodes.

Case 1: ***Perfect Decoupling***. The first case to analyze is when $\dot{z}$ and $\dot{\xi}$ are perfectly decoupled, i.e., $L_f E(x) = f_z(z)$, $L_g E(x)\Psi^{-1}(v, w) = g_z(z)v$, $L_f R(x) = f_\xi(\xi)$, and $L_g R(x)\Psi^{-1}(v, w) = g_\xi(\xi)w$. Because $\dot{z} = f_z(z) + g_z(z)v$ and $\dot{\xi} = f_\xi(\xi) + g_\xi(\xi)w$, the z-system is relative degree one with respect to v, and the ξ system is relative degree one with respect to w. The perfect decoupling case corresponds to a unique block-diagonal structure in the input-output matrix. Define the input matrix $A(x)$ and control transformation Jacobian matrix $J(v, w)$ as follows:

$$A(x) = \begin{bmatrix} L_g E(x) \\ L_g R(x) \end{bmatrix}, \quad J(v, w) = \begin{bmatrix} \frac{\partial \Psi^{-1}}{\partial v}(v, w) & \frac{\partial \Psi^{-1}}{\partial w}(v, w) \end{bmatrix}$$

Observe that $L_g E(x)\Psi^{-1}(v, w) = g_z(z)v$ and $L_g R(x)\Psi^{-1}(v, w) = g_\xi(\xi)w$ imply that $L_g E(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) = 0$ and $L_g R(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) = 0$ respectively. Thus the product $A(x)J(v, w)$ must have block-diagonal structure:

$$A(x)J(v, w) = \begin{bmatrix} L_g E(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) & L_g E(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) \\ L_g R(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) & L_g R(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) \end{bmatrix} = \begin{bmatrix} L_g E(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) & \mathbf{0}_{m \times p-l} \\ \mathbf{0}_{n-m \times l} & L_g R(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) \end{bmatrix}\ . \tag{2}$$

In Case 1, both child nodes can be further decomposed.

Case 2: ***Control Decoupling with State Cascade***. The second case is when $\dot{z} = f_z(z) + g_z(z)v$ and $\dot{\xi} = f_\xi(\xi, z) + g_\xi(\xi, z)w$. We call $\dot{z}$ the driving system, and $\dot{\xi}$ the driven system, because the state $z(t)$ appears as an input to the system f_ξ and g_ξ. The requisite conditions are $L_f E(x) = f_z(z)$, $L_g E(x)\Psi^{-1}(v, w) = g_z(z)v$, and $L_g R(x)\Psi^{-1}(v, w) = g_\xi(\xi, z)w$. Because $L_g E(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) = 0$ and $L_g R(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) = 0$ must also hold as in Case 1, $A(x)J(v, w)$ must also have block-diagonal structure as in eq. (2). Because $\dot{\xi}$ is now driven by an external variable $z(t)$ that is being controlled in a parallel branch, we can no longer decompose $\dot{\xi}$ and must do control design with respect to this exogenous input $z(t)$, to be discussed later. Thus,

In Case 2, one child node, the driven system, must be a leaf node.

Case 3: ***Actuation Decomposition with State Cascade***. The third case dynamics of the form $\dot{z} = f_z(z) + g_z(z)v$ and $\dot{\xi} = f_\xi(\xi, z)$. Like in Case 2, we require that $L_f E(x) = f_z(z)$. This case is most similar to partial feedback linearization (cf. Definition 8), where the η-system is unactuated, as is the case for the ξ-system here. The unactuated condition on $\dot{\xi}$ requires that $L_g R(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) = 0$ and $L_g R(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) = 0$, so that we have the following form for $A(x)J(v, w)$:

$$A(x)J(v, w) = \begin{bmatrix} L_g E(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) & L_g E(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) \\ L_g R(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) & L_g R(x)\frac{\partial \Psi^{-1}}{\partial w}(v, w) \end{bmatrix} = \begin{bmatrix} L_g E(x)\frac{\partial \Psi^{-1}}{\partial v}(v, w) & \mathbf{0}_{m \times p-l} \\ \mathbf{0}_{n-m \times l} & \mathbf{0}_{n-m \times p-l} \end{bmatrix}\ . \tag{3}$$

Because the driven system has no control input to design, there is no point decomposing it further, thus,

In Case 3, the driven system must be a leaf node

Case 4: ***Control Coupling with State Cascade***. The fourth case is when $\dot{z} = f_z(z) + g_z(z)v$ and $\dot{\xi} = f_\xi(\xi, z) + g_{\xi,v}(\xi, z)v + g_{\xi,w}(\xi, z)w$. The requisite conditions for akin to those for Case 2, except that $A(x)J(v, w)$ must now have lower-triangular structure:

$$A(x)J(v,w) = \begin{bmatrix} L_g E(x)\frac{\partial \Psi^{-1}}{\partial v}(v,w) & L_g E(x)\frac{\partial \Psi^{-1}}{\partial w}(v,w) \\ L_g R(x)\frac{\partial \Psi^{-1}}{\partial v}(v,w) & L_g R(x)\frac{\partial \Psi^{-1}}{\partial w}(v,w) \end{bmatrix} = \begin{bmatrix} L_g E(x)\frac{\partial \Psi^{-1}}{\partial v}(v,w) & \mathbf{0}_{m\times p-l} \\ L_g R(x)\frac{\partial \Psi^{-1}}{\partial v}(v,w) & L_g R(x)\frac{\partial \Psi^{-1}}{\partial w}(v,w) \end{bmatrix}. \quad (4)$$

In Case 4, the driven system must also be a leaf node.

Case 5 (Base Case): ***Full Coupling***. This is the final case, where $\dot{z} = f_z(z, \xi) + g_{z,v}(z, \xi)v + g_{z,w}(z, \xi)w$ and $\dot{\xi} = f_\xi(\xi, z) + g_{\xi,v}(\xi, z)v + g_{\xi,w}(\xi, z)w$. This case subsumes the case that the controls are fully coupled but states are decoupled (which is also a base case). In this case, the product $A(x)J(v, w)$ has no special structure, and there are no requirements on $L_f E(x)$ and $L_f R(x)$. **Both systems terminate here.**

Conclusion: ***Final Analysis and Future Directions***

In the previous section, we saw how a control system can be decomposed into a tree of subsystems by leveraging the structure of input-state relationships. However, we have yet to solve the problem of stabilization. For this, we must design a controller $\pi(x, \Psi^{-1}(v, w))$. Our approach to this will be hierarchical, designing the control-law first for driving child nodes, and then for their driven siblings.

Remark 1 (Challenges in Hierarchical Stabilizing Control Design) *We start with the base-case of the decomposition, fully-coupled leaf nodes of the ODE tree. Because these systems are minimal in their dimension, since they sit at the bottom of the tree, we can use any stabilizing control design to obtain $\pi_{\text{leaf}}(z(t), \xi(t)) = (v(t), w(t))$, without having to worry about computational complexity. Imagine we then move up the graph, and the next pair of nodes satisfies Case 3, with $\Psi^{-1}(\pi_{\text{leaf}}(z, \xi)) = v'$. Thus π_{leaf} is the control for a driving subsystem $f_{z'}(z') + g_{z'}(z')v'$, and has no way to control its corresponding driven subsystem $f_{\xi'}(\xi', z')$. In order for $\dot{\xi}'$ to be stable, it must be minimum phase (cf. Definition 8). The critical minimum phase property here must be established when designing the state decompositions Φ. This, and further complications that arise when propagating the leaf control signal up through the tree, will hopefully be studied in an extension of this document.*

Next, we briefly discuss the propagation of Lyapunov functions up the tree, with the goal of transforming a set of Lyapunov functions for leaf nodes into a Lyapunov function for the original system $f(x, u)$

Remark 2 (Recursive Composition of Lyapunov Functions) *Assume that we have Lyapunov functions V_z and V_ξ from child nodes (cf. Definition 6). In Cases 1, 2, 3, and 4, a parent Lyapunov function can be constructed as a sum of the two child functions. In cases 2, 3, and 4, we specifically use a notion called input-to-state stability (ISS) [8]. These cases use input-to-state stability due to the presence of a state cascade, where the driven subsystem receives the state of the driving subsystem as an input. ISS says that the state of the driven system is bounded when its input is bounded, and the stability of its input implies the stability of its state. In the case that the driven system is unactuated, this relates to the concept of minimum phase and stable zero dynamics. In Case 4, where there is both a state cascade and control coupling, ISS also allows the control the disturbance introduced by the control input to the driving system appearing in the driven system. A more rigorous discuss of combining Lyapunov functions should appear in the extentsion of this document.*

Finally, we briefly discuss the applicability of learning to this problem.

Remark 3 (Learning Diffeomorphisms) *In the case of partial feedback linearization, diffeomorphisms are typically computed analytically, starting by computing various Lie derivatives. For the degree-one diffeomorphisms discussed in the case-wise analysis, there may not be a simple analytical way to compute them. Nevertheless, there are a number of state-of-the-art machine learning techniques that encode diffeomorphims: for instance, [3], [7], [6].*

In conclusion, we have analyzed the problem of decomposing a control system (an ODE with control paremeter) into a tree of subsystems, and derived a number of important rules that characterize how stability of the subsystems relates to stability of the original system. Furthermore, we have identified a number critical problems to be analyzed in order to recursively design control laws and Lyapunov functions for the tree of systems. After further analysis, discussion, and reading, it is hoped that the ideas here might be expanded into novel research.

References

[1] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. Ieee, 2019.

[2] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[4] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[5] Alberto Isidori. *Nonlinear control systems: an introduction*. Springer, 1985.

[6] Matthew D Kvalheim and Eduardo D Sontag. Autoencoding dynamics: Topological limitations and capabilities. *arXiv preprint arXiv:2511.04807*, 2025.

[7] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[8] Eduardo D Sontag and Yuan Wang. On characterizations of the input-to-state stability property. *Systems & Control Letters*, 24(5):351–359, 1995.

[9] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644, 2024.

[10] Jiaming Zhou, Ke Ye, Jiayi Liu, Teli Ma, Zifan Wang, Ronghe Qiu, Kun-Yu Lin, Zhilin Zhao, and Junwei Liang. Exploring the limits of vision-language-action manipulations in cross-task generalization. *arXiv preprint arXiv:2505.15660*, 2025.