CIT/CDS 96-017 (revised)
To appear, Intl J. Robust & Nonl Control

REAL TIME TRAJECTORY GENERATION FOR DIFFERENTIALLY FLAT SYSTEMS

MICHIEL J. VAN NIEUWSTADT AND RICHARD M. MURRAY
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA 91125

Submitted, Int'l Journal of Robust and Nonlinear Control
May 13, 1997

Abstract. This paper considers the problem of real time trajectory generation and tracking for nonlinear control systems. We employ a two degree of freedom approach that separates the nonlinear tracking problem into real time trajectory generation followed by local (gain-scheduled) stabilization. The central problem which we consider is how to generate, possibly with some delay, a feasible state space and input trajectory in real time from an output trajectory that is given online. We propose two algorithms that solve the real time trajectory generation problem for differentially flat systems with (possibly non-minimum phase) zero dynamics. One is based on receding horizon point to point steering, the other allows additional minimization of a cost function. Both algorithms explicitly address the tradeoff between stability and performance and we prove convergence of the algorithms for a reasonable class of output trajectories. To illustrate the application of these techniques to physical systems, we present experimental results using a vectored thrust flight control experiment built at Caltech. A brief introduction to differentially flat systems and its relationship with feedback linearization is also included.

1. Introduction

A large class of industrial and military control problems consist of planning and following a trajectory in the presence of noise and uncertainty. Examples range from unmanned and remotely piloted airplanes and submarines performing surveillance and inspection tasks, to mobile robots moving on factory floors, to multi-fingered robot hands performing inspection and manipulation tasks inside the human body under the control of a surgeon. All of these systems are highly nonlinear and demand accurate performance.

Research supported in part by NSF Grant CMS-9502224 and AFOSR Grant F49620-95-1-0419.

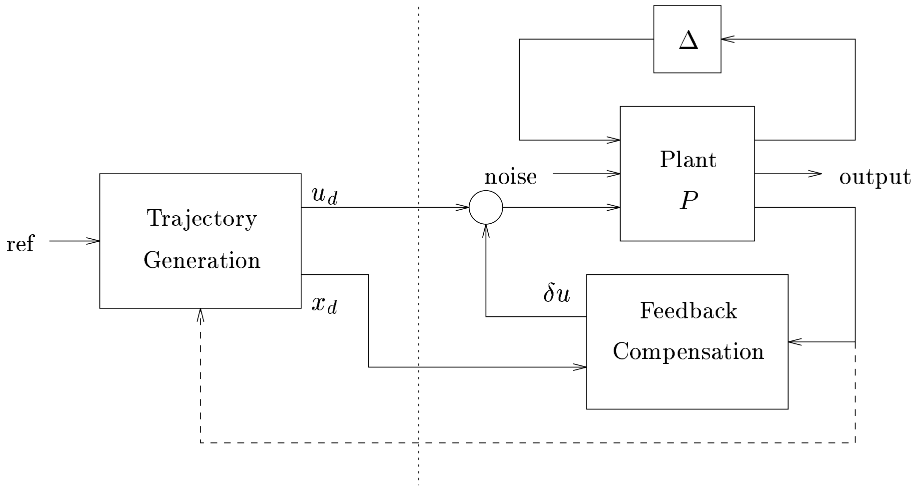

FIGURE 1. Two degree of freedom controller design.

In this paper, we focus on the problem of trajectory generation and tracking for such motion control systems. Roughly speaking, we wish to design a controller that tracks a desired trajectory for a set of outputs of the system. We assume that the desired trajectory is not known ahead of time, so that the controller must perform all operations in "real-time". A good prototype example for this class of problems is control of an autonomous aircraft tracking a moving target (perhaps a speeding car on the ground or another aircraft). The target's trajectory provides the desired trajectory for the pursuing aircraft. If the target and the controlled system have identical dynamics, then in principal one can achieve perfect tracking. In general, however, it will not be possible to exactly track the reference signal, and so we must tradeoff the tracking performance with the stability of the controlled system.

In order to design a nonlinear controller that approximately tracks a given trajectory, we separate the problem into two pieces: a *trajectory generation* block and a *feedback compensation* block, as shown in Figure 1.

The purpose of the trajectory generation block is to synthesize a feasible state space trajectory for the system given a desired reference signal. For example, the reference signal might be the desired position of the center of mass of an aircraft. This trajectory may or may not be something that can actually be executed by the aircraft, either due to limitations on the actuation system or because there does not exist a trajectory in state space that simultaneously satisfies the equations of motion and gives the desired trajectory for the center of mass. It is the responsibility of the trajectory generation block to use this reference signal to generate a *feasible* state space trajectory, as well as a set of nominal inputs that drive the system along this path.

Given the feasible state space trajectory, the feedback compensation block is used to correct for any errors due to noise or plant uncertainty (depicted in the figure as a feedback with unknown parameters Δ). Given the desired state space trajectory, x_d, the error dynamics can then be written as a time-varying control system in terms of the state error, $e = x - x_d$. Under the assumption that the tracking error remains small, we can linearize this time-varying system about $e = 0$ and stabilize the $e = 0$ state. One method of doing this is to solve the linear quadratic optimal control problem to obtain the optimal (time-varying) feedback gain for the path. More advanced techniques include the use of linear-time varying robust synthesis (see, for example, Shamma [46] for recent results and a survey of the literature) and the use of linear parameter varying synthesis developed by Packard [39] and others.

The use of two degree of freedom design techniques is a classical one in the control literature. In the linear setting, two degree of freedom controllers are usually thought of as containing a feedforward compensator (or shaping filter) that modifies the input to the feedback compensator. In our context, we do something slightly stronger since we generate a *full state*, feasible trajectory and nominal input, and we allow use of the current state in the "feedforward" block. This paradigm is also superficially similar to traditional optimal control techniques, where optimal control theory is used to generate a feasible trajectory minimizing some cost function and a feedback compensator is used to track this trajectory. However, for the motion control applications that we consider, the reference signal is not available ahead of time (as in the example of an airplane following a moving target) and the problems are typically high dimensional and nonlinear, making the optimal control problem difficult to solve in real-time.

Many modern nonlinear control methodologies can also be viewed as synthesizing controllers that fall into the two degree of freedom framework. For example, traditional nonlinear trajectory tracking approaches, such as feedback linearization [16, 19] and nonlinear output regulation [18], are easily viewed as a feedforward piece and a feedback piece. Both methods use the structure of the plant dynamics to compute a nominal path for the system and then stabilize the trajectory in local (possibly transformed) coordinates. Indeed, when the tracking error is small, the primary difference between the methods is the form of the error correction term: output regulation uses the linearization of the system about a single equilibrium point; feedback linearization uses a linear control law in an appropriate set of coordinates.

In some cases, it is possible only to linearize the input/output response of the system and not the full state dynamics. In this case, the feedback linearizing controller specifies only a portion of the dynamics of the system. The remaining internal dynamics (zero dynamics) are not specified and evolve independently, driven by the inputs required to track the output signal. If the internal dynamics are unstable (non-minimum phase), this technique cannot be used without modification. It is important to note that all of these modern nonlinear approaches rely on the availability of state feedback in order to generate feedforward commands. Thus, they are not traditional open loop feedforward controllers and this difference is crucial to their operation.

A more sophisticated approach for trajectory generation in the presence of internal dynamics is to extend the trajectory for the outputs and their derivatives to a full state space trajectory. One such method is reported by Chen and Paden [6] and Devasia and Paden [7], who use noncausal inversion for trajectory generation for systems with well-defined, hyperbolic zero dynamics. The method of noncausal inversion tries to find a stable solution for the full state space trajectory by steering from the unstable zero dynamics manifold to the stable zero dynamics manifold. The noncausality results from the fact that we first have to get from the origin to the right position on the unstable zero dynamics manifold. The solution is found by repeatedly solving a two point boundary value problem for the linearized zero dynamics driven by the desired trajectory. This iteration can also be performed in the frequency domain, as shown by Meyer et al. [30].

Finally, an approach that does not generate a feasible state space trajectory, but improves on the output-only trajectory has been explored by Getz et al. [14]. The method generates an approximate trajectory for the internal dynamics by following an instantaneous equilibrium for the internal dynamics. The first and higher order derivatives of the internal states are set to zero. Therefore the total state trajectory is not feasible. Further refinements of this technique can be found in [13].

In this paper we concentrate on a special class of systems, called differentially flat systems, for which there is a one-to-one correspondence between trajectories of a set of "flat outputs" and full state space and input trajectories. Trajectories can be planned in output space and then lifted to the state and input space, through an algebraic mapping. Differentially flat systems were introduced by Fliess et al. [10, 11] and are ideally suited for trajectory generation tasks. A variety of examples have been shown to be differentially flat (or approximately flat) and controllers based on trajectory generation by interpolation and then closing the loop on the obtained trajectory have been developed. This examples include overhead cranes [9, 12], cars with trailers [9, 44, 45], conventional aircraft [23, 28], induction motors [5, 27], active magnetic bearings [22], and chemical reactors [42, 43]. An introduction to differentially flat systems, and a description of their relationship with feedback linearizable systems, is given in Section 2.

The point of view taken in this paper also exploits differential flatness, but with a stronger emphasis towards real-time trajectory generation and explicit tradeoffs between performance in the tracking variables and stability of the internal variables. In Section 3 we introduce the real-time trajectory generation problem and provide two algorithms for tracking a (possibly non-minimum phase) output. These algorithms are shown to converge for a reasonable class of output trajectories and allow explicit tradeoff between stability and performance. We apply one of these algorithms to simulations and experiments of a flight control experiment in Section 4. We demonstrate that the two degree of freedom techniques advocated in this paper provide significant improvement over traditional linear controls. Finally, we summarize our results and outline some directions for future research in Section 5.

2. Differential Flatness

In this section we provide a brief description of differential flatness and compare it with (dynamic) feedback linearization. More details about differentially flat systems can be found in [10, 11, 22, 37]. For an introduction to feedback linearization, see [17, 38].

2.1. **Differentially flat systems.** Differential flatness was originally introduced by Fliess et al. [10] in a differentially algebraic context and later using Lie-Bäcklund transformations [11]. The important property of flat systems is that we can find a set of outputs (equal in number to the number of inputs) such that we can express all states and inputs in terms of those outputs and their derivatives. More precisely, a nonlinear system

$$\begin{aligned} \dot{x} &= f(x,u) \qquad && x \in \mathbb{R}^n,\, u \in \mathbb{R}^m \\ y &= h(x) && y \in \mathbb{R}^m \end{aligned} \tag{1}$$

is differentially flat if we can find outputs $z \in \mathbb{R}^m$ of the form

$$z = \zeta(x, u, \dot{u}, \dots, u^{(l)}) \tag{2}$$

such that

$$\begin{aligned} x &= x(z, \dot{z}, \dots, z^{(l)}) =: x(\bar{z}) \\ u &= u(z, \dot{z}, \dots, z^{(l)}) =: u(\bar{z}). \end{aligned} \tag{3}$$

We call y the *tracking outputs* and z the *flat outputs*. These outputs are not necessarily the same and care must be taken when the tracking outputs result in right half plane zeros for the linearized system, in order to avoid exciting instabilities in the internal dynamics. For ease of notation, we stack the flat outputs and their derivatives in the *flat flag*, $\bar{z} := (z, \dot{z}, \dots, z^{(l)})$.

Differentially flat systems are useful in situations where explicit trajectory generation is required. Since the behaviour of flat systems is determined by the flat outputs, we can plan trajectories in output space, and then map these to appropriate inputs. A common example is a kinematic car, where the xy position of the rear wheels provides flat outputs [44, 45]. This implies that all feasible trajectories of the system can be determined by specifying only the trajectory of the rear wheels. This idea is illustrated in more detail in the following sections.

Differential flatness can also be characterized using tools from exterior differential systems [37]. In the beginning of this century, the French geometer E. Cartan developed this set of powerful tools for the study of equivalence of systems of differential equations [2, 3, 47]. Equivalence need not be restricted to systems of equal dimensions. In particular a system can be *prolonged* to a bigger system on a bigger manifold, and equivalence between these prolongations can be studied. Two systems that have equivalent prolongations are

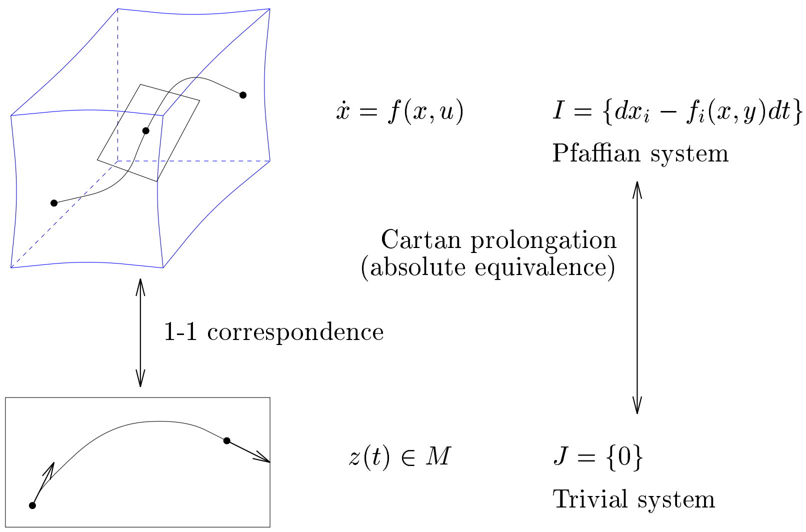

FIGURE 2. Differential flatness and absolute equivalence. A system $\dot{x} = f(x, u)$ is said to be differentially flat if the solutions are in one-to-one equivalence with the solutions of the trivial system $z(t) \in M$ (no dynamics).

called *absolutely equivalent*. Differentially flat systems can be interpreted as being those systems which are absolutely equivalent to the trivial system, i.e., having no dynamic constraints on the free variables [37]. This is illustrated in Figure 2.

2.2. **Flatness versus feedback linearization.** It is easy to show that any (full-state) feedback linearizable system is differentially flat by choosing the flat output as the feedback linearizing output. Indeed, differential flatness can be shown to be *equivalent* to dynamic feedback linearization on an open and dense set using a class of invertible dynamic feedbacks [4, 8, 19, 29]. Hence, the class of systems which is differentially flat is essentially the same as dynamically feedback linearizable systems (up to some regularity conditions). However, the point of view used in controlling differentially flat systems is substantially different than feedback linearization: one concentrates on *generating feasible trajectories* rather than transforming the system into a single linear system. Consistent with the two degree of freedom paradigm discussed in the introduction, a (scheduled) linear controller can then be used to maintain stability of the system around the generated trajectory. This has the advantage of allowing the local control design to be performed in the original coordinates for the system, where the various weights and other characteristics of the controller can be specified more naturally (see, for example, [1]).

To illustrate this viewpoint, consider the feedback linearization problem for a single input, single output nonlinear control system

$$\begin{aligned}\dot{x} &= f(x) + g(x)u\\ y &= h(x).\end{aligned}$$

Suppose that the output y has relative degree n, so that the system is full state linearizable. Then the dynamics of the system can be transformed to a linear system using a state transformation $\xi = \phi(x)$ and an input transformation $u = \alpha(x) + \beta(x)v$ where $\xi \in \mathbb{R}^n$ is the new state vector and v is the new input. The functions $\alpha(x)$ and $\beta(x)$ are determined using repeated Lie derivatives of the output function (see [38]). A standard control law for tracking a desired trajectory in y is one of the form

$$u = \alpha(x) + \beta(x)\big(y_d^{(n)} + K(\xi - \xi_d)\big), \tag{4}$$

where $y_d(\cdot)$ is the desired output trajectory and ξ_d is determined by differentiating the desired output (see [38] for a more complete description).

The control law in equation (4) can be viewed as a two degree of freedom controller by slightly rearranging terms:

$$u = \underbrace{\big(\alpha(x) + \beta(x)y_d^{(n)}\big)}_{\text{feedforward}} + \underbrace{\beta(x)K(\xi - \xi_d)}_{\text{feedback}}.$$

The feedforward terms are the inputs that are required in order to track the trajectory; the feedback terms are used to correct any errors due to system uncertainty. Note that the "feedforward" controller makes use of the current state information as well as the desired output signal, y_d.

Treating the system as being differentially flat instead of feedback linearizable, a controller for trajectory tracking would take the form

$$u = \big(\alpha(x, \bar{z}_d) + \beta(x, \bar{z}_d)z_d^{(n)}\big) + K(x, \bar{z}_d)(x - x_d(\bar{z}_d)). \tag{5}$$

There are several differences between this controller and the feedback linearizing controller. First, the nominal inputs are allowed to depend on both the current state and the flat flag, depending on how one implements certain computations. Thus, the entire trajectory can be computed in a truly open-loop fashion (as in optimal control) or the current state can be used (as in feedback linearization). Second, feedback of the error term in the feedback linearizing controller inverts the coupling function $\beta(x)$. This can lead to numerical problems if $\beta(x)$ is near singularities or the system has substantial uncertainty. In contrast, the controller in equation (5) uses a scheduled gain, allowing tradeoffs between performance and input magnitude to be varied depending on the operating conditions. Finally, as we show in the sequel, the flatness based controller allows numerical methods to be used for trajectory generation, rather than requiring symbolic computations.

Additional differences between flatness and feedback linearization are evident when the outputs that one wants to track are different than the flat outputs. In this case, standard I/O linearization techniques can fail if the zero dynamics of the system are unstable (non-minimum phase). And even

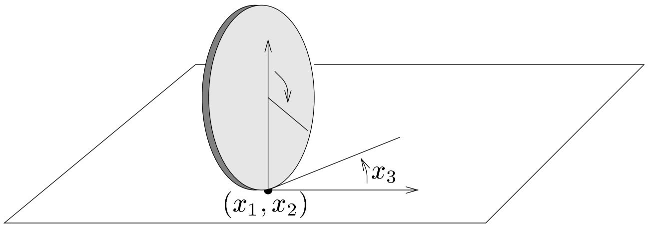

FIGURE 3. Rolling penny.

if the zero dynamics are stable, it is possible that the controller may cause the internal motion to have unacceptably large transient performance. The flatness based controllers which we develop in the next section avoid this problem by allowing an explicit tradeoff between the trajectory tracking performance and the stability of the internal motions (see [34] for a more complete discussion of this point).

2.3. **Examples.** In this subsection we give various examples of mechanical systems that are flat. Additional examples and characterizations of flat systems can be found in [9, 25, 26, 31, 32, 45].

Example 1 (Rolling penny)**.** Consider the motion of a rolling penny, as shown in Figure 3. Let (x_1, x_2) represent the xy position of the penny on the plane, x_3 represent the heading angle of the penny relative to a fixed line on the plane, and x_4 represent the rotational velocity of the angle of Lincoln's head, i.e., the rolling velocity. We restrict $x_3 \in [0, \pi)$ since we cannot distinguish between a positive rolling velocity x_4 at a heading angle x_3 and a negative rolling velocity at a heading angle $x_3 + \pi$.

The dynamics of the penny can be written as

$$\begin{aligned} \dot{x}_1 &= \cos x_3 \, x_4 \\ \dot{x}_2 &= \sin x_3 \, x_4 \\ \dot{x}_3 &= x_5 \\ \dot{x}_4 &= u_1 \\ \dot{x}_5 &= u_2, \end{aligned} \tag{6}$$

where $x_5 = \dot{x}_3$ is the velocity of the heading angle. The controls u_1 and u_2 correspond to the torques around the rolling and heading axes.

This system is differentially flat using the outputs x_1 and x_2 plus knowledge of time. Given x_1 and x_2, we can use the first two equations to solve uniquely for x_3 and x_4. Then given these three variables plus time, we can solve for all other variables in the system by differentiation with respect to time.

Example 2 (Planar ducted fan)**.** Consider the motion of the planar, vectored thrust vehicle shown in Figure 4. This system consists of a rigid body with body fixed forces and is a simplified model for the Caltech ducted fan described in [20].

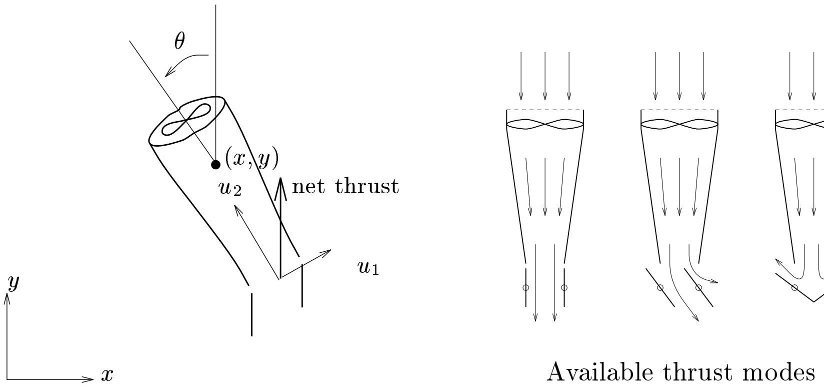

FIGURE 4. Planar ducted fan engine. Thrust is vectored by moving the flaps at the end of the duct, as shown on the right.

The ducted fan is mounted on a stand with a counterweight that moves in as the fan moves up. This results in inertial masses m_x and m_y in the x and y direction respectively, that change with the y coordinate. We do not take the variation of these inertial masses with y into account but take their value around hover. The counterweight also results in an effective weight m_g different than the inertial masses in x and y direction. We can apply any force on the center of mass by adjusting the magnitude and the direction of the thrust, or equivalently by commanding the parallel and perpendicular thrust. After shifting the control variables to compensate for gravity, $u_2 \to u_2 + m_g g$, the equations of motion are

$$\begin{pmatrix} m_x\ddot{x} \\ m_y\ddot{y} \\ J\ddot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \\ r & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 + m_g g \end{pmatrix} + \begin{pmatrix} 0 \\ -m_g g \\ 0 \end{pmatrix}, \tag{7}$$

where (x, y) are the coordinates center of the center of mass, θ is the angle with the vertical, u_1 is the force perpendicular to the fan shroud, u_2 is the force parallel to the fan shroud, r is the distance between the center of mass and the point where the force is applied, g is the gravitational constant, m_x, m_y is the inertial mass of the fan in the (x, y) direction respectively, $m_g g$ is the weight of the fan, and J is the moment of inertia. The tracking outputs are the (x, y) coordinates of the center of mass. Analogous to [7, 24], the flat outputs are

$$x_f = x - \frac{J}{m_x r}\sin\theta \qquad y_f = y + \frac{J}{m_y r}\cos\theta. \tag{8}$$

Note that these outputs are not fixed in body coordinates. The variable θ can be expressed in terms of the flat outputs as

$$\tan\theta = \frac{-m_x\ddot{x}_f}{m_y\ddot{y}_f + m_g g}. \tag{9}$$

From θ and the flat outputs we can find the other states and the inputs.

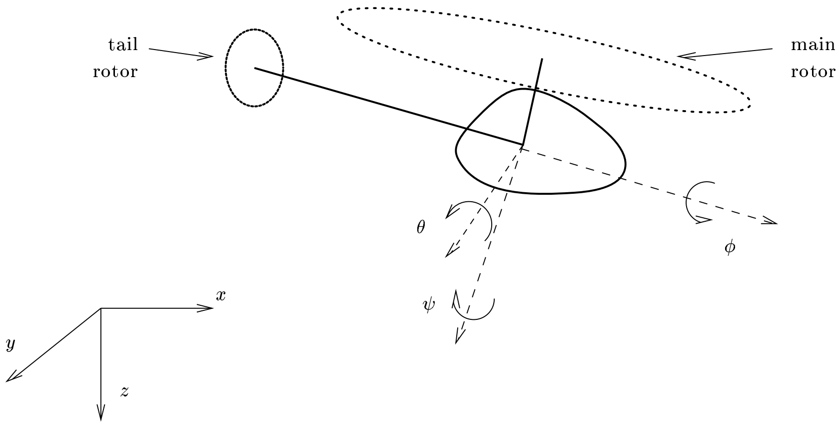

FIGURE 5. Simplified helicopter.

Example 3 (Simplified helicopter model)**.** Consider the simplified helicopter depicted in Figure 5. At some level of abstraction we can look at the helicopter as a rigid body actuated by the thrust of the main rotor and the tail rotor. The tail rotor exerts a thrust along the body y axis and a torque along the body z axis. The tail rotor force is small compared to the thrust of the main rotor and we neglect it. The main thrust is roughly aligned with the body z axis. We can measure the XYZ Euler angles (ϕ, θ, ψ). The tail rotor torque τ_b and the main thrust T_b then both act along the body z axis and can be transformed to spatial coordinates by rotations about the y and x axis about angles θ and ϕ respectively. The subscript b indicates that the vector is in body coordinates, the subscript s indicates spatial coordinates. Note that according to aerodynamic convention the z axis is positive pointing down, hence $T_b < 0$ is a thrust upward. Writing (x, y, z) for the center of mass in spatial coordinates, the rigid body equations for the model helicopter then take the form

$$\begin{pmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \\ J\ddot{\psi} \end{pmatrix} = \begin{pmatrix} T_b \sin\theta \\ -T_b \cos\theta \sin\phi \\ T_b \cos\phi \cos\theta + mg \\ \tau_b \cos\phi \cos\theta \end{pmatrix}, \tag{10}$$

where m is the mass of the helicopter, J is the moment of inertia about the z axis, and g is the gravitational acceleration. Note that we have no direct control over roll (ϕ) and pitch (θ) but only through left-right (aileron) and fore-aft (elevator) cyclic control respectively (see [34] for more details). The thrust T_b and the torque τ_b are real control inputs, the pitch angle θ and the roll angle ϕ are pseudo inputs.

The system of rigid body equations (10) is flat since from (x, y, z, ψ) we can recover the inputs (T_b, τ_b) and pseudo-inputs (ϕ, θ):

$$\begin{aligned} |T_b|^2 &= m^2(\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2) \\ \theta &= \arcsin \frac{m\ddot{x}}{T_b} \\ \phi &= -\arcsin \frac{m\ddot{y}}{T_b \cos\theta} \\ |\tau_b| &= \frac{\ddot{\psi}}{J \cos\phi \cos\theta} \end{aligned} \tag{11}$$

We cannot determine the sign of T_b, since flying right side up with positive thrust cannot be distinguished from flying upside down with negative thrust. We will assume that the the helicopter always flies right side up.

3. Real Time Trajectory Generation

In this section we will try to come to a meaningful definition of the real time trajectory generation problem and present some algorithms for generating feasible trajectories in real time, along with their proofs of convergence.

3.1. **Trajectory tracking: definition and limitations.** The notion of "real time" is somewhat ill defined and must be considered relative to the physical process under consideration. In our case we reference the time scale to the rate at which the reference signal is updated. A real time computation will therefore be one that can be performed faster than the reference update. As before, we consider nonlinear systems of the form

$$\begin{aligned} \dot{x} &= f(x, u) \qquad & x \in \mathbb{R}^n,\ u \in \mathbb{R}^m \\ y &= h(x) \qquad & y \in \mathbb{R}^m. \end{aligned} \tag{12}$$

Usually control objectives are stated as performance criteria subject to stability. For real time trajectory generation we only have a finite time history of the desired trajectory available, and therefore stability as defined in an infinite time horizon does not make sense. Instead we can capture the notion of stability as some norm bound on the internal dynamics generated when following a desired trajectory. The "performance under stability" requirement then translates to minimizing a weighted norm between tracking error and magnitude of the internal dynamics. In agreement with $\mathcal{H}^\infty$ control theory we take this norm to be the L_2 norm on a finite time interval. This leads to the following cost to be minimized at each time instant:

$$\int_{t-T_d}^{t} (h(x) - y_d(s))^*(h(x) - y_d(s)) + \lambda K(x, u) ds, \tag{13}$$

where K is an appropriate penalty on the internal dynamics, and T_d defines the time horizon, or the delay with which the trajectory is generated.

This formulation allows a tradeoff between performance and stability, as seen in [34, 35]. We can increase stability at the expense of performance by increasing the penalty on the internal dynamics (i.e. λ). Since we have to minimize the cost in equation (13) at every time instant, we need to do this subject to fixed initial conditions, namely, the state that we happen to be at.

There are theoretical limits to the tracking performance that can be obtained in systems with unstable zero dynamics, as was shown in a theorem by Grizzle et al. [15] that we repeat here for completeness. Recall that a left inverse of a system Σ is a system Σ_L that reconstructs the *unique* input that generates a given output of Σ, given that output and the initial state (see [38]).

Theorem 1 (Grizzle et al. [15]). *Suppose the nonlinear control system* (12)

1. *is analytic*
2. *possesses a zero dynamics manifold*
3. *is left invertible*
4. *has a controllable linearization,*

and let $Y(\epsilon, N) = \{y(t) \mid \|y(t)\| \leq \epsilon, \ldots, \|y^{(N)}(t)\| \leq \epsilon, \forall t\}$. *Then a necessary condition for asymptotic tracking of signals in* $Y(\epsilon, N)$ *for any* N *and* ϵ *is that the system have asymptotically stable zero dynamics.*

Note that this theorem shows that we cannot achieve asymptotic tracking even by decreasing the magnitude of the desired outputs and their derivatives. To achieve asymptotic tracking we need to relax either the analyticity requirement, or reduce the set of desired trajectories, or resort to some approximate scheme. The relevance of this theorem for trajectory generation is born out by the fact that trajectory generation combined with a linear controller based on the Jacobi linearization of the plant will achieve asymptotic tracking of signals in $Y(\epsilon, N)$ for N large enough and ϵ small enough. This follows from Lemma 4.5 in [21] and the fact that the higher order terms in the error system for $x - x_d$ are uniformly Lipschitz in time for desired signals in $Y(\epsilon, N)$. Hence asymptotically stable zero dynamics are also necessary for real time trajectory generation, unless we relax the conditions of Theorem 1 somehow. Theorem 1 is proven by the construction of a signal that cannot be asymptotically tracked by non-minimum phase systems. An essential feature of this signal is that it has a time derivative with infinite support. One way to circumvent the requirement for minimum phase zero dynamics is to restrict attention to asymptotic tracking of signals whose derivatives have finite support. More precisely, we make the following definition.

Definition 1 (eventually constant signals). The set of functions

$$S = \{y(t) \in \mathcal{L}_\infty(\mathbb{R}^m) \mid \exists\, t_s : \dot{y}(t) \equiv 0 \text{ for } t > t_s\}, \tag{14}$$

where t_s is not given in advance, is called the set of *eventually constant signals.*

Definition 2 (asymptotic trajectory generation). We say an algorithm achieves *asymptotic trajectory generation* for a class of signals Y if the algorithm generates from $y_d \in Y$ a feasible full state and input trajectory (x_d, u_d) such that $\lim_{t\to\infty} h(x_d(t)) - y_d(t) = 0$ for all $y_d \in Y$.

From the above discussion, it follows that asymptotic trajectory generation combined with a linear controller based on the Jacobi linearization (assuming the Jacobi linearization is controllable, as in the conditions of Theorem 1) results in local asymptotic tracking. In practice, the desired trajectory is only updated at discrete times, so we replace the continuous limit with $\lim_{k\to\infty} h(x_d(t_k)) - y_d(t_k) = 0$ in applications.

We require that our trajectory generation scheme achieve asymptotic trajectory generation for all signals in S. This comes down to requiring zero steady state error. Of course we need to make sure that eventually constant output signals lead to feasible state space trajectories. Hence the following assumption.

Assumption 1. We assume that to each value of the output y_d, there is an equilibrium value for the states and inputs, i.e. there exist (x_d, u_d) such that $y_d \equiv h(x_d)$, $f(x_d, u_d) \equiv 0$. We denote the mapping that maps each output value y_d to a full state and input space equilibrium by $\mathrm{Eq} : \mathbb{R}^m \to \mathbb{R}^{m+n}$, so that $f(\mathrm{Eq}(y_d)) \equiv 0$ and $h(\mathrm{Eq}(y_d)) = y_d$.

Based on the above discussion we pose the following problem:

Problem 1 (Real time trajectory generation). Find an algorithm that calculates in real time from $y_d(t)$ a feasible full state and input trajectory $(x_d(t), u_d(t))$ while trading off stability of the internal dynamics against tracking error, and such that $\lim_{t\to\infty} h(x_d(t)) - y_d(t) = 0$ for all $y_d \in S$.

One might object that this problem definition still allows the trajectory generation module to wait until the desired trajectory reaches its steady state value and then compute the trajectory offline. We still would achieve asymptotic trajectory generation. The key is that the time t_s after which $\dot{y}(t) \equiv 0$ is not given to us in advance, so that we cannot determine when to start the offline computation. This point is mainly philosophical, since it should be clear that it is better to start acting when sufficient knowledge of the desired trajectory is available.

3.2. **Point to point motion using flatness.** We begin by considering the problem of steering from an initial state to a final state. For flat systems, we parametrize the flat outputs z_i, $i = 1 \dots m$ by

$$z_i(t) = \zeta_i(x(t)) := A_{ij}\phi_j(t), \tag{15}$$

where the $\phi_j(t)$, $j = 1 \dots N$ are basis functions. This reduces the problem from finding a function in an infinite dimensional space to finding a finite set of parameters.

Suppose we have available to us an initial state x_0 at time τ_0 and a final state x_f at time τ_f. Steering from an initial point in state space to a desired point in state space is trivial for flat systems. We have to calculate the values of the flat outputs and their derivatives from the desired points in state space and then solve for the coefficients A_{ij} in the following system of equations:

$$\begin{aligned} z_i(\tau_0) &= A_{ij}\phi_j(\tau_0) \qquad z_i(\tau_f) = A_{ij}\phi_j(\tau_f) \\ &\vdots \qquad\qquad\qquad\qquad \vdots \\ z_i^{(l)}(\tau_0) &= A_{ij}\phi_j^{(l)}(\tau_0) \qquad z_i^{(l)}(\tau_f) = A_{ij}\phi_j^{(l)}(\tau_f). \end{aligned} \tag{16}$$

To streamline notation we write the following expressions for the case of *one* flat output only. The multi-output case follows by repeatedly applying the single output case, since the algorithm is decoupled in the flat outputs. Let $\Phi(t)$ be the $l+1$ by N matrix $\Phi_{ij}(t) = \phi_j^{(i)}(t)$ and let

$$\begin{aligned} \bar{z}_0 &= (z_1(\tau_0), \ldots, z_1^{(l)}(\tau_0)) \\ \bar{z}_f &= (z_1(\tau_f), \ldots, z_1^{(l)}(\tau_f)) \\ \bar{z} &= (\bar{z}_0, \bar{z}_f). \end{aligned} \tag{17}$$

Then the constraint in equation (16) can be written as

$$\bar{z} = \begin{pmatrix} \Phi(\tau_0) \\ \Phi(\tau_f) \end{pmatrix} A =: \Phi A. \tag{18}$$

That is, we require the coefficients A to be in the plane defined by equation (18). The only condition on the basis functions is that Φ is full rank, in order for (18) to have a solution.

3.3. **Basic algorithm for trajectory generation with point to point steering.** Suppose now that we are at time t, and have available to us the desired output trajectory over the time interval $[t - T_d, t]$, where T_d is a delay time. We consider $t - T_d$ and t as the initial and final time for a point-to-point steering trajectory, so we set $[\tau_0, \tau_f] := [t - T_d, t]$. For each interval $[\tau_0, \tau_f]$ we can generate a full state space trajectory from z_0 to z_f. On this trajectory we pick a state corresponding to some time $\tau \in [\tau_0, \tau_f]$ and use this as the instantaneous desired state for the linear controller. This simple idea is illustrated in Figure 6. The solid line is the desired reference output. At time t_k we know the reference between A_k and B_k. We then generate an arbitrary trajectory from A_k to B_k, depicted as the dashed line in Figure 6. We expand B_k to a full state and input by using the map $\mathrm{Eq}(\cdot)$. On this dashed line we pick a destination point, say C_k to be fed forward as the desired goal for sampling instant $k+1$. Then at sampling time $k+1$ we know the reference output up to point B_{k+1}. The intermediate point C_k takes the role of the initial point A_{k+1}, and we generate the dotted trajectory from A_{k+1} to B_{k+1}. Again we pick a point C_{k+1} as the desired goal for sampling time $k+2$. This process is repeated ad infinitum.

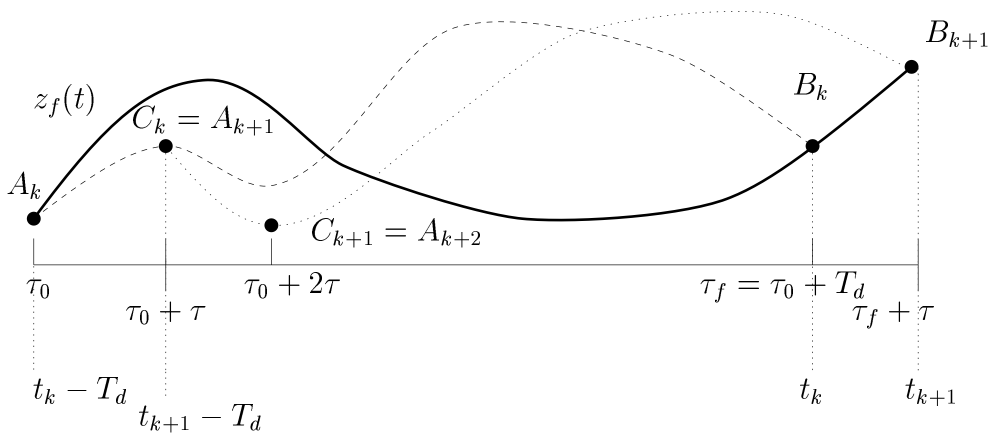

FIGURE 6. Description of algorithm for real time trajectory generation.

The generated trajectory can be anything that connects A_k to B_k, but a simple and elegant solution is obtained if we solve this as a simple point-to-point steering, as described in Section 3.2. This leads to the first algorithm:

Algorithm 1. Given: the delay time T_d, the current flat flag $\bar{z}_0$, the desired output y_d. At each sampling instant t_k:

1. Let $\tau_f = t_k$, $\tau_0 = t_k - T_d$, $\bar{z}_f = \bar{\zeta}(\mathrm{Eq}(y_d(t_k)))$.
2. Compute a trajectory of the flat outputs by solving $\bar{z}_0 = \Phi(\tau_0)A$, $\bar{z}_f = \Phi(\tau_f)A$ for A.
3. Compute a point on that trajectory with $\bar{z}_1(\tau) = \Phi(\tau)A$ where $\tau \in [\tau_0, \tau_f]$.
4. Solve for $(x_1(\tau), u_1(\tau))$ from $\bar{z}_1(\tau)$.
5. $(x_1(\tau), u_1(\tau))$ is the next desired state and input to feed forward at time t_k.

The times τ_* are "virtual" times within the algorithm that shift along as physical time proceeds. They are reassigned with every new sample time. The times t_* are physical times. This algorithm steers us from the current position to an equilibrium state with the desired values for the outputs. We generate a trajectory over the time interval $[t_k - T_d, t_k]$, and pick a time τ and corresponding point (x_1, u_1) on this trajectory. This will be the desired state to steer to. We repeat this process at every sampling instant.

Figure 7 illustrates the application of the algorithm to a step change in the desired output. Note that even though the input is delayed by T_d, the response to the reference input is immediate.

A particular feature of the point-to-point steering trajectory is that we can bypass solving for the coefficients A_{ij} in the matrix A by noting that

$$\bar{z}_1 = \Phi(\tau)\Phi^{-1}\bar{z} =: F(\tau)\bar{z}_0 + G(\tau)\bar{z}_f, \tag{19}$$

for some matrices F and G that only depend on τ. If we execute this scheme every sample instant we get a dynamical equation for $\bar{z}_1 =: \bar{z}_{k+1}$ for each

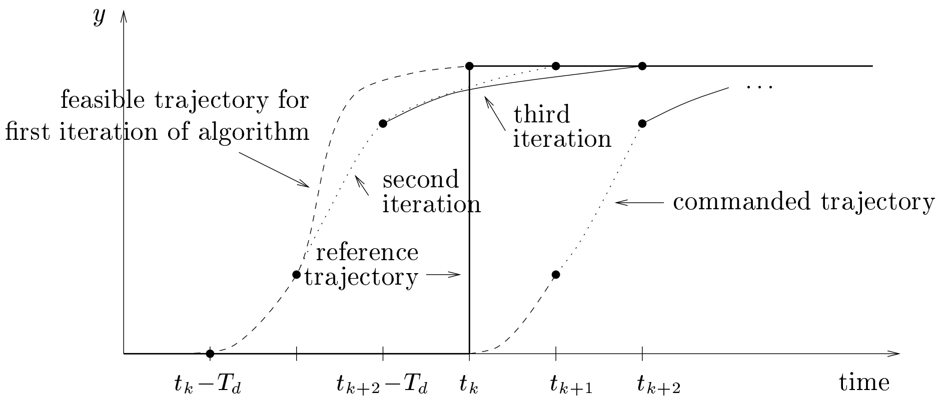

FIGURE 7. Trajectory shifting for the algorithm for real time trajectory generation.

$\bar{z}_0 =: \bar{z}_k$, namely

$$\bar{z}_{k+1} = F(\tau)\bar{z}_k + G(\tau)\bar{z}_f(k), \tag{20}$$

which has the desired output $\bar{z}_f(k) = \zeta(\mathrm{Eq}(y_d(t_k)))$ at time instant k for its input.

Proposition 1. *There is a $\tau \in]\tau_0, \tau_f[$ such that Algorithm 1 achieves real time asymptotic trajectory generation of all desired outputs in S.*

Proof. We will show that $F(\tau)$ is stable for appropriate choice of τ, and then that the steady state error is zero for $y_d \in S$. Since we constructed the $F(\tau)$, $G(\tau)$ to steer us from $\bar{z}_0$ to $\bar{z}_f$, it follows that $G(\tau_f) = 0$ and $F(\tau_f) = I$. So for $\tau = \tau_f$ all eigenvalues of $F(\tau)$ are at the origin. Since the eigenvalues of $F(\tau)$ are continuous functions of τ, there exists a $\tau \in [\tau_0, \tau_f]$ such that the eigenvalues of $F(\tau)$ are in the open unit circle. Now $y_d \in S$ means that there is a k_s such that $\bar{z}_f(k)$ is a constant, say $\bar{z}_f$, for all $k > k_s$. Therefore $\bar{z}_k$ converges to a constant value, say $\bar{z}_\infty$ which will be a multiple of $\bar{z}_f$ due to linearity of (20). So $\bar{z}_\infty = \gamma\bar{z}_f$, where γ depends only on F and G, and not on z_f. Since there is a trajectory from $\bar{z}_\infty$ to $\bar{z}_f$ that will bring us closer to $\bar{z}_f$ for appropriate choice of τ, we have $\gamma = 1$ for that value of τ. Then we have $\lim_{k\to\infty} \bar{z}_k = \bar{z}_f$. □

This algorithm does not involve the explicit minimization of a cost function to trade off stability versus performance. However, the choice of T_d can be used to affect this tradeoff. Picking $T_d = t_k - t_{k-1}$ in the above scheme corresponds to a one step deadbeat controller. This requires large control signals which might saturate the actuators and generate unacceptable internal motion. Increasing T_d will increase stability at the expense of performance, as we demonstrate below in Section 4.1.

Note that the matrices $F(\tau)$ and $G(\tau)$ are fixed once τ is selected, and can be computed ahead of time. We should mention that it is not hard to find τ such that $F(\tau)$ is stable. In fact, it requires considerable effort to construct a set of basis functions and a τ such that $F(\tau)$ is unstable. For polynomial basis functions any $\tau \in]\tau_0, \tau_f[$ will do. This follows from the fact that the degree of a polynomial is an upper bound on the number of its zeros.

Finally, we point out the important role played by the equilibrium map $\mathrm{Eq}(\cdot)$ in Algorithm 1. It serves to extend the final reference output to a full state, allowing the point-to-point computation to be performed. For some systems, it may not be possible to find an equilibrium point corresponding to every output. A straightforward extension of Algorithm 1 for this case is to replace the map $\mathrm{Eq}(\cdot)$ by a map $\widetilde{\mathrm{Eq}} : \mathbb{R}^m \to \mathbb{R}^{n+m}$ that only maps some output values to an equilibrium and replace the set of eventually constant signals by

$$\widetilde{S} = \{y(t) \in \mathcal{L}_\infty(\mathbb{R}^m) | \; \exists\, t_s : \dot{y}(t) \equiv 0 \text{ for } t > t_s \text{ and } f(\widetilde{\mathrm{Eq}}(y(t_s))) \equiv 0\}. \tag{21}$$

That is, we only require existence of an equilibrium for a set of outputs that we are interested in. The preceding development carries through directly and Algorithm 1 converges as before.

3.4. **Improved algorithm with performance optimization.** Step 2 in Algorithm 1 computes a trajectory between the flat flags $\bar{z}_0$ and $\bar{z}_f$ by using a point-to-point steering algorithm. In fact, we can use any trajectory that links z_0 to z_f. It just so happens that the point-to-point steering problem is particularly attractive since it results in a linear update for the flat flag, as in equation (19). In particular, we can augment this algorithm with an additional minimization that allows tradeoff between stability and performance as discussed in Section 3. The cost criterion takes the form

$$J = \min_A \int_{\tau_0}^{\tau_f} (y(A,s) - y_d(s))^* (y(A,s) - y_d(s)) + \lambda K(A,s) ds, \tag{22}$$

subject to $\bar{z}_0 = \Phi(\tau_0)A, \bar{z}_f = \Phi(\tau_f)A$. Here y is the tracking output, and y_d the desired tracking output. K is a function that bounds the internal dynamics. We can perform this minimization by finding a particular solution that satisfies the initial and final constraints, $A_0 = \Phi^\dagger \bar{z}$, and parametrizing the general solution as $A = A_0 + \Phi^\perp A_1$ where $\Phi^\perp$ is a basis for the nullspace of Φ. This optimization problem is in general nonlinear and nonconvex. We therefore have to resort to an iterative scheme. Since the optimization has to be performed in real time, we might not be able complete the minimization procedure and have to preempt the procedure. We will show that this will not result in loss of convergence. This leads to the following algorithm:

Algorithm 2. Given: the delay time T_d, the current flat flag $\bar{z}_0$, the desired output y_d. At each sampling instant t_k:

1. Let $\tau_f = t_k$, $\tau_0 = t_k - T_d$, $\bar{z}_f = \bar{\zeta}(\mathrm{Eq}(y(t_k)))$.

2a. Compute a trajectory for the flat outputs by finding a particular solution A_0 to $\bar{z}_0 = \Phi(\tau_0)A$, $\bar{z}_f = \Phi(\tau_f)A$.

2b. Optimize A_1 to minimize J in equation (22).

2c. Let $A = A_0 + N^\perp A_1$.

3. Compute a point on the nominal trajectory with $\bar{z}_1(\tau) = \Phi(\tau)A$ where $\tau \in [\tau_0, \tau_f]$.
4. Solve for $(x_1(\tau), u_1(\tau))$ from $\bar{z}_1(\tau)$.
5. $(x_1(\tau), u_1(\tau))$ is the next desired state and input to feed forward at time t_k.

Note that the optimization over A_1 can be preempted if computation time runs out.

Proposition 2. *There is a $\tau \in [\tau_0, \tau_f]$ such that Algorithm 2 achieves real time asymptotic trajectory generation of all desired outputs in S.*

Proof. We will show that $\bar{z}_1$ converges to a constant value for constant $\bar{z}_f$, and then that this constant value equals $\bar{z}_f$. Even though we cannot dispense with the computation of the coefficients A as we could in Algorithm 1, we know that for $\tau = \tau_f$ the algorithm steers to the desired output in one step. Regardless of the values for A_1, from continuity of $\bar{z}_1 = \Phi(\tau)A$ in τ, we can find a τ such that

$$\|\bar{z}_1 - \bar{z}_f(k)\| < \|\bar{z}_0 - \bar{z}_f(k)\| \tag{23}$$

so that if $\bar{z}_f(k)$ is a constant for $k \geq k_s$, say $\bar{z}_f$, we achieve convergence to a constant value for $\bar{z}_k$. Similar to the proof of Proposition 1 we can show that this constant value has to be $\bar{z}_f$. □

It might seem curious at first sight that convergence of Algorithm 2 does not depend on the cost criterion J. On second thought this is quite advantageous since we cannot guarantee that the optimization of J converges in the allotted computation time. Preemption of the minimization will not result in loss of convergence. The additional optimization allows us to get better performance (in the sense of a lower cost criterion J) if the computation time allows it. If no improvement can be obtained, the algorithm returns the solution of the point-to-point steering algorithm 1. This is essential for convergence.

As a final comment, we note that it is somewhat unsatisfactory that we have to fix the final conditions in the above algorithms. For nonlinear systems we have in general multiple equilibria corresponding to the same output values, most of which are undesirable. Without fixing the final condition we cannot guarantee that the trajectory will converge to the desired equilibrium, even though we still get asymptotic tracking for signals in S. Indeed, simulations showed that the trajectory might end up in an undesired equilibrium.

4. Application to the Caltech Ducted Fan

In this section we apply the real time trajectory generation algorithms to the Caltech ducted fan, a thrust vectored, flight control experiment pictured in Figure 8.

The numerical computations in this section are done using the trajectory generation library `tglib`, developed by M. van Nieuwstadt at Caltech. This

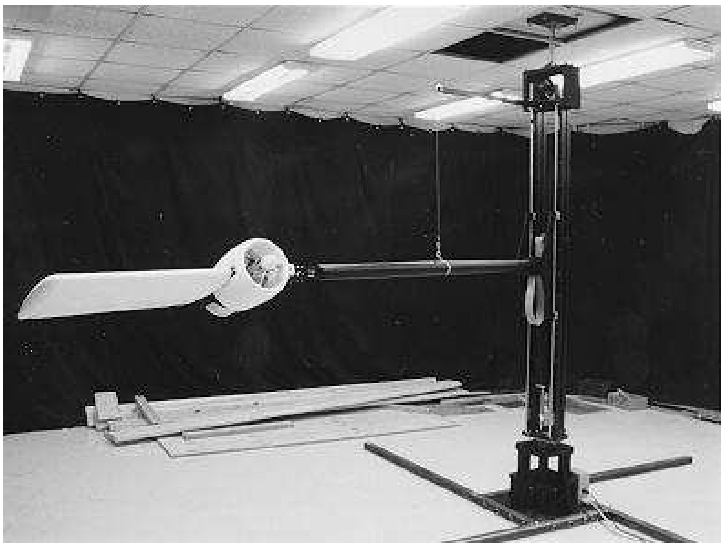

FIGURE 8. Photograph of the Caltech ducted fan experiment (courtesy of R. Bodenheimer).

library is available through anonymous ftp from `avalon.caltech.edu` in the file `/pub/vannieuw/software/trajgen.tar.gz`. This file contains the libraries, examples and documentation. The routines are in ANSI C and will compile under different platforms. In particular, the simulations presented in this paper used the library compiled on a UNIX platform, and the real time experiments used the library compiled under MS-DOS.

4.1. **Simulations.** We first simulate the real time algorithms to study the role of the parameter T_d and the tradeoff between tracking and stability. For this study we have used a nonlinear model of the *original* Caltech ducted fan, described in detail in [20]. The main difference between this model and the experiment shown in Figure 8 is the stand dynamics are slightly more complicated for the original experiment and no wing was attached to the fan unit. The simulation model takes into account the nonlinear stand dynamics, aerodynamic drag, inertial effects from the rotating propeller, and viscous friction. The flat model in equation (7) is used to generate the nominal trajectories and only models the dynamics of the thrust and gravity.

We simulate the reference input as a file from which successive samples are read every sample instant. The reference trajectory is input to a trajectory generation module, whose output serves as a nominal trajectory. We wrap a simple static LQR controller around a nonlinear model of the fan. This controller was designed to stabilize hover. See [20] for a detailed analysis of several controller designs for this experiment. We assume we have knowledge of the full state. On the experiment this is achieved by differentiating and filtering the position signals.

First we show how the ducted fan behaves without feedforward. The reference input is used to generate an error signal to the controller. The

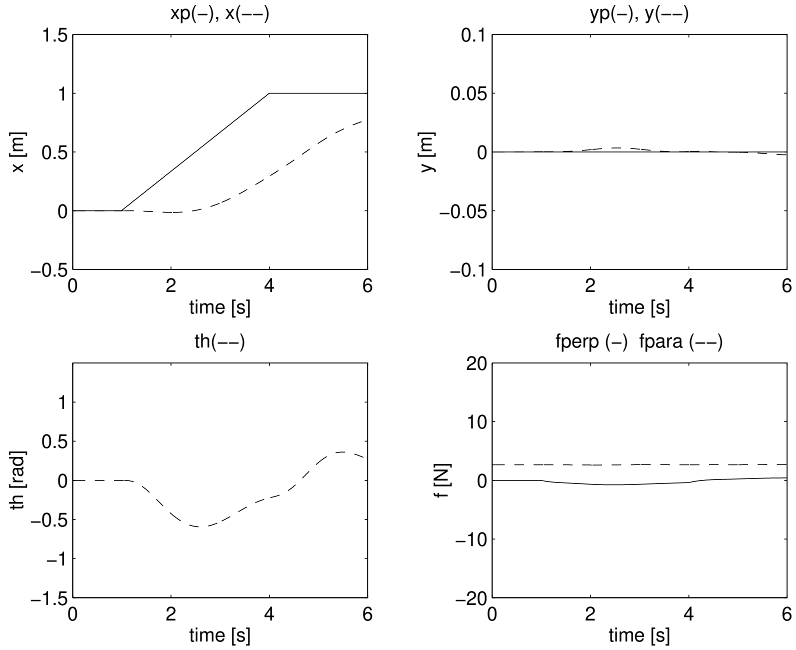

FIGURE 9. Ducted fan simulation: tracking with one degree of freedom controller.

reference input is a 1 meter ramp in 3 seconds in the x direction at constant altitude. At each time instant we stabilize around the equilibrium point generated by setting x and y equal to the reference input, and all other states equal to zero. This is the conventional "one degree of freedom" controller. Figure 9 shows that the trajectory followed by the fan lags far behind the desired trajectory.

In this plot and subsequent plots, the reference input is denoted by (xp, yp), the generated desired trajectory (identical to the reference input in the one degree of freedom case) is denoted (xd, yd) and the variable name without a suffix denotes the real (experimental or simulated) time trace of a quantity. The force parallel to the fan shroud is denoted "fpara", the force perpendicular to the fan shroud is called "fperp".

Next we show for Algorithm 1 plots of the reference input, the generated trajectory, the simulated trajectory for the (x, y) position of the fan, as well as the generated and simulated trajectory for θ and the nominal forces.

Figure 10 shows these for a delay time of $T_d = 60$ sampling periods of $T_s = 0.01$ seconds. We see that the fan follows the reference input much better than in the one degree of freedom design. Clearly, there is an advantage in real time trajectory generation. Figure 11 shows these for a delay time of $T_d = 100$ sampling periods. It is clear that the larger delay results in better stability, i.e. lower magnitude of θ and the nominal forces, but poorer performance, since the delay is bigger.

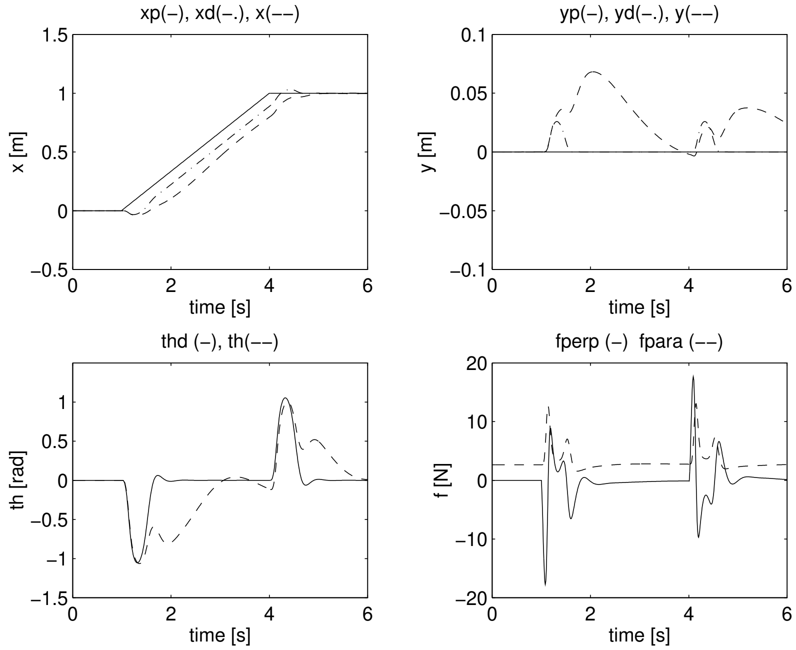

FIGURE 10. Ducted fan simulation: tracking with algorithm 1, $T_d = 60 \times T_s = 0.6$ s.

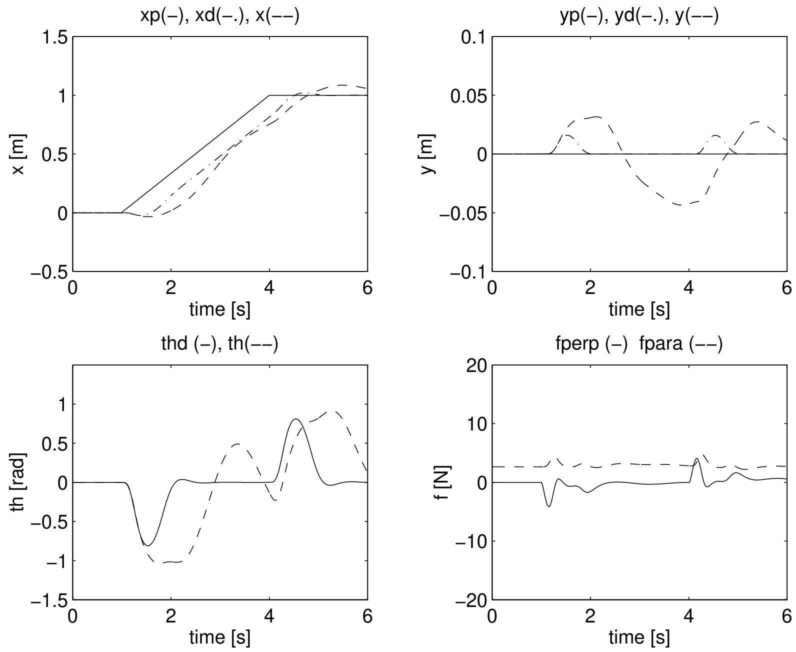

FIGURE 11. Ducted fan simulation: tracking with algorithm 1, $T_d = 100 \times T_s = 1.0$ s.

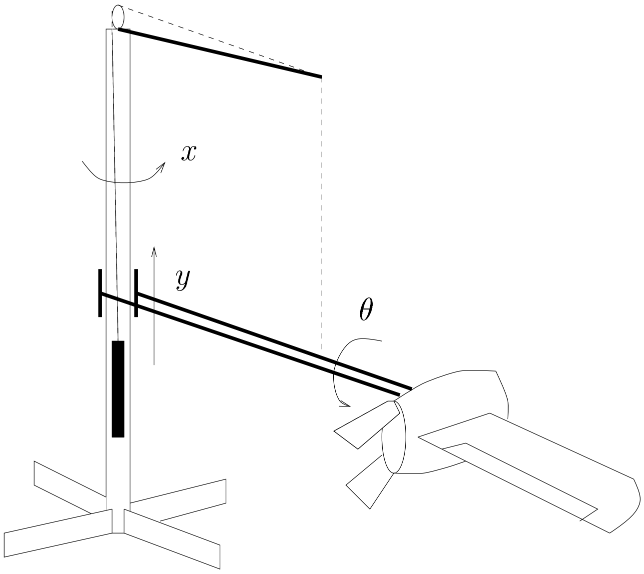

FIGURE 12. Ducted fan with stand.

Figures 10 and 11 both show a large error in θ right after the first and second peak of the nominal θ trace. We suspect this is caused by the inertia changing with altitude, which is not taken into account in the nominal flat model, but is present in the simulation model. Note that the errors occur simultaneously with a substantial error in altitude y.

Algorithm 2 was tested in simulation and the results are reported in [34, 35]. It behaves as expected, in the sense that it penalizes the cost. The major problem is that the optimization is a factor 10 too slow for realistic operator sampling rates. Improvement of this optimization is a subject of current research.

4.2. **Hover to hover experiments.** To demonstrate the viability of the two degree of freedom approach, we first show some hover to hover transitions, both with one and two degree of freedom controllers. These experiments are performed on the Caltech ducted fan experiment. The ducted fan is mounted on a stand, as shown in Figure 12, and is controlled by an Intel 486 66 MHz PC. It uses a linear current amplifier to regulate the current to the propeller, and PWM servos to steer the paddles. It has a NACA 0015 airfoil to generate lift, although for the experiments presented in this paper, the lift forces are negligible. Horizontal, vertical and pitch position are measured with encoders. Velocities are obtained by numerical differentiation and smoothing. Closed loop control of the ducted fan is accomplished using the Sparrow real time kernel [33]. Sparrow takes care of reading sensor input, writing actuator output, data logging, and necessary controller computations.

The dynamics of the ducted fan are essentially those given in Example 2, with additional aerodynamic forces playing a role at high flight speeds. In the absence of these forces, the fan dynamics are very close to the flat model. Therefore, for the hover to hover transitions considered in this section, the flat model provides a good approximation to the actual dynamics of the system.

The computations in Section 3.2 were used to generate point-to-point motion of the system. The desired trajectory is a 4 meter step in the positive x direction in 4 seconds. The one degree of freedom controller uses a linear interpolation between initial and final state while keeping pitch zero. The two degree of freedom controller generates a non-zero nominal pitch trajectory. Figures 13 and 14 show that the two degree of freedom controller achieves considerable performance increase. The steady-state error in y is due to stiction in the stand.

4.3. **Real time trajectory generation experiments.** Algorithm 1 was also implemented on the experimental apparatus, with the reference input comes from a joystick with two degrees of freedom. In order to obtain repeatable experiments, we record the joystick input and read it back from a file for each trial (but without making use of the knowledge of future inputs). The joystick is interpreted as a commanded velocity signal, so that the reference trajectory for the (x, y) position is given by

$$\begin{aligned} x_p(k) &= f_x \sum_{i=0}^{k} \eta_x(i) * T_s \\ y_p(k) &= f_y \sum_{i=0}^{k} \eta_y(i) * T_s, \end{aligned} \tag{24}$$

with $\eta_{x,y}$ the joystick command in the (x, y) direction respectively, and (f_x, f_y) some scaling factors. We run the trajectory generation algorithm at 100 Hz, and the controller at 200 Hz. The delay time $T_s = 1.0$ seconds, corresponding to 100 samples for the trajectory generation algorithms.

We conducted 2 experiments. The first one was the one degree of freedom controller: the reference signal was used to generate an error signal in the output around which the fan was stabilized. As in the simulations, the reference is denoted by (xp, yp), the desired trajectory by (xd, yd), and the measured position by (x, y). In the one degree of freedom case $(xp, yp) = (xd, yd)$. The results are depicted in Figure 15.

In the second experiment, we used the reference signal to generate a trajectory, as described in this paper. The desired trajectory (xd, yd) is generated by the trajectory generation module and is no longer equal to the reference input (xp, yp). The results are depicted in Figure 16. The real time trajectory generation algorithm gives a more aggressive response.

5. Summary

In this paper, we have proposed a formulation for the real time trajectory generation problem which is compatible with a two degree of freedom approach for tracking in motion control algorithms. We considered only the

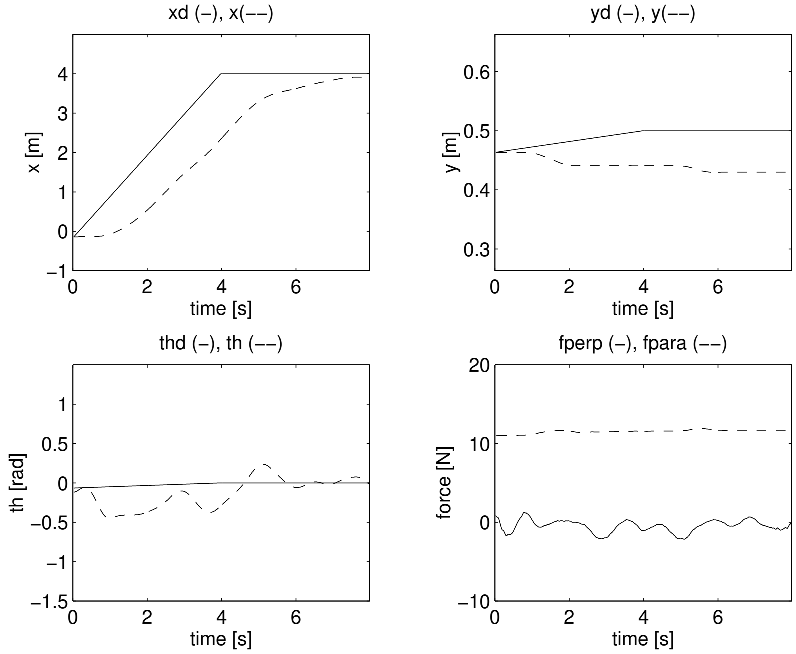

FIGURE 13. Ducted fan experiment: hover to hover transition for one degree of freedom controller.

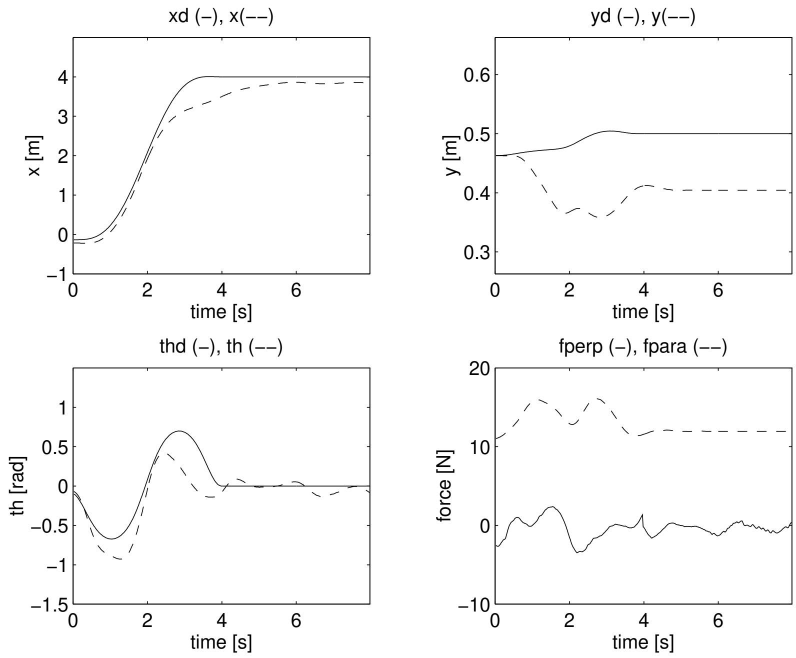

FIGURE 14. Ducted fan experiment: hover to hover transition for two degree of freedom controller.

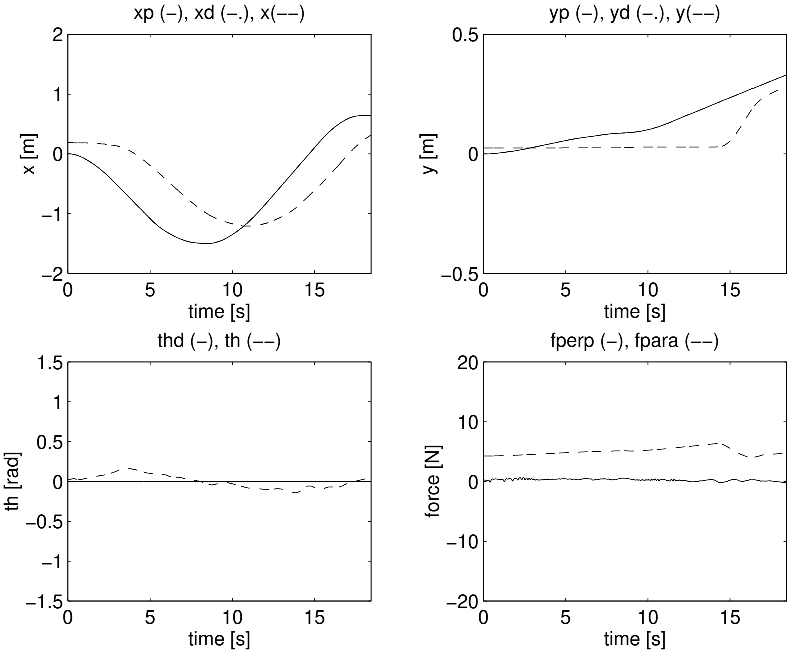

FIGURE 15. Ducted fan experiment: tracking with one degree of freedom controller.

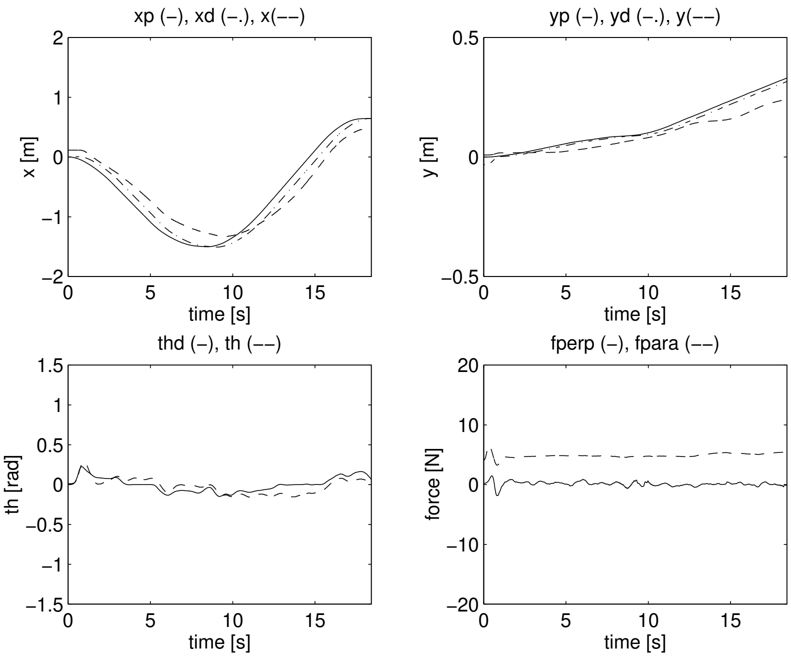

FIGURE 16. Ducted fan experiment: tracking with real time trajectory generation.

case of differentially flat systems, for which the trajectory generation problem can be formally reduced to an algebraic problem in terms of the flat outputs for the system. By explicitly separating the tracking problem into a trajectory generation block and a feedback compensation block, we are able to exploit the geometric structure of differentially flat systems without transforming the system into a single linear system. We thereby avoid some of the pitfalls commonly associated with feedback linearization and allow more freedom in the control design.

We have described two algorithms for real time trajectory generation for differentially flat systems with unstable zero dynamics, and proved stability and convergence properties. The first algorithm generated a trajectory that steers from the current position to a desired final position given by the reference input. We can trade off stability versus performance by varying the delay time. The second algorithm steers to a desired final position while minimizing a cost criterion, that typically limits the magnitude of the zero dynamics and/or the control inputs. The current implementation of the minimization is too slow for real time implementation. Improving this algorithm is a subject of current research.

We have implemented the trajectory generation algorithms on a vectored thrust, flight control experiment constructed at Caltech. Experimental results show that the two degree of freedom controllers perform much better than simpler linear controllers. For the experiments in this paper, the fan was flown at low speeds. More aggressive trajectories have also been tested [34, 36].

Much work remains to be done on differentially flat systems, both from the theoretical perspective and in the context of applications. At the present, constructive conditions for finding the flat outputs of a mechanical system are not available except in a few special (i.e. low-dimensional) cases [40, 41]. In addition, for systems which are not differentially flat, it is likely that approximations can be used which will allow fast and efficient generation of approximately feasible trajectories. Bounds on the sizes of the error in the performance of the system as a function of the degree of approximation will be needed in order to pursue efforts in this direction.

Acknowledgments. The authors would like to thank Mark Milam for his work on the ducted fan experiment.

References

[1] F. Bullo and R. M. Murray. Experimental comparison of trajectory trackers for a car with trailers. In *IFAC World Conference*, pages 407–412, 1996.

[2] E. Cartan. Sur l'équivalence absolue de certains systèmes d'équations différentielles et sur certaines familles de courbes. In *Œuvres Complètes*, volume II, pages 1133–1168. Gauthier-Villars, 1953.

[3] E. Cartan. Sur l'intégration de certains systèmes indéterminés d'équations différentielles. In *Œuvres Complètes*, volume II, pages 1169–1174. Gauthier-Villars, 1953.

[4] B. Charlet, J. Lévine, and R. Marino. On dynamic feedback linearization. *Systems and Control Letters*, 13:143 – 151, 1989.

[5] K. Chelouah, E. Delaleu, P. Martin, and P. Rouchon. Differential flatness and control of induction motors. *Multiconference on Computational Engineering in Systems Applications*, pages 80–85, 1996.
[6] D. G. Chen and B. Paden. Stable inversion of nonlinear nonminimum-phase systems. *International Journal of Control*, 64(1):81–97, 1996.
[7] S. Devasia and B. Paden. Exact output tracking for nonlinear time-varying systems. In *Proc. IEEE Control and Decision Conference*, pages 2346–2355, 1994.
[8] M. Fliess, J. Lévine, P. Martin, F. Ollivier, and P. Rouchon. Flatness dynamic feedback linearizability: two approaches. In *Proc. European Control Conference*, 1995.
[9] M. Fliess, J. Levine, P. Martin, and P. Rouchon. Flatness and defect of nonlinear systems: Introductory theory and examples. *International Journal of Control*, 61(6):1327–1361, 1995.
[10] M. Fliess, J. Lévine, Ph. Martin, and P. Rouchon. Sur les systèmes non linéaires différentiellement plats. In *C. R. Acad. Sci. Paris, t. 315, Série I*, pages 619–624, 1992.
[11] M. Fliess, J. Lévine, Ph. Martin, and P. Rouchon. Linéarisation par bouclage dynamique et transformations de Lie-Bäcklund. In *C. R. Acad. Sci. Paris, t. 317, Série I*, pages 981–986, 1993.
[12] M. Fliess, J. Lévine, and P. Rouchon. Genearlized state variable representation for a simplified crane description. *International Journal of Control*, 58(2):277–283, 1993.
[13] N. H. Getz. *Dynamic Inversion of Nonlinear Maps with Applications to Nonlinear Control and Robotics*. PhD thesis, UC Berkeley, Berkeley, California, 1995.
[14] N. H. Getz and K. Hedrick. An internal equilibrium manifold method of tracking for nonlinear nonmimimum phase systems. In *Proc. IEEE Control and Decision Conference*, pages 2241–2245, 1995.
[15] J. W. Grizzle, M. D. Di Benedetto, and F. Lamnabhi-Lagarrigue. Necessary conditions for asymptotic tracking in nonlinear systems. *IEEE Transactions on Automatic Control*, 39(9):1782–1795, 1994.
[16] L. R. Hunt, R. Su, and G. Meyer. Global transformations of nonlinear systems. *IEEE Transactions on Automatic Control*, AC-28(1):24–31, 1983.
[17] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 2nd edition, 1989.
[18] A. Isidori and C. I. Byrnes. Output regulation of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(2):131–140, 1990.
[19] B. Jakubczyk and W. Respondek. On linearization of control systems. *Bulletin de L'Academie Polonaise des Sciences, Série des sciences mathématiques*, XXVIII:517–522, 1980.
[20] M. Kantner, B. Bodenheimer, P. Bendotti, and R. M. Murray. An experimental comparison of controllers for a vectored thrust, ducted fan engine. In *Proc. American Control Conference*, page 1956, 1995.
[21] H. Khalil. *Nonlinear Systems*. Macmillan Publishing Company, 1992.
[22] J. Lévine, J. Lottin, and J. C. Ponsart. A nonlinear approach to the control of magnetic bearings. *IEEE Transactions on Control Systems Technology*, 5(524–544), 1996.
[23] P. Martin. Aircraft conrol using flatness. In *Multiconference on Computational Engineering in Systems Applications*, pages 194–199, 1996.
[24] P. Martin, S. Devasia, and B. Paden. A different look at output tracking—Control of a VTOL aircraft. *Automatica*, 32(1):101–107, 1994.
[25] P. Martin and P. Rouchon. Feedback linearization and driftless systems. *Math. Control Signal Syst.*, 7:235–254, 1994.
[26] P. Martin and P. Rouchon. Any (controllable) driftless system with 3 inputs and 5 states is flat. *Systems and Control Letters*, 25:167 – 173, 1995.
[27] P. Martin and P. Rouchon. Flatness and sampling control of induction motors. In *Workshop IFAC Symposium*, 1996.
[28] Ph. Martin. *Contribution à l'étude des systèmes différentiellement plats*. PhD thesis, L'Ecole Nationale Supérieure des Mines de Paris, 1993.

[29] Ph. Martin. Endogenous feedbacks and equivalence. In *Mathematical Theory of Networks and Systems*, Regensburg, Germany, 1993.

[30] G. Meyer, L. R. Hunt, and R. Su. Nonlinear system guidance. In *Proc. IEEE Control and Decision Conference*, pages 590–595, 1995.

[31] R. M. Murray. Trajectory generation for a towed cable system using differential flatness. In *IFAC World Conference*, pages 395–400, 1996.

[32] R. M. Murray, M. Rathinam, and W. Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In *asmeIMECE*, San Francisco, 1995.

[33] R. M. Murray, E. L. Wemhoff, and M. Kantner. *Sparrow 2.1 Reference Manual*. California Institute of Technology, 1995. Available electronically from `http://avalon.caltech.edu/~sparrow`.

[34] M. van Nieuwstadt. *Trajectory Generation for Nonlinear Control Systems*. PhD thesis, California Institute of Technology, 1996.

[35] M. van Nieuwstadt and R. M. Murray. Approximate trajectory generation for differentially flat systems with zero dynamics. In *Proc. IEEE Control and Decision Conference*, pages 4224–4230, 1995.

[36] M. van Nieuwstadt and R. M. Murray. Fast mode switching for a thrust vectored aircraft. In *Multiconference on Computational Engineering in Systems Applications*, Lille, France, 1996.

[37] M. van Nieuwstadt, M. Rathinam, and R. M. Murray. Differential flatness and absolute equivalence. In *Proc. IEEE Control and Decision Conference*, pages 326–333, 1994. Also available as Caltech technical report CIT/CDS 94-006.

[38] H. Nijmeijer and A. van der Schaft. *Nonlinear Dynamical Control Systems*. Springer Verlag, 1990.

[39] A. Packard. Gain scheduling via linear fractional transformations. *Systems and Control Letters*, 22(2):79–92, 1994.

[40] M. Rathinam and R. Murray. Configuration flatness of lagrangian systems underactuated by one control. In *Proc. IEEE Control and Decision Conference*, 1996. (to appear).

[41] M. Rathinam and R. Murray. Flatness of nonlinear systems underactuated by two controls. In *Proc. European Control Conference*, 1996. (submitted).

[42] R. Rothfuss, J. Rudolph, and M. Zeitz. Flatness based control of a nonlinear chemical reactor model. *Automatica*, 32(10):1433–1439, 1996.

[43] P. Rouchon. Vibrational control and flatnes of chemical reactors. In *Multiconference on Computational Engineering in Systems Applications*, pages 211–212, 1996.

[44] P. Rouchon, M. Fleiss, J. Lévine, and P. Martin. Flatness, motion planning and trailer systems. In *Proc. IEEE Control and Decision Conference*, pages 2700–2705, 1993.

[45] P. Rouchon, M. Fliess, J. Lévine, and P. Martin. Flatness and motion planning: the car with n trailers. In *Proc. European Control Conference*, pages 1518–1522, 1992.

[46] J. S. Shamma. Robust stability with time-varying structured uncertainty. *IEEE Transactions on Automatic Control*, 39(4):714–724, 1994.

[47] W. M. Sluis. *Absolute Equivalence and its Applications to Control Theory*. PhD thesis, University of Waterloo, Waterloo, Ontario, 1992.