JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 29, No. 2, March–April 2006

Nonlinear Trajectory Generation for Autonomous Vehicles via Parameterized Maneuver Classes

Chris Dever,* Bernard Mettler,† Eric Feron,‡ and Jovan Popović§
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
and
Marc McConley¶
Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts 02139

A technique is presented for creating continuously parameterized classes of feasible system trajectories. These classes, which are useful for higher-level vehicle motion planners, follow directly from a small collection of user-provided example motions. A dynamically feasible trajectory interpolation algorithm generates a continuous family of vehicle maneuvers across a range of boundary conditions while enforcing nonlinear system equations of motion as well as nonlinear equality and inequality constraints. The scheme is particularly useful for describing motions that deviate widely from the range of linearized dynamics and where satisfactory example motions may be found from off-line nonlinear programming solutions or motion capture of human-piloted flight. The interpolation algorithm is computationally efficient, making it a viable method for real-time maneuver synthesis, particularly when used in concert with a vehicle motion planner. Experimental application to a three-degree-of-freedom rotorcraft test bed demonstrates the essential features of system and trajectory modeling, maneuver example selection, maneuver class synthesis, and integration into a hybrid system path planner.

Nomenclature

a_i	=	vehicle model coefficients
$B_{i,k}$	=	ith B-spline basis function of order k
b	=	binary variable
b_i	=	vehicle model coefficients
b_{arv}	=	goal-attainment binary variable
b_{man}	=	maneuver class binary variable
$\boldsymbol{C}_{c,j}$	=	time duration constant for maneuver class j
$\boldsymbol{C}_{s,j}$	=	time duration matrix for maneuver class j
c_i	=	vehicle model coefficients
$c_{i,v}$	=	ith spline coefficient for signal v
c_m	=	maneuver class duration state
D_x	=	differentiation operator with respect to variable x
d_i	=	vehicle model coefficients
e	=	data matching error metric
F	=	nonlinear equation set
f	=	nonlinear program objective function
$\tilde{f}$	=	continuous-time dynamic consistency function
g	=	inequality constraint vector
H	=	planning decision horizon
h	=	equality constraint vector
h_{bc}	=	boundary condition equality constraint vector
h_{bc}^0	=	boundary condition equality constraints invariant with α
h_{em}	=	dynamic consistency equality constraint vector
$\tilde{h}_{\text{em}}$	=	continuous-time dynamic consistency function
i	=	summation index
J	=	planning objective function
J_i	=	active constraint set i
j	=	summation index
k	=	spline order; planning decision step
k_i	=	affine maneuver design constants for planning
l	=	summation index
ℓ	=	linear constraint
$\boldsymbol{M}_{c,j}$	=	state transition vector for maneuver class j
$\boldsymbol{M}_{s,j}$	=	state transition matrix for maneuver class j
M	=	vector of large numbers
m_j	=	binary variable for maneuver class j
N	=	number of data samples
N_f	=	number of final condition maneuver parameters
N_i	=	number of initial condition maneuver parameters
n_v	=	number of spline coefficients for signal v
p	=	finite-dimensional trajectory parametrization
S_e	=	sampling of unit interval for equality constraints
S_g	=	sampling of unit interval for inequality constraints
s	=	arc length in v-space
s_i	=	ith sampling point
T	=	maneuver duration
T_s	=	linear-time invariant (LTI)-mode discretization interval
t	=	time
u	=	direct difference vector
$\hat{u}$	=	projected feasible difference vector
V_{coll}	=	collective voltage
V_{cyc}	=	cyclic voltage
v	=	combined trajectory and maneuver parameter vector; helicopter velocity
W_k	=	weighting matrix at kth sample instance
$\boldsymbol{x}$	=	planning state vector
$\boldsymbol{x}_0$	=	initiation state for fixed maneuvers
$\bar{\boldsymbol{x}}_0, \underline{\boldsymbol{x}}_0$	=	initiation upper, lower bounds for parameterized maneuver classes
$\boldsymbol{x}_F$	=	planning goal state

Presented as Paper 2004-5143 at the AIAA Guidance, Navigation, and Control Conference, Providence, RI, 16–19 August 2004; received 13 September 2004; revision received 21 December 2004; accepted for publication 19 January 2005.

*Ph.D. Candidate, Department of Mechanical Engineering; currently Senior Member, Technical Staff, Charles Stark Draper Laboratory, Inc., Cambridge, MA 02139; cdever@draper.com. Student Member AIAA.

†Research Scientist, Laboratory for Information and Decision Systems, Room 32-D784; bmettler@mit.edu. Associate Member AIAA.

‡Associate Professor of Aeronautics and Astronautics, Laboratory of Information and Decision Systems, Room 32-D724; feron@mit.edu. Associate Fellow AIAA.

§Assistant Professor of Electrical Engineering and Computer Science, Computer Science and Artificial Intelligence Laboratory, Room 32-D534; jovan@lcs.mit.edu.

¶Principal Member, Technical Staff, 555 Technology Square; mcconley@draper.com.

x	=	helicopter position
y_1, y_2	=	affine boundary condition design vectors
Z	=	null space of current constraint derivative matrix
$\mathcal{z}$	=	user-selected system behavior function
z	=	helicopter elevation
α	=	maneuver class/boundary condition parameters
γ	=	dimensionality-reducing map
ζ, ω_n	=	damping ratio and natural frequency for closed-loop LTI-mode model
θ	=	helicopter pitch angle
θ_a	=	trim pitch angle at hover
σ	=	reduced maneuver parameterization (scalar)
τ	=	normalized time
ϕ, ψ	=	boundary condition constraint terms

Subscripts

bc	=	trajectory boundary condition constraints
data	=	related to flight data
e	=	dynamic feasibility constraints
f	=	final condition
goal	=	boundary condition of desired maneuver instance
hov	=	hover state
i	=	initial condition; summation index; sampling index; ith coefficient
$J_{(.)}$	=	current active constraint set
j	=	summation index; jth maneuver class
k	=	index set; data sample index; spline order
l	=	summation index
max	=	upper bound
min	=	lower bound
n	=	number of sampling points
t_f	=	maneuver duration
0	=	initial guess; initial condition; initial maneuver boundary condition set
1	=	first trajectory example
2	=	second trajectory example

Superscripts

T	=	vector transpose
$\bar{y}$	=	equilibrium, or trim, value for quantity y
$+$	=	matrix pseudoinverse
$'$	=	differentiation with respect to normalized time
0	=	equality constraint not involving boundary conditions

I. Introduction

A CENTRAL challenge in the guidance and control of autonomous vehicles is the difficulty of generating, in a computationally efficient manner, system reference trajectories that exploit interesting domains of vehicle nonlinear behavior. A desirable improvement over existing methods is the ability to select and execute families of agile vehicle maneuvers with the same ease as a human pilot flying a manned rotorcraft or fixed-wing airplane. In addition, for motion-planning purposes, it is beneficial to have a concise representation of agile, dynamically feasible maneuver classes with continuously variable boundary conditions. Such classes help reduce the dimensionality of the planning space while increasing the richness of available guidance solutions, allowing greater situational flexibility, and resulting in improved planning performance.

For the first problem of vehicle maneuver design, traditional trajectory generation methods for nonlinear systems formulate a continuous-time optimal control problem, with necessary conditions for optimality following from the calculus of variations.[1] For computational tractability, it is frequently necessary to reduce the mathematics to a finite-dimensional space, often by formulating an approximating nonlinear program (NLP).[2] Typically, the NLP problem statement employs equality constraint functions to dictate boundary conditions and enforce nonlinear model feasibility, whereas inequality constraints impose bounds on states and controls and describe obstacles lying in the vehicle navigation space. The literature presents many useful methods for converting infinite-dimensional variational trajectory design problems to finite-dimensional nonlinear programs.[3–5]

Many modern approaches for generating vehicle motions employ differentially flat, approximately flat, or other output-space and inverse dynamics trajectory parameterizations.[6–13] These formulations provide algorithms with a direct handle on the output signals that best describe vehicle motion while helping to avoid costly forward integrations and sensitivity calculations.

However, these methods, and even those that exploit highly simplified models of vehicle motion,[14] typically resort to an on-line NLP solution procedure. While perhaps reasonable in some specialized cases, NLPs have several key liabilities when considered for real-time implementation: extreme sensitivity to initial solution guesses, no guarantees of convergence to optimal (or even feasible) results, and trajectory overparameterization. This last condition of having too many variables to optimize drives up algorithm dimensionality while allowing superfluous solution options, especially when what is often required is simply a specific instance of a common trajectory type. Thus, a desirable objective is that of bundling sets of useful maneuvers into continuous classes, organizing the flight envelope into something akin to a human pilot's mental model, and simultaneously providing a more direct method to synthesize motions.

Many research fields outside of aerospace confront similar challenges when designing trajectories for complicated nonlinear systems. Several areas employ data capture of real-world motion as a means of seeding the trajectory generation process or even defining concise motion basis sets. For example, in the realm of robotics, it is possible to adapt observed human walking motions for the control of manmade legged robots[15]; other works illustrate a method for machine learning that is "primed" by example nonlinear system demonstrations[16]; and yet further research provides nonlinear system basis functions for capturing, reproducing, and modifying humanoid appendage motions.[17] The practice of motion capture is also common in synthesizing computer animations[18–21] from physical examples, as well as in biological motion research,[22–26] in which the goal is to understand the fundamental bases of motor control.

The concept of using example motions as an aid in trajectory generation and control of complicated nonlinear systems is also emerging in the aerospace community. In some cases, it is possible to use mathematical homotopy methods to transform trajectories for simple models into similar dynamically feasible motions for more complicated nonlinear systems.[27] In another line of pursuit, several very practically oriented works directly study flight data from human expert-piloted aerobatic maneuvers performed on agile small-scale helicopters.[28–30]

This paper combines the notion of working from known motion examples with the rigor of nonlinear programming to create continuously variable maneuver classes for autonomous vehicles. The two essential features behind the scheme are the definition of a parameterized trajectory space and a numerical procedure for feasibly interpolating between elements in that space. The specific characteristics of the maneuver classes follow from user-provided motion examples, which may come from off-line nonlinear programming solutions or real-world motion capture. The feasible space description and the interpolation process center around a low-dimensional set of variable parameters describing a connected set of trajectory boundary conditions. Following user-specified variations in these boundary condition descriptors, the interpolation algorithm employs a continuation method[31,32] adapted from nonlinear parametric programming (NLPP) to trace the corresponding variations in the vehicle feasible reference trajectory (which includes full system state and input information). The result is a straightforward method for combining the rigors of off-line trajectory design with an efficient on-line procedure to generate desired feasible motions.

The parameterized maneuver class scheme provides several key practical benefits. First, it achieves a drastic dimensionality reduction of the nonlinear trajectory generation problem, similar to locally linear embedding methods,[33,34] in effect describing maneuvers in terms of intuitive engineering variations and not only as sets of polynomial coefficients.[35] In addition, the interpolation process does

not include an explicit objective function, and thus allows access to all feasible and useful maneuvers, including trajectories known to be particularly useful for typical vehicle missions yet hard to derive through mathematical optimization. The relaxation of strict optimality eliminates the need for Lagrange multipliers from the continuation procedure, cutting the problem dimensionality essentially in half compared to NLPP. A further benefit of the method is its natural ability to represent vehicle motions as hierarchical elements in existing motion-planning schemes, in particular, those inspired by general hybrid model analysis and control frameworks.[36–38]

The paper first discusses maneuver classes as continuous paths through a parameterized feasible space model of the vehicle flight envelope. This feasible space arises from familiar nonlinear equality and inequality constraint functions that define the characteristics of useful maneuvering trajectories. Introduction of a feasible trajectory interpolation algorithm then allows the user to create continuous maneuver classes easily from a pair of vehicle motion examples. Next, experimental application to a three-degree-of-freedom (DOF) Quanser helicopter illustrates the method in practice, using nonlinear equations of motion, nonlinear constraint functions, and the creation of a specific maneuver class example. Finally, a brief introduction to the mixed integer–linear programming hybrid system framework demonstrates a method of incorporating and planning with parameterized maneuver sets. Application to an intuitive guidance scenario and closed-loop tracking of planner reference solutions shows the complete framework in practice. Conclusions and an appendix giving identified nonlinear helicopter model numerical coefficients then follow.

II. Parameterized Maneuvers

In this paper, a maneuver class is defined as a family of related feasible vehicle trajectories and an associated (small) collection of parameters α describing variable boundary conditions within that family. As seen in Fig. 1, these "maneuver parameters" vary within a useful user-chosen set A, whereas a higher-dimensional dependent collection of conventional "trajectory parameters" $p(\alpha)$ are used to capture the corresponding reference vehicle state and control histories. The parameter vector p is typically a set of spline or basis coefficients used to cast continuous-time-system "behavior" $z(t)$ into a finite-dimensional form. The behavior function $z(t)$ itself is a user-selected set of system state, output, and/or control signals that completely specify the vehicle reference trajectory.

Practically speaking, the parameters α and the set A span useful ranges of vehicle maneuvering capabilities and provide free planning variables for higher-level guidance algorithms. In general, the vector α may include N_i initial condition descriptors $\{\alpha_i^k\}$, N_f final condition descriptors $\{\alpha_f^k\}$, and even a trajectory time length descriptor α_{t_f}:

$$\alpha = \begin{bmatrix} \alpha_i^1 & \cdots & \alpha_i^{N_i} & \alpha_f^1 & \cdots & \alpha_f^{N_f} & \alpha_{t_f} \end{bmatrix}^T \tag{1}$$

A. Trajectory Interpolation

The methods of nonlinear parametric programming (NLPP)[39–41] determine $p(\alpha)$ families when every member of the maneuver class optimizes a common objective function. In contrast, the trajectory interpolation method of this paper relaxes the strict optimality condition of NLPP and instead uses a feasible continuation procedure between pairs of user-provided example trajectories to generate a range of new feasible motions $p(\alpha)$. This process is computationally less expensive than the full NLPP procedure, and thus makes possible fast, on-line feasible trajectory generators for nonlinear systems. Moreover, the framework is extremely general in that the example trajectories may come from any source, including rigorous off-line trajectory optimization or motion capture of human pilot-inspired aerobatic and agile maneuvers. These latter trajectories are extremely useful for autonomous vehicle operation, yet are typically difficult to codify through an objective function. Given the trajectory interpolation mechanism, an engineer can construct a maneuver class by defining a useful set of control parameters α and then finding off-line a finite set of feasible trajectories from which an entire family $p(\alpha)$ can be efficiently computed. Note that trajectory interpolation attempts to retain the qualitative behavior of the user-provided examples by taking the most "direct" available feasible path between them. For instance, time-optimal example trajectories from off-line NLP can serve as prototypes for construction of a class of fast, agile (though likely suboptimal) maneuvers.

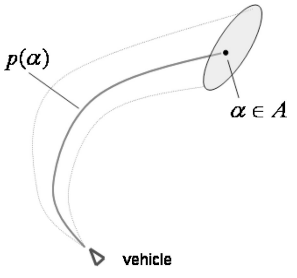

Fig. 1 Autonomous vehicle with a parameterized maneuver family. The low-dimensional vector α specifies the particular maneuver boundary conditions, whereas the dependent function $p(\alpha)$ gives the corresponding feasible trajectory state and control histories.

B. Constraint Functions

In most cases of interest, the maneuver parameters α dictate initial and/or final boundary conditions and therefore occur only in the equality constraint vector. This assumption, along with the discarding of an explicit objective function, leads to the following parameterized trajectory feasible space:

$$h(p, \alpha) = 0, \qquad g(p) \leq 0 \tag{2}$$

It is useful to think of a feasible trajectory as a concatenated vector

$$v = \begin{bmatrix} p \\ \alpha \end{bmatrix} \tag{3}$$

whose components satisfy Eqs. (2). (Here and in the sequel, a semicolon separator ; denotes vertical vector concatenation, so that $v = [p; \alpha] = [p^T \alpha^T]^T$.)

For most autonomous vehicle models, the equality constraint function $h(p)$ breaks down into two general partitions according to

$$h(p, \alpha) = \begin{bmatrix} h_{\text{em}}(p) \\ h_{\text{bc}}(p, \alpha) \end{bmatrix} \tag{4}$$

Here, $h_{\text{em}}(p)$ denotes those constraints not specifying boundary conditions, but instead used to ensure basic dynamic feasibility. These feasibility constraints typically arise in working with nonholonomic path constraints, collocation formulations, quasi-differentially flat vehicle models, and/or mesh-sampled equations of motion. The $h_{\text{bc}}(p, \alpha)$ partition dictates the trajectory initial and final boundary conditions, and thus depends explicitly on α:

$$h_{\text{bc}}(p, \alpha) = \begin{bmatrix} h_{\text{bc}}^0(p) \\ h_i(p, \alpha_i) \\ h_f(p, \alpha_f) \end{bmatrix} \tag{5}$$

(assuming here that $N_i = N_f = 1$ for illustration purposes; more general cases follow similarly). The subpartition $h_{\text{bc}}^0(p)$ denotes boundary constraints not directly depending on α, and $h_i(p, \alpha_i)$ and $h_f(p, \alpha_f)$ are vectors containing control parameters for initial conditions α_i and final conditions α_f, respectively. Typically, h_i has a general form $h_i(p, \alpha_i) = \psi_i(p) - \phi_i(\alpha_i)$, as a difference of some nonlinear vector functions $\psi_i(p)$ and $\phi_i(\alpha_i)$. The helicopter example of the following section will illustrate the role of these functions ψ and ϕ. A similar breakdown applies for $h_f(p, \alpha_f)$.

The inequality constraints in Eqs. (2) assume the general form $g(p) \leq 0$ and capture state and control bounds imposed on vehicle trajectories during maneuvering. Because α typically determines the motion boundary conditions, which are captured in the equality constraint expression, the vector function g depends only the finite-dimensional trajectory parameterization p.

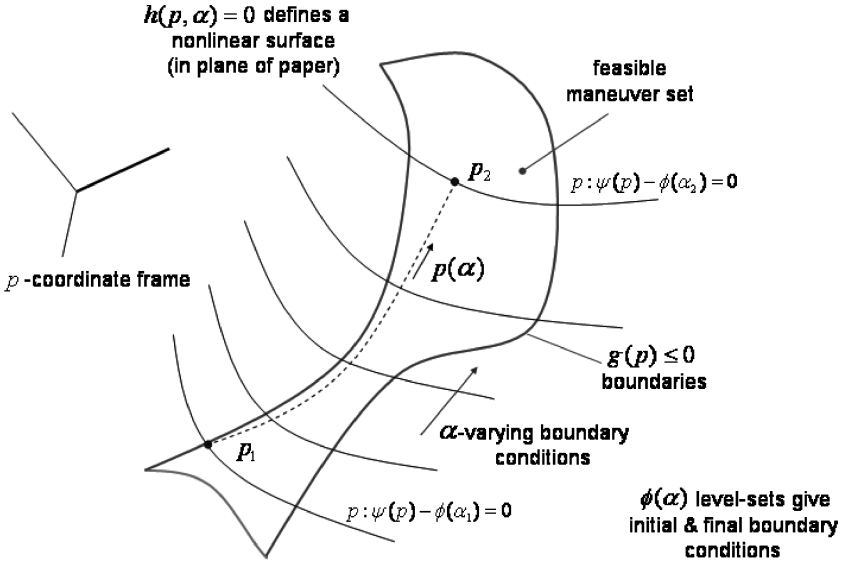

Fig. 2 General feasible maneuver space. Example maneuvers (p_1 and p_2) can come from any source, including off-line nonlinear programming and real-flight motion capture. The parameterized family of feasible maneuvers based on these examples is then given by the feasible space arc $p(\alpha)$.

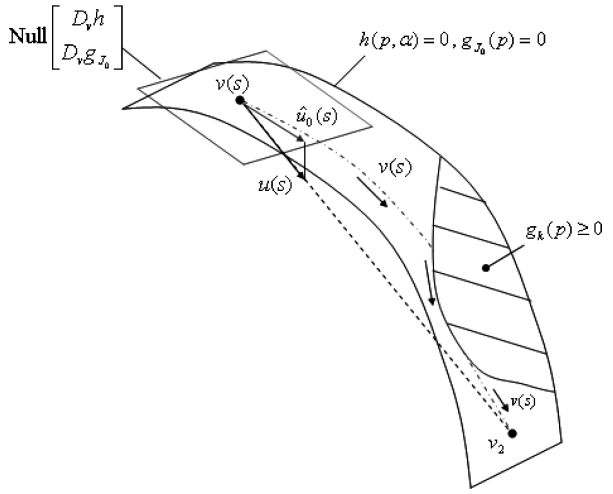

Fig. 3 Projection method for choosing a unique feasible direction. The difference vector $u(s)$ is projected onto the local tangent hyperplane to obtain $\hat{u}(s)$, a tangent vector for the feasible path $v(s)=[p(s);\alpha(s)]$. The path must take into account any inequality constraints occurring on the feasible arc between v_1 and v_2.

C. Algorithm

Now, given the parameterized feasible space of Eqs. (2), the interpolation algorithm takes as its inputs two user-provided feasible motions v_1 and v_2 describing different instances of the same general maneuver type (but with different boundary condition values) and computes as its output a continuous arc $v(s)$ of feasible vehicle trajectories that lie "between" v_1 and v_2. The algorithm computes the parameterized family $p(\alpha)$ seen in Fig. 2, working through a vehicle feasible p space from p_1, with boundary condition α_1, to p_2, with boundary condition α_2. Note that in the figure, and in the following discussion, it is sufficient to treat α as a scalar without loss of generality, since it is typically possible to choose a (further) dimensionality-reducing function $\alpha=\gamma(\sigma)$ mapping a vector-valued α down to a single scalar parameter σ with some σ_1 and σ_2 such that $\alpha_1=\gamma(\sigma_1)$ and $\alpha_2=\gamma(\sigma_2)$.

Leaving specific details to the algorithm pseudo-code presented below, the interpolation uses feasible projections of the numerical difference between vectors v_1 and v_2 to define a unique, continuous arc that is traceable with standard numerical integration methods. Consider Fig. 3, which shows a general trajectory $v(s)$ on the feasible solution path (at arc length s) and shows the projection of the (generally infeasible) difference vector $u(s)=v_2-v(s)$ onto the tangent plane of equality constraints h and locally active inequality constraints $g_{J_0(s)}$. Here, $J_0(s)$ denotes the active inequality constraint index set at a current feasible point $v(s)$. A first-order feasible projection vector $\hat{u}(s)$ is easily obtained using the standard 2-norm least-squares formula[42]

$$\hat{u}(s)=\mathrm{Proj}_{Z(s)}u(s)\equiv Z(s)^{+}u(s)$$

$$=Z(s)[Z(s)^{T}Z(s)]^{-1}Z(s)[v_2-v(s)] \tag{6}$$

where $Z(s)$ is a matrix whose columns form a basis for the nullspace $\mathrm{Null}(D_v[h(p,\alpha);g_{J_0(s)}(p)])$. Note that $\hat{u}(s)$ defines a first-order feasible direction from $v(s)$ toward v_2. Because active inequalities $g_{J_0(s)}$ function as equality constraints, $\mathrm{Null}(D_v[h(p,\alpha);g_{J_0(s)}(p)])$ defines the tangent space at $v(s)$, in which any direction vector moves towards feasible vehicle motion. The projection operation of Eq. (6) simply chooses the direction pointing to the known feasible v_2.

Recalling that α is treated as scalar, it is possible to normalize the projection vector $\hat{u}(s)$ according to $\hat{u}(s)\leftarrow\hat{u}(s)\cdot(\mathrm{d}\alpha/\mathrm{d}s)^{-1}$ so that $s\equiv\alpha-\alpha_1$ at every point along the arc. (This scaling allows one to treat $v(s)$ and $p(\alpha)$ as equivalent functions.)

Given this arc-length normalization, and assuming a mechanism for updating the active constraint set along the solution arc (as seen in Fig. 3), the numerically integrated feasible path gives a solution family $p(\alpha)$ (equivalently $v(s)$) satisfying

$$F_{J_0(s)}(p,\alpha)=0,\qquad g(p)\le 0\qquad\forall\ \alpha_1\le\alpha\le\alpha_2 \tag{7}$$

where the definition

$$F_{J_0(s)}(p,\alpha)\equiv\begin{bmatrix}h(p,\alpha)\\ g_{J_0(s)}(p)\end{bmatrix} \tag{8}$$

indicates that the interpolation algorithm regards equality constraints h and locally active inequality constraints $g_{J_0(s)}$ as a parameterized set of nonlinear equations.

The complete trajectory interpolation algorithm in pseudo-code is formulated as starting from the known feasible $v_1=[p_1;\alpha_1]$ and ending at a maneuver with boundary condition parameter α_{goal} satisfying $\alpha_1\le\alpha_{\mathrm{goal}}\le\alpha_2$. The entire maneuver class results from choosing $\alpha_{\mathrm{goal}}\equiv\alpha_2$. Note that the algorithm begins at each integration step by first assuming that no inequality constraints are active, testing for any active components of g, and then adding them to, or deleting them from $J_0(s)$, depending on the direction of $\hat{u}(s)$ relative to the first-order behavior of $g_{J_0}(s)$. For compactness of notation, $h(v(s))$ denotes $h(p,\alpha)$ in the pseudo-code.

Pseudo-code for Trajectory Interpolation Algorithm:

Inputs: Two feasible maneuvers $v_1 = [p_1; \alpha_1]$ and $v_2 = [p_2; \alpha_2]$ of the same type but satisfying numerically different boundary conditions.

Output: Continuous maneuver family $v(s)$ (equivalently $p(\alpha)$) "between" the examples v_1 and v_2. This family follows by integrating the ordinary differential equation $\dot{v}(s) = \hat{u}(s, v(s))$ with scalar independent variable s along $[0, \alpha_{\text{goal}} - \alpha_1]$ with initial condition $v(0) = v_1$ and $\hat{u}(s, v(s))$ at any point s given by the following computational procedure:

```
u(s) = v2(s) − v(s)                                   direct difference
evaluate Dv h(v(s))                                   local equality derivatives
Z(s) = basis of Null[Dv h(v(s))]                      tangent space basis
û0(s) = Proj_Z(s) u(s)                                project difference
û0(s) ← û0(s) · (dα/ds)^−1                            normalize arc length
J1(s) = {i | gi(p(s)) ≥ 0}                            test for active inequalities
if J1(s) = ∅                                          none active
    û(s, v(s)) = û0(s)
else
    J2(s) = {j ∈ J1(s) | Dv gj(p(s)) · û0(s) ≥ 0}     test inequality behavior
    if J2(s) = ∅                                      none active to first order
        û(s, v(s)) = û0(s)
    else
        Z'(s) = basis of Null{[Dv h(v(s)); Dv g_J2(p(s))]}   follow active inequalities
        û(s) = Proj_Z'(s) u(s)                        reproject difference
        û(s, v(s)) ← û(s) · (dα/ds)^−1                normalize arc length
    end
end
```

It is worth noting that this trajectory interpolation algorithm seeks something akin to a shortest distance path between two known feasible maneuvers, adjusting the direction of the feasible arc whenever inequality constraints become active or inactive. In this manner, the interpolation process resembles NLP active-set methods in its general construction. However, the main difference from NLP methods is that the resulting feasible maneuver class is defined solely by the given example motions, v_1 and v_2, because the interpolation process works between these known feasibles and does not attempt to simultaneously minimize an additional user-provided objective function $f(p)$. Therefore, the method will, in most cases, construct a suboptimal class of maneuvers, when evaluated with most common objective functions. However, because the interpolation process seeks a "short" feasible arc, the family $p(\alpha)$ generally retains the attributes of the example maneuvers, and therefore reasonable engineering judgment in choosing v_1 and v_2 will imply that $p(\alpha)$ both exists and is satisfactory for subsequent motion planning purposes.

The active set methods in the above algorithm are necessary because the feasible arc will generally encounter nonlinear control and/or state constraint functions. Because the interpolation process is based on a vector-valued ordinary differential equation subject to nonlinear constraints, the overall interpolation procedure can be regarded itself as a nonlinear dynamic system. Therefore, if it desired to further speed up the algorithm by eliminating active-set switching logic and instead avoiding constraint boundaries by some fixed margin, one can formulate the interpolation process using barrier function methods. References 43 and 44 give examples of using such techniques to rapidly find feasible solutions to sets of inequality-constrained ordinary differential equations.

D. Maneuver Motion Capture

It has previously been mentioned that in addition to generating example maneuvers for trajectory interpolation by off-line nonlinear programming, it is possible to record pilot-flown vehicle trajectories and then cast them as feasible solutions of Eqs. (2). This process achieves a mathematical transformation $z_{\text{data}}(t_k) \to p_{\text{data}}$, taking a feasible trajectory of the true system and finding a corresponding feasible motion for the system model. It is useful to begin with an initial guess for the feasible point by using a standard, weighted data-matching procedure (such as a polynomial or basis function fitting algorithm), giving

$$p_{\text{data},0} = \arg\min_p \sum_{k=0}^{N} \|z_{\text{data}}(t_k) - z(t_k;\, p)\|_{W_k} \tag{9}$$

Here, the t_k argument indicates a discrete-time sampled data set. A user-selected weighting matrix W_k can be used to highlight particular system states that best describe maneuver characteristics or to perform data time-windowing. Note that because Eq. (9) does not contain any model information, the resulting $p_{\text{data},0}$ estimate is merely an initial guess for a feasible p vector.

To obtain an actual feasible point of Eqs. (2), use $p_{\text{data},0}$ as an initial condition for solving the off-line nonlinear programming problem

$$\min_p e(p) \qquad \text{subject to} \quad \begin{matrix} h(p, \alpha_{\text{data}}) = 0 \\ g(p) \le 0 \end{matrix} \tag{10}$$

where α_{data} gives the maneuver boundary conditions as observed in the flight data. The error minimization objective function $e(p)$ can favor either the initial parameter estimate, as in

$$e(p) = \|p - p_{\text{data},0}\| \tag{11}$$

or attempt to match the flight data directly as closely as possible

$$e(p) = \sum_{k=0}^{N} \|z_{\text{data}}(t_k) - z(t_k;\, p)\|_{W_k} \tag{12}$$

Because nonlinear programming solutions are extremely sensitive to initial guesses, using $p_{\text{data},0}$ from Eq. (9) tends to be far more effective than trying a user-selected initial estimate. Indeed, for highly accurate vehicle models, the optimal solution to Program (10) and $p_{\text{data},0}$ from Eq. (9) can be quite close.

III. Application to a Three-Degree-of-Freedom Helicopter

The three-degree-of-freedom helicopter from Quanser Consulting[45] presents a useful experimental platform for applying the parameterized maneuver framework. The tabletop-mounted vehicle, depicted in Fig. 4, emulates the longitudinal dynamics of full-scale helicopters, with the ability to climb vertically (with elevation z, measured positive downward from level), travel horizontally (with angular position x and velocity v, measured positive clockwise), and pitch about its elevation arm (with angle θ, as shown in the figure). Note that all coordinate variables are angular quantities, because the helicopter is constrained mechanically to an essentially

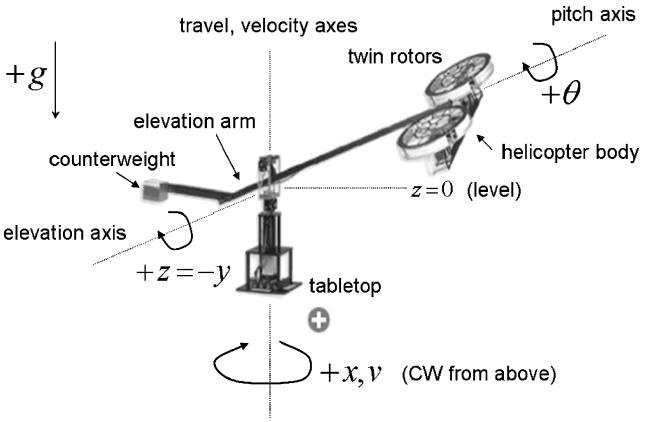

Fig. 4 Three-degree-of-freedom helicopter from Quanser Consulting. The instantaneous vehicle configuration is given by the travel angle x, pitch angle θ, and elevation angle z (downward). (Device photograph by Quanser Consulting; axes and text added by authors.)

spherical flight space. Actuation comes from two fixed-blade pitch propellers driven by dc voltage-controlled motors. The "collective" and "cyclic" inputs control common and differential voltages, respectively, which, in turn, vary the aerodynamic elevation and pitching torques. A joystick input device enables human operation of the helicopter.

A. Modeling and Spline Selection

Physical modeling combined with system identification experiments produced the following nonlinear equations of motion:

$$\dot{x} = v, \qquad \dot{v} = -a_1 v - a_2 V_{\text{coll}}^2 \sin(\theta - \theta_a)$$

$$\ddot{\theta} = -b_1\dot{\theta} - b_2 \sin(\theta) + b_0 + b_3 v|v| + b_4 V_{\text{coll}} V_{\text{cyc}}$$

$$\ddot{z} = -d_1\dot{z} + d_2 \cos(z) - d_3 \sin(z) - d_5 v^2 - d_4 V_{\text{coll}}^2 \cos(\theta) \quad (13)$$

As seen in Eqs. (13), the three-DOF helicopter shares many common nonlinearities with full-scale rotorcraft, including trigonometric terms from thrust vector tilting and vehicle kinematics, a speed-damping effect containing an absolute value expression, and products of thrust expressions (i.e., collective input) with angular quantities. In addition, the twin-rotor configuration leads to quadratic collective voltage terms and a cyclic-collective input product term. Note that the helicopter accelerates under the action of both cyclic control (which alters pitch θ and therefore the horizontal, or driving, component of thrust) and collective control (which directly governs the overall thrust magnitude, represented by the V_{coll}^2 term). Over the speed range of interest, the velocity dynamics exhibited a near-linear speed-damping effect, whereas the pitch dynamics strongly followed a nonlinear quadratic $v|v|$ damping term. (The Appendix gives numerical coefficient values for these equations and briefly describes the vehicle-modeling process.)

For trajectory-generation purposes, define the system behavior function $\boldsymbol{z}$ as the 3-vector $\boldsymbol{z}(t) = [v(t), z(t), \theta(t)]^T$ and then cast each degree of freedom in a finite-dimensional space, employing a B-spline[46] basis on a normalized time interval according to

$$v(\tau) = \sum_{i=1}^{n_v} c_{i,v} B_{i,k}(\tau, i : i + k)$$

$$z(\tau) = \sum_{j=1}^{n_z} c_{j,z} B_{j,k}(\tau, j : j + k)$$

$$\theta(\tau) = \sum_{l=1}^{n_\theta} c_{l,\theta} B_{l,k}(\tau, l : l + k) \quad (14)$$

Here, the splines are of sixth order ($k = 6$), allowing a sufficient number of smooth derivatives to be computed during model inversion and control input calculation. Additionally, each signal basis has the same uniform knot sequence: $S_v = S_z = S_\theta = \{0^6, 1/10, 2/10, \ldots, 9/10, 1^6\}$. The exponents 6 imply knots of multiplicity 6 at normalized time values 0 and 1, allowing the state signals to take nonzero values at the start and end of maneuver segments. Together, the spline order and knot selection imply $n_v = n_z = n_\theta = 15$, so that 15 coefficients completely describe each helicopter degree of freedom. This type of signal parameterization has previously been proven useful for autonomous vehicle trajectory design and is presented with numerous examples in existing references.[6,9,10]

The normalized time scale $\tau \in [0, 1]$ is related to the actual time interval $t \in [0, T]$ according to $\tau = t/T$, where T is the trajectory time length. Naturally, derivatives of a quantity $y(\tau)$ on the normalized scale (denoted with primes) and derivates of the same quantity $y(t)$ expressed on the actual time scale (denoted with overdots) relate according to $\dot{y}(t) = (1/T)y'(\tau)$, $\ddot{y}(t) = (1/T^2)y''(\tau)$, and so forth. Given the spline coefficients in Eqs. (14) and the quantity T, a useful output-space finite trajectory parameter p is

$$p \equiv \left[\{c_{i,v}\}_{i=1}^{n_v}, \{c_{j,z}\}_{j=1}^{n_z}, \{c_{l,\theta}\}_{l=1}^{n_\theta}, T\right]^T \quad (15)$$

Working with a normalized time scale allows T to become a free parameter and will enable direct control over trajectory duration within maneuver classes. It is useful to adopt the expanded notation $v(\tau) = v(\tau; p)$, $z(\tau) = z(\tau; p)$, $\theta(\tau) = \theta(\tau; p)$ as a reminder that these time domain signals depend explicitly on the components of p.

Given the above choices for $\boldsymbol{z}$ and p, it is a simple matter to invert the system model of Eqs. (13) and solve for the two input voltage signals:

$$V_{\text{coll}}(\tau; p) = \sqrt{\frac{(1/T^2)z''(\tau; p) + (1/T)z'(\tau; p) - d_2\cos[z(\tau; p)] + d_3\sin[z(\tau; p)] + d_5 v^2(\tau; p)}{-d_4\cos[\theta(\tau; p)]}} \quad (16)$$

$$V_{\text{cyc}}(\tau; p) = \frac{(1/T^2)\theta''(\tau; p) + b_1(1/T)\theta'(\tau; p) + b_2\sin[\theta(\tau; p)] - b_0 - b_3(1/T^2)v(\tau; p)|v(\tau; p)|}{b_4 V_{\text{coll}}(\tau; p)} \quad (17)$$

Note that both control inputs are nonlinear functions of the components of p.

B. Dynamic Feasibility and Boundary Conditions

Because the chosen system behavior function $\boldsymbol{z}(t)$ contains more system degrees of freedom (three states) than there are input channels (two voltages), it is necessary to impose equality constraints to enforce dynamic consistency. Note that such constraints will define the $h_{\text{em}}(p)$ partition of Eq. (4). First, find a differential relation $\tilde{h}_{\text{em}}$ that contains all three component states of the behavior function $\boldsymbol{z}(t)$, that is, some

$$\tilde{h}_{\text{em}}(\tau; p) = \tilde{f}(v(\tau; p), z(\tau; p), \theta(\tau; p), T) = 0, \quad \forall \quad \tau \in [0, 1] \quad (18)$$

Such a relation follows easily from elimination of the V_{coll}^2 term between lines 2 and 4 of Eqs. (13), giving

$$\tilde{f}(v, z, \theta, T) = d_4[(1/T)v' + a_1 v]\cos(\theta) - a_2\left[(1/T^2)z'' + d_1 z' - d_2\cos(z) + d_3\sin(z) + d_5 v^2\right] \times \sin(\theta - \theta_a) \quad (19)$$

To obtain an approximating finite-dimensional vector $h_{\text{em}}(p)$, sample the continuous time constraint of Eqs. (18) and (19) along a finite point set to obtain

$$h_{\text{em}}(p) \equiv \begin{bmatrix} \tilde{h}_{\text{em}}(\tau_1;\, p) \\ \vdots \\ \tilde{h}_{\text{em}}(\tau_n;\, p) \end{bmatrix} \tag{20}$$

where the individual points τ_i are elements of a finite sampling S_e of the unit interval: $S_e = \{\tau_1, \ldots, \tau_n\} \subset [0, 1]$. For the three-DOF helicopter, a useful sampling is

$$S_e = \{0, 1/30, 1/15, 2/15, \ldots, 14/15, 29/30, 59/60\} \tag{21}$$

The other partition of the general equality constraint vector $h(p)$ is the boundary condition constraint $h_{\text{bc}}(p, \alpha)$. As the presence of the α quantity shows, this vector is the main driver behind the creation of continuous maneuver classes. Without selecting a specific α yet, consider first the set of quantities necessary to define an equilibrium, or trim state at the beginning and end of a maneuver segment. Analysis of Eqs. (13) indicates that a steady velocity–elevation pair $(\bar{v}, \bar{z})$ is sufficient to determine trim values of pitch, collective voltage, and cyclic voltage. For instance, setting all time derivatives to zero in Eqs. (13) gives the steady pitch relation

$$a_2 \sin(\bar{\theta} - \theta_a)\left[d_2 \cos(\bar{z}) - d_3 \sin(\bar{z}) - d_5 \bar{v}^2\right] + a_1 d_4 \bar{v} \cos(\bar{\theta}) = 0 \tag{22}$$

and steady input relations

$$\bar{V}_{\text{coll}} = \sqrt{\frac{d_2 \cos(\bar{z}) - d_3 \sin(\bar{z}) - d_5 \bar{v}^2}{d_4 \cos(\bar{\theta})}}$$

$$\bar{V}_{\text{cyc}} = \frac{b_2 \sin(\bar{\theta}) - b_0 - b_3 \bar{v}|\bar{v}|}{b_4 \bar{V}_{\text{coll}}} \tag{23}$$

Solution for exact trim values in Eqs. (22) and (23) requires a numerical search procedure, similar to thrust-inflow iterations seen in full-scale helicopter models.[47] However, a fortunate consequence of the trajectory interpolation algorithm is that only first-order equality constraint derivative information is required, requiring only implicit differentiation of the above relations.

The overall trim boundary condition constraint function requires the attainment of initial and final equilibrium values for velocity, elevation, pitch, collective, and cyclic. In addition, it is required that elevation and pitch signals have first-order time derivatives equal to zero at the trajectory endpoints. (A corresponding zero-derivative condition on velocity proved redundant, given the trim state boundary value constraints on the two inputs.) Concatenation of normalized time expressions for all these conditions gives the desired boundary condition equality constraint vector (denoted here temporarily by $\hat{h}_{\text{bc}}$ until the α argument is introduced):

$$\hat{h}_{\text{bc}}(p) \equiv \begin{bmatrix} v(0;\, p) - \bar{v}_i \\ v(1;\, p) - \bar{v}_f \\ z(0;\, p) - \bar{z}_i \\ z(1;\, p) - \bar{z}_f \\ \theta(0;\, p) - \bar{\theta}_i \\ \theta(1;\, p) - \bar{\theta}_f \\ V_{\text{coll}}(0;\, p) - \bar{V}_{\text{coll},i} \\ V_{\text{coll}}(1;\, p) - \bar{V}_{\text{coll},f} \\ V_{\text{cyc}}(0;\, p) - \bar{V}_{\text{cyc},i} \\ V_{\text{cyc}}(1;\, p) - \bar{V}_{\text{cyc},f} \\ (1/T)z'(0;\, p) - 0 \\ (1/T)z'(1;\, p) - 0 \\ (1/T)\theta'(0;\, p) - 0 \\ (1/T)\theta'(1;\, p) - 0 \end{bmatrix} = 0 \tag{24}$$

C. State and Control Bounds

Last, inequality constraint functions are needed to keep helicopter trajectories within a reasonable flight envelope and control signals within physical limits. For the three-DOF helicopter, it is useful to bound elevation to prevent collisions with the supporting surface and the upper mechanical restraint limits; the pitch angle must stay away from extreme $\theta = \pm 90^\circ$ values which induce a singularity in Eqs. (16) and (23). Additionally, it is important to prevent the control inputs from exceeding reasonable ranges, especially when working with agile maneuvers that require large control efforts.

These state and input constraints appear in continuous time as

$$\begin{aligned} z_{\min} &\le z(\tau;\, p) \le z_{\max} && \forall \quad \tau \in [0, 1] \\ \theta_{\min} &\le \theta(\tau;\, p) \le \theta_{\max} && \forall \quad \tau \in [0, 1] \\ V_{\text{coll},\min} &\le V_{\text{coll}}(\tau;\, p) \le V_{\text{coll},\max} && \forall \quad \tau \in [0, 1] \\ V_{\text{cyc},\min} &\le V_{\text{cyc}}(\tau;\, p) \le V_{\text{cyc},\max} && \forall \quad \tau \in [0, 1] \end{aligned} \tag{25}$$

Similarly to the continuous-time equality constraint of Eq. (18), it is necessary to sample Inequalities (25) along some sampling of the unit interval [0, 1] to obtain a finite-dimensional constraint vector. Therefore, employ the following sampled version of the preceding inequalities:

$$g(p) \equiv \begin{bmatrix} z_{\min} - z^{S_g}(p) \\ z^{S_g}(p) - z_{\max} \\ \theta_{\min} - \theta^{S_g}(p) \\ \theta^{S_g}(p) - \theta_{\max} \\ V_{\text{coll},\min} - V_{\text{coll}}^{S_g}(p) \\ V_{\text{coll}}^{S_g}(p) - V_{\text{coll},\max} \\ V_{\text{cyc},\min} - V_{\text{cyc}}^{S_g}(p) \\ V_{\text{cyc}}^{S_g}(p) - V_{\text{cyc},\max} \end{bmatrix} \le 0 \tag{26}$$

where, for a generic parameterized unit time domain signal $y(\tau;\, p)$, the symbol $y^{S_g}(p)$ has the definition $y^{S_g}(p) \equiv [y(s_1;\, p), \ldots, y(s_m;\, p)]^T$, where the s_i are the sampling points $S_g = \{s_1, \ldots, s_m\} = \{0, 1/20, \ldots, 1\} \subset [0, 1]$.

D. Example: Bounded-Control Quick-Stop Maneuver Class

Given the helicopter boundary constraint expression of Eq. (24), it is a simple matter to create a parameterized class of quick-stop maneuvers. Here, all trajectories end at a steady hover state with $\bar{v}_i = \bar{z}_i = 0$. However, let the initial trim velocity $\bar{v}_i$ be variable; that is, assign $\alpha \equiv \bar{v}_i$. For simplicity, assume that $\bar{z}_i = 0$ is fixed over the maneuver class.

Given this specific choice of α, introduce it into all boundary condition rows of Eq. (24) that depend on the initial helicopter velocity. Recalling from Eqs. (22) and (23) that the initial trim pitch angle and input voltages vary with $\bar{v}_i$, insert the boundary parameter α into the corresponding rows of Eq. (24) to obtain

$$h_{\text{bc}}(p, \alpha) = \begin{bmatrix} v(0;\, p) - \alpha \\ v(1;\, p) - 0 \\ z(0;\, p) - 0 \\ z(1;\, p) - 0 \\ \theta(0;\, p) - \bar{\theta}_i(\alpha) \\ \theta(1;\, p) - \bar{\theta}_{\text{hov}} \\ V_{\text{coll}}(0;\, p) - \bar{V}_{\text{coll},i}(\alpha) \\ V_{\text{coll}}(1;\, p) - \bar{V}_{\text{coll,hov}} \\ V_{\text{cyc}}(0;\, p) - \bar{V}_{\text{cyc},i}(\alpha) \\ V_{\text{cyc}}(1;\, p) - \bar{V}_{\text{cyc,hov}} \\ (1/T)z'(0;\, p) - 0 \\ (1/T)z'(1;\, p) - 0 \\ (1/T)\theta'(0;\, p) - 0 \\ (1/T)\theta'(1;\, p) - 0 \end{bmatrix} = 0 \tag{27}$$

Note that all boundary trim constraints depend on the spline parameterization p (which includes T) whereas only four rows depend on α. While carrying out the interpolation algorithm, it is necessary to compute the derivative matrix $D_v h(p, \alpha)$, where $h(p, \alpha)$ has the decomposition given in Eq. (4). Therefore this derivative matrix has the following block structure:

$$D_v h(p, \alpha) = \begin{bmatrix} D_p h_{\text{em}}(p) & 0 \\ D_p h_{\text{bc}}(p, \alpha) & D_\alpha h_{\text{bc}}(p, \alpha) \end{bmatrix} \tag{28}$$

where comparison with Eq. (27) reveals that the $D_\alpha h_{\text{bc}}(p, \alpha)$ is a column vector with only four nonzero elements; these elements are the mathematical drivers for interpolation of this maneuver class.

An off-line feasible point NLP algorithm provides two example quick-stop maneuvers. The first has $\alpha_1 = -10$ deg/s, whereas the second corresponds to $\alpha_2 = -50$ deg/s, thus covering a wide range of initial velocity conditions. (The helicopter dynamics are essentially identical in both travel directions; the velocity sign simply indicates the specific direction.) Applying the interpolation algorithm across the entire $\alpha_1 \le \alpha \le \alpha_2$ range produces the feasible velocity, elevation, and pitch profile curves of Figs. 5, 6, and 7, respectively. Note that each signal exhibits smooth variations over the range of initial velocity (denoted by v_0), making continuous transitions between the known feasible motions at either extreme. In particular, the pitch profiles exhibit an intuitive large-angle reverse flare to reduce helicopter speed, with the flare duration (and thus total deceleration) growing with increasing initial speed magnitude.

Figures 8 and 9 display the corresponding control behavior over the maneuver class, with upper and lower bounds imposed (and required) for both signals. (These two figures are plotted vs normalized time τ for visual clarity.) Contact between the collective profile set and the upper constraint boundary is visible near the beginning and end of the control sequence for initial maneuver speeds around 20 deg/s and near the middle of the control sequence for a majority

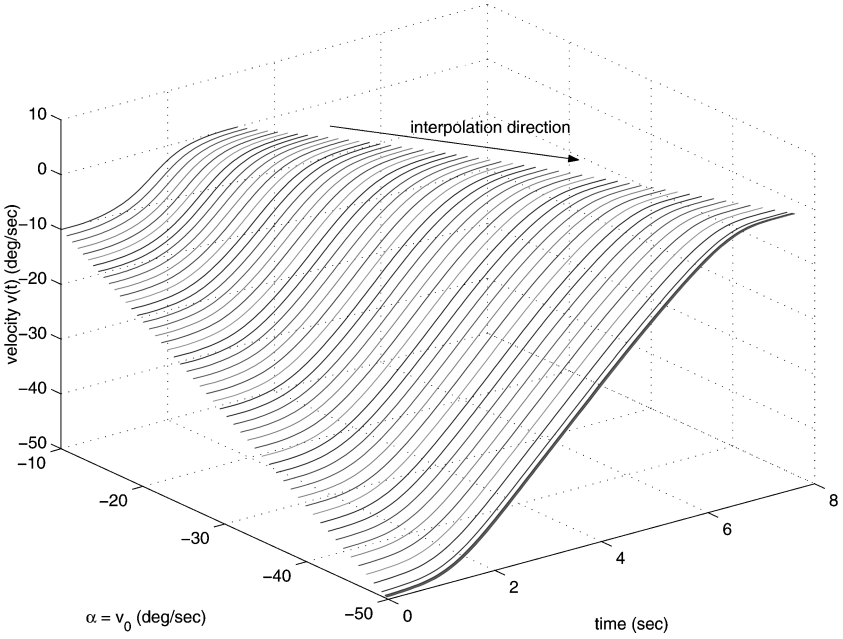

Fig. 5 Velocity profiles for quick-stop maneuver class.

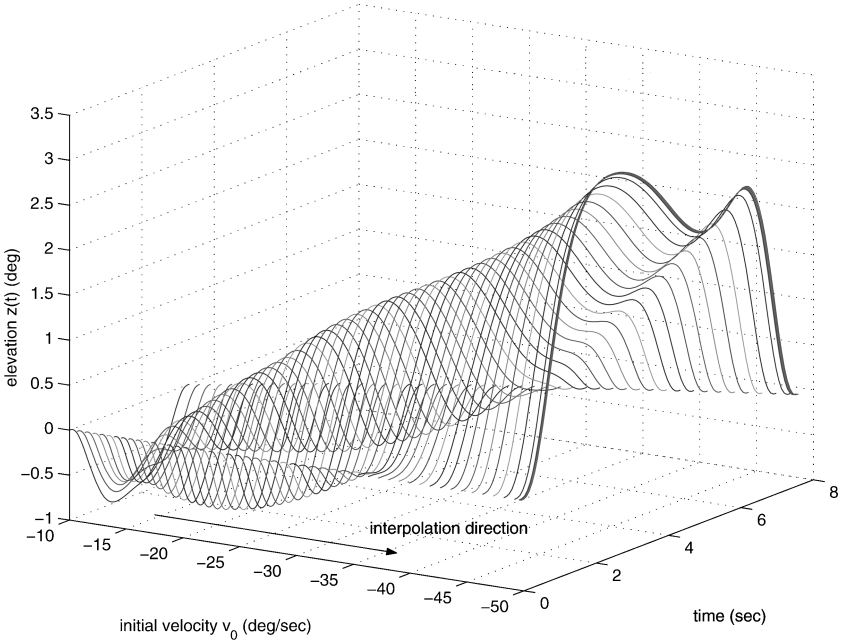

Fig. 6 Elevation profiles for quick-stop maneuver class.

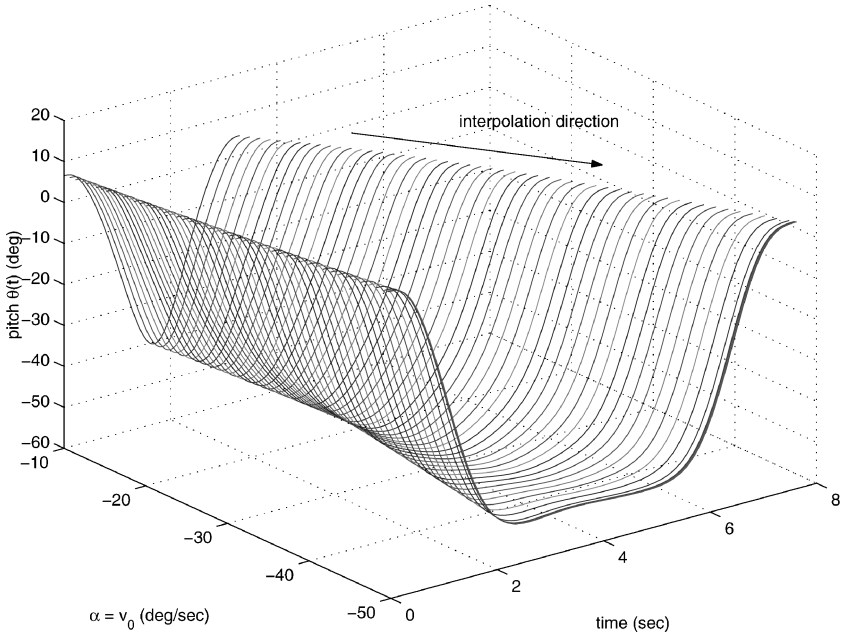

Fig. 7 Pitch profiles for quick-stop maneuver class.

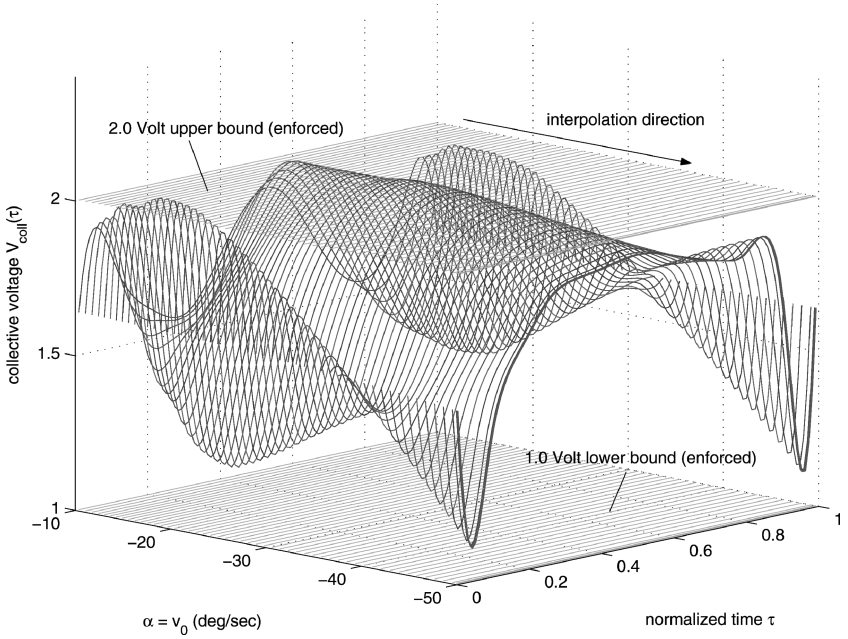

Fig. 8 Collective voltage profiles for quick-stop maneuver class.

of initial speeds. The cyclic profiles tend to encounter the control limits toward the beginning and end of their respective profiles. The nonlinearity of Eqs. (16) and (17) is evident, especially in the range $-10 \leq \alpha \leq -30$ deg/s. However, despite the nonlinearity, the control strategy is essentially the same, reflecting the intended property of retaining the maneuver "style" during the interpolation process. The cyclic input displays a brief spike at the beginning and end of each quick-stop, essentially pitching the helicopter initially backward and then restoring it to a steady hover trim value as seen in Fig. 7. The collective input profiles assist somewhat in this motion [recall the product input term in line 3 of Eqs. (13)], but more importantly display a broad hump for most of the maneuver, applying a large reverse thrust to decelerate the helicopter.

The bounding example maneuvers used to generate this class were slightly suboptimal, given the control bounds, when compared to a minimum-maneuver time objective function. Overall, each member of the interpolated maneuver class stays within a 1-s time margin of a corresponding optimal flare-stop maneuver.

E. Coupled Boundary Conditions and Further Dimensionality Reduction

As mentioned in the earlier discussion of trajectory interpolation techniques, it is often useful to map a general boundary condition vector α down to a single scalar variable of integration σ by employing a user-selected function γ such that $\alpha = \gamma(\sigma)$. Such a map reduces the interpolation procedure to a single scalar numerical integration process, avoiding the need to integrate in the full multicomponent vector space of α.

Proper choice of σ and γ has the added benefit of enforcing exact, coupled variations in boundary conditions over a given maneuver class. For example, in the above quick-stop maneuver class, the net position change resulting from vehicle deceleration will be some general nonlinear function of the initial speed, determined by the vehicle dynamics and the selected control bounds. However, as will be seen next in the context of hybrid motion planning, it is often desirable to enforce exact, coupled affine relationships between initial and final boundary conditions.

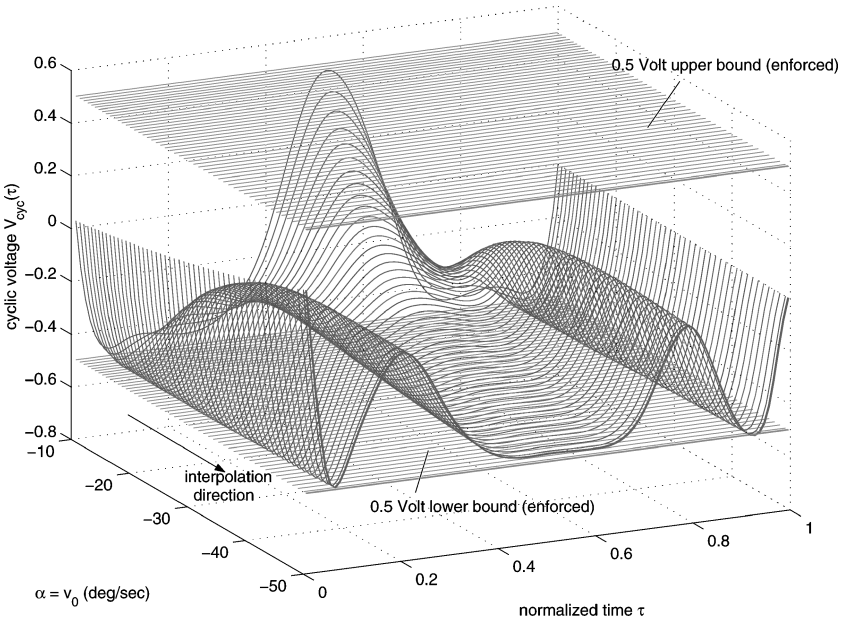

Fig. 9 Cyclic voltage profiles for quick-stop maneuver class.

Such an explicit relation can easily be accommodated by choosing σ equal to some initial boundary condition parameter *and* then expressing the general α vector (containing all initial *and* final boundary condition parameters) as $\alpha = y_1\sigma + y_2$ for some user-selected constant vectors y_1 and y_2. In the case of the quick-stop maneuver, one could choose an α vector as $\alpha = [\bar{v}_i, \Delta x]^T$ and then enforce a linear correlation between initial velocity and net position change by choosing $\sigma \equiv \bar{v}_i$ with $\alpha = y_1\sigma + y_2$, where $y_1 = [1, c_1]^T$ and $y_2 = [0, c_2]^T$, for some constants c_1 and c_2.

IV. Motion Planning with Parameterized Maneuver Classes

Vehicle motion planners often employ hybrid system representations to reduce the system state space to only the essential variables, thus simplifying the planning problem's search space. Such hybrid vehicle models are based on a principle of hierarchical decomposition, quantifying system dynamics to enable planning in a practical subspace of useful vehicle motions.[38]

For hybrid models of the maneuver automaton (MA) form,[48,49] which combine a finite collection of trim trajectories and fixed maneuver elements, the state-space is small enough to allow for policy-based guidance systems based on a value function precomputed off-line. In more recent hybrid linear time-invariant maneuver automaton (LTI-MA) models, in which a finite number of linearly controlled modes (termed LTI-modes) replace the trim trajectory elements, motion planning typically requires solving an on-line optimization problem. Mixed integer–linear programming (MILP) optimization techniques have been highly effective for formulating and solving such trajectory planning problems, because they enable one to explicitly account for hybrid structures and allow easy incorporation of other useful problem elements.[50–53]

Given the availability of parameterized maneuver classes, it is now possible to generalize the hybrid LTI-MA model's representation of maneuvering elements. The following section presents the extensions necessary to include parameterized maneuver sets in the MILP formulation, using few additional constraint functions. This extension increases the range of trajectories captured by the hybrid model, thus allowing better satisfaction of planning objectives.

A. MILP Trajectory Optimization

The basic elements of a mixed integer–linear program for a free continuous variable x and binary integer variable b include a (piecewise) linear objective function $J(x)$, linear constraint functions $\ell(x)$, and appropriately sized vectors M of large numbers. The typical optimization problem statement has the form

$$\min_{x,b} J(x)$$

subject to:

$$\ell_1(x) \le Mb$$

$$\text{AND} \quad \ell_2(x) \le M(1-b)$$

$$b \in \{0, 1\} \tag{29}$$

In motion-planning applications, the objective function defines the goal of a vehicle guidance problem. The constraints and binary variables model the specific system and problem structure, in particular, modeling closed-loop vehicle dynamics and enforcing numerous constraints, such as flight envelope restrictions and the presence of obstacle boundaries.

Of particular interest is the ability to consider mode-switching logic and other conditional expressions. For example, in program (29), when $b = 0$, the constraint $\ell_1(x) \le 0$ must be satisfied, whereas $\ell_2(x) \le M(1-b) = M$ effectively relaxes the $\ell_2(x)$ constraint (since M is selected to be sufficiently large relative to other problem data). Note that the constraint activations are reversed when $b = 1$. Because b can take on only the binary values 0 or 1, exactly one of the inequality constraints must be satisfied.

In the basic LTI-MA MILP formulation,[50–53] the inequality constraints include the LTI-mode state update equations (one for each mode) and state transition relations describing the effect of maneuvering actions (one for each maneuver), as well as maneuver initiation requirements (e.g., vehicle speed within some range). Note that at every point in time, either one of the LTI-modes *or* one of the maneuvering actions must be active. MILP binary variables are critical for enforcing this either/or condition. The reader may consult the existing literature[50–53] for the detailed motivation and treatment of the basic LTI-MA MILP formulation.

B. MILP Trajectory Optimization with Parameterized Maneuver Classes

To include parameterized maneuver classes in the standard MILP formulation,[50–53] their corresponding maneuvering state transition equations must be in the form of a linear constraint $\ell(x)$. The minimum-time formulation also requires a single affine expression to describe the maneuver time duration across the entire maneuver class.

To this effect, one uses affine state transition inequalities that capture the variation in maneuvering effect across the entire maneuver class. Considering the jth maneuver class, with a corresponding maneuver binary variable $b_{\text{man},j}[k]$, the state transition inequalities take the form

$$\begin{aligned} \boldsymbol{x}[k+1] - \boldsymbol{M}_{s,j}\boldsymbol{x}[k] - \boldsymbol{M}_{c,j} - \boldsymbol{x}[k] &\le M(1 - b_{\text{man},j}[k]) \\ -\boldsymbol{x}[k+1] + \boldsymbol{M}_{s,j}\boldsymbol{x}[k] + \boldsymbol{M}_{c,j} + \boldsymbol{x}[k] &\le M(1 - b_{\text{man},j}[k]) \end{aligned} \quad (30)$$

The matrix $\boldsymbol{M}_{s,j}$ represents the state-dependent component of the transformation while $\boldsymbol{M}_{c,j}$ gives the constant component. Note the members of this maneuver class can now be designed for execution over a continuous range of initial planning states, as enforced by the constraint pair

$$\begin{aligned} \boldsymbol{x}[k] - \bar{\boldsymbol{x}}_0 &\le M(1 - b_{\text{man}}[k]) \\ -\boldsymbol{x}[k] + \underline{\boldsymbol{x}}_0 &\le M(1 - b_{\text{man}}[k]) \end{aligned} \quad (31)$$

in contrast to single, fixed-maneuver initial condition in existing LTI-MA-based planners.[52,53]

Because it is difficult to include the full cost update expression directly in the objective function, we define a maneuvering cost state $c_m[k]$, which is set as follows when a member of the jth maneuver class is executed:

$$\begin{aligned} c_m[k] - \boldsymbol{C}_{s,j}\boldsymbol{x}[k] - \boldsymbol{C}_{c,j} &\le M(1 - b_{\text{man},j}[k]) \\ -c_m[k] + \boldsymbol{C}_{s,j}\boldsymbol{x}[k] + \boldsymbol{C}_{c,j} &\le M(1 - b_{\text{man},j}[k]) \end{aligned} \quad (32)$$

Similar to inequalities (30), the state-dependent and constant coefficients are $\boldsymbol{C}_{s,j}$ and $\boldsymbol{C}_{c,j}$, respectively. In LTI-mode, the maneuver cost term is naturally set to zero:

$$\begin{aligned} c_m[k] &\le M \sum_j b_{\text{man},j}[k] \\ -c_m[k] &\le M \sum_j b_{\text{man},j}[k] \end{aligned} \quad (33)$$

This cost state then enters the planning objective function as follows:

$$J = \sum_{k=0}^{H} \left[b_{\text{arv}}[k] k T_s + c_m[k] - \sum_j b_{\text{man},j}[k] T_s \right] \quad (34)$$

C. Obtaining/Satisfying Affine State-Transition and Cost-Update Equations

Both the affine state-transition and affine duration requirements of the MILP planner introduce additional constraints on a maneuver class. These requirements can be imposed by simply applying an affine map $\alpha = \gamma(\sigma)$ as previously discussed and then including this relation in the appropriate α-dependent rows of $h_{\text{bc}}(p, \alpha)$. The next section gives a specific example of this method. (Note that because the MILP planner considers the maneuver class to have variable initial conditions, α must include at least one component corresponding to a vehicle initial condition.) By first generating a class of time-optimal maneuvers off-line, it is possible to choose the affine state and duration functions to minimize the overall optimality gap, thus helping to maintain agile vehicle performance characteristics within the hybrid planning framework.

V. Motion-Planning Example

This section describes a simple but general one-dimensional example of using parameterized maneuver sets for practical helicopter guidance problems, exploiting the capabilities of trajectory classes in the MILP hybrid system framework. References 54, 55 present additional planning examples, including motions involving the full two-dimensional planning space of the Quanser helicopter.

The scenario requires the helicopter to start at a forward speed of 50 deg/s and then return, in minimum time, to a position located 500 deg behind it. The elevation z is restricted to a level flight condition. The MILP planner is used to find the minimum-time solution, taking advantage of a velocity control mode and two parameterized maneuvers classes: a direction-reversal and a flaring quick-stop.

For this example, the nonmaneuver linearly controlled mode of the LTI-MA framework[52,53] models the closed-loop system in a velocity command-tracking loop. Therefore, it is useful to select a three-dimensional planning state $\boldsymbol{x}[k] \equiv [\dot{v}[k], v[k], x[k]]^T$. Given this form of $\boldsymbol{x}[k]$, the initial state is $\boldsymbol{x}_0 = [0, -50, 0]^T$, corresponding to an initial steady cruise velocity of -50 deg/s with a reference planning initial position of 0 deg. The hover waypoint destination, located 500 deg behind the helicopter, gives a goal state of $\boldsymbol{x}_F = [0, 0, +500]^T$.

The helicopter motion planner has access to two parameterized maneuver families. (A general planner would of course include many maneuver classes, but only those useful to this particular guidance problem are mentioned here.) The first is a direction-reversal maneuver, which quickly decelerates the helicopter, reverses direction, rapidly accelerates, and ultimately returns to a steady cruise state of the same speed, but with opposite sign. Symbolically, the resulting velocity change corresponds to $\bar{v}_f = -\bar{v}_i$. Further, the maneuver class can be designed so that $x_f = x_i$ exactly, so that no *net* position change occurs from the reversal. In setting up this maneuver class for trajectory interpolation, it is useful to select the control parameters $\alpha \equiv [\bar{v}_i, \bar{v}_f, T]^T$ and apply a dimensionality-reducing map $\alpha = \gamma(\sigma)$. The scalar σ is chosen identically equal to initial velocity $\sigma \equiv \bar{v}_i$, so that the map γ has the specific form $\gamma = [\sigma, -\sigma, k_1\sigma + k_2]^T$, where k_1 and k_2 are constants selected to create a dynamically feasible reversal class with time duration being an affine function of $\sigma = \bar{v}_i$. [Recall the requirements of inequalities (32).] The speed reversal and zero net displacement fit easily within the affine state transition requirements of inequalities (30). Finally, choose maneuver initiation bounds [recall inequalities (31)] requiring the helicopter initial speed to be in the range $-65 \le \bar{v}_i \le -5$ deg/s. Creation of the maneuver class follows easily by generating example maneuvers at either extreme of the initial speed range, inserting the components of α into the suitable rows of the boundary condition constraint vector $h_{\text{bc}}(p, \alpha)$, and then applying the interpolation algorithm using σ as the independent variable.

Design of a quick-stop maneuver follows similarly, closely resembling the example of Figs. 5 through 9, but this time starting with a positive velocity. Here, choose $\alpha \equiv [\bar{v}_i, x_f, T]^T$, noting that $\bar{v}_f \equiv 0$ since each member of the maneuver class ends at a steady hover state. To allow for a single-independent-variable interpolation procedure, apply a dimensionality-reducing map $\alpha = \gamma(\sigma)$ with $\sigma \equiv \bar{v}_i$ and $\gamma = [\sigma, k_3\sigma + k_4, k_5\sigma + k_6]^T$. Again, the k_i coefficients are selected so that position change and time duration are feasible affine functions of the maneuver initial speed, which may vary over the range $10 \le \bar{v}_i \le 60$ deg/s.

Optimization of the minimum-time guidance problem with the preceding LTI-mode and two maneuver classes gives the reference position and velocity solutions depicted in Figs. 10 and 11,

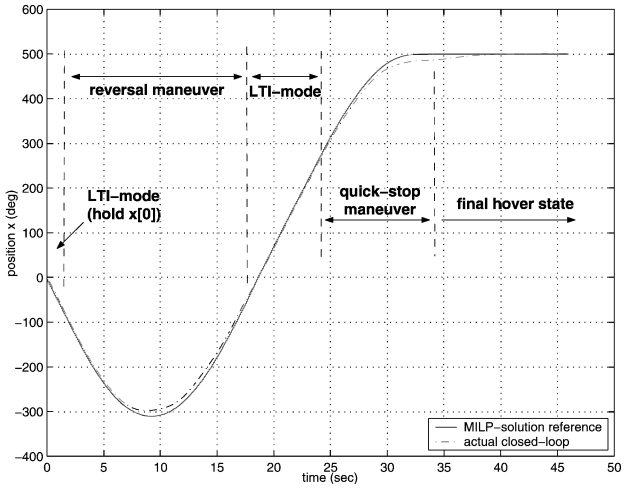

Fig. 10 Path-planning travel (position) solution for one-dimensional retreat-to-hover scenario.

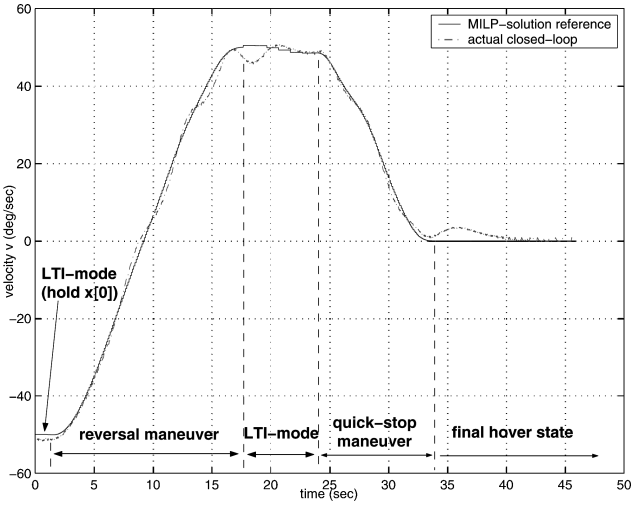

Fig. 11 Path-planning velocity solution for one-dimensional retreat-to-hover scenario.

respectively. Also shown (dashed line) are the actual experimental closed-loop vehicle tracking signals for these planning states.

After an initial one-decision step hold (a consequence of a user choice to fix the first LTI-mode command at the initial cruise speed), the planner immediately executes a direction-reversal, then follows a slowly varying LTI-mode cruise mode for around 7 s, and then executes a quick-stop maneuver at $\bar{v}_i = +48.48$ deg/s. Repeated guidance solutions over varying sets of initial helicopter cruise speeds consistently result in similar strategies. The rapid reversal maneuver is significantly faster than an LTI-mode-only deceleration–acceleration motion and, thanks to the continuously parameterized maneuver capability, can be executed at any speed satisfying $-65 \le v \le -5$. Compared to previous MILP planners with fixed maneuver elements,[52] there is no need to spend valuable time changing speed to meet their fixed maneuver boundary condition.

VI. Conclusions

The method for synthesizing parameterized maneuver classes described in this paper represents a very useful capability for real-time trajectory generation problems commonly found in autonomous vehicle guidance. It provides the following practical advantages:

First, it enables user freedom in choosing example motions and imposing design constraints. Proper example choice allows the engineer to select a "style" for a given maneuver class. For example, if a set of agile maneuvers is desired, one might choose highly aggressive prototype motions that maximize control effort. Alternatively, one might select comparatively slower motions that minimize control effort when slowly varying, smooth motions are desired. Constraint selection allows the user to design specific variations within a maneuver class or govern how the class interfaces with higher-level motion planners.

Further, trajectory interpolation is a viable method for real-time trajectory generation. Depending on computing resources, nonlinear programming can take tens of seconds or even several minutes to calculate a single feasible vehicle maneuver. In contrast, interpolation can compute an entire class of feasible maneuvers in about one tenth the computing time. On-line trajectory generation can be made extremely fast by storing samples of maneuver class members in memory and then performing interpolation over shorter intervals, enabling near real-time trajectory computation.

Also, when combined with higher-level path planners, such as the MILP-based framework, parameterized trajectories can greatly improve planning performance, because maneuver boundary conditions are no longer fixed, but can instead vary over significant ranges. This broadening of available trajectories requires very few extra formalisms in the MILP problem statement. Overall, the maneuver classes help create a logical hierarchical decomposition of path planning and inner-loop trajectory generation functions and is not necessarily limited to MILP-based frameworks.

The basic method outlined in this paper is applicable to any system the trajectories of which can be generated by nonlinear programs with differentiable constraint functions. Therefore, applicable systems include those with differentially flat models, invertible system models subject to nonholonomic constraints (the three-degree-of-freedom helicopter is of this type), state feedback linearizable models, and systems employing state-input collocation formulations (also known as transcription).

Future applications include parameterized aerobatic helicopter maneuvers, design of agile fixed-wing motions based on expert fighter pilot demonstration, and human demonstrationbased robotic machine learning. Of theoretical interest is the determination of general "interpolability" conditions, indicating when two arbitrary feasible motions will define a well-posed maneuver class.

Table A1 Identified model parameters for three-degree-of-freedom helicopter nonlinear model

Parameter	Estimate	Parameter	Estimate
a_1	0.0252	b_4	1.42
a_2	0.0525	d_1	0.112
θ_a	0.0827	d_2	0.243
b_0	0.131	d_3	0.504
b_1	0.163	d_4	0.0905
b_2	1.58	d_5	0.0400
b_3	0.449		

Appendix: System Identification

The data in Table A1 give numerical coefficient values for the helicopter nonlinear model of Eqs. (13) and resulted from a two-part vehicle system identification experiment. The first step involved developing a linear hover model (for steady flight condition: $v = 0$, $z = 0$) using a combination of piloted and signal-generated frequency sweep inputs and measured vehicle responses. Application of the rotorcraft system identification package CIFER® then produced frequency and time domain linear hover models.[56] The second step determined the full nonlinear equations of motion and coefficients by physically modeling the observed nonlinear effects (thrust vector tilting, speed sensitivity, voltage-to-rotor-thrust relations,[47] etc.) and then performing specific regression experiments to determine actual coefficient values. Reference 54 gives a detailed discussion of the physical modeling and system identification procedures used to obtain the nonlinear vehicle model.

Acknowledgments

This research was funded under Draper Laboratory Internal Research and Development Project 13177, NASA Ames Research Center Project NAG 2-1552 for "Motion Planning for Agile Maneuvering Vehicles," U.S. Air Force Research Laboratory Project F33615-01-C-1850 for "Safe Operation of Multi-Vehicle Systems," and Navy–Office of Naval Research Project N00014-03-1-0171 for "Integrated Flight Management and Situational Awareness for Highly Maneuverable Autonomous Systems." The authors thank Leena Singh, John Hauser, and Brent Appleby for their suggestions regarding algorithm development, as well as Tom Schouwenaars for his helpful discussions on higher-level motion planning and the incorporation of maneuver elements. Masha Ishutkina, Steve Hall, and the MIT Department of Aeronautics and Astronautics were extremely helpful in providing access to and support for the Quanser flights. The authors also thank the anonymous reviewers for their helpful feedback.

References

[1]Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*, revised printing, Taylor and Francis, New York, 1975, pp. 177–179.

[2]Bertsekas, D. P., *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont, MA, 1999, pp. 307–309.

[3]Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.

[4]Betts, J. T., *Practical Methods for Optimal Control Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, Philadelphia, 2001, pp. 61–79.

[5]Hull, D. G., "Conversion of Optimal Control Problems into Parameter Optimization Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 57–60.

[6]Chauvin, J., Sinegre, L., and Murray, R. M., "Nonlinear Trajectory Generation for the Caltech Multi-vehicle Wireless Testbed," European Control Conf., 2003.

[7]Faiz, N., Agrawal, S. K., and Murray, R. M., "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 219–227.

[8]Kim, S. K., and Tilbury, D., "Trajectory Generation for a Class of Nonlinear Systems with Input and State Constraints," *Proceedings of the American Control Conference*, American Automatic Control Council, Evanston, IL, 2001, pp. 4908–4913.

[9]Milam, M. B., Franz, R., and Murray, R. M., "Real-Time Constrained Trajectory Generation Applied to a Flight Control Experiment," *IFAC World Conference*, Barcelona, July 2002.

[10]Milam, M., Mushambi, K., and Murray, R. M., "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems," *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 1, IEEE Publications, Piscataway, NJ, 2000, pp. 845–851.

[11]Petit, N., Milam, M., and Murray, R., "Inversion Based Constrained Trajectory Optimization," 5th IFAC Symposium on Nonlinear Control System Design, 2001.

[12]van Nieuwstadt, M., Rathinam, M., and Murray, R. M., "Differential Flatness and Absolute Equivalence of Nonlinear Control Systems," *SIAM Journal on Control and Optimization*, Vol. 36, No. 4, 1998, pp. 1225–1239.

[13]Verma, A. J., and Junkins, J. L., "Trajectory Generation for Transition from VTOL to Wing-Bourne Flight Using Inverse Dynamics," AIAA Paper 2000-971, Jan. 2000.

[14]Seywald, H., "Trajectory Optimization Based on Differential Inclusion," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 3, 1994, pp. 480–487.

[15]Dasgupta, A., and Nakamura, Y., "Making Feasible Walking Motion of Humanoid Robots from Human Motion Capture Data," *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ, 1999, pp. 1044–1049.

[16]Schaal, S., "Learning from Demonstration," *Advances in Neural Information Processing Systems*, edited by M. C. Mozer, M. I. Jordan, and T. Petsché, Vol. 9, MIT Press, Cambridge, MA, 1997, pp. 1040–1046.

[17]Ijspeert, A. J., Nakanishi, J., and Schaal, S., "Trajectory Formation for Imitation with Nonlinear Dynamical Systems," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2001)*, IEEE Publications, Piscataway, NJ, 2001, pp. 752–757.

[18]Arikan, O., and Forsyth, D. A., "Interactive Motion Generation from Examples," *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, ACM SIGGRAPH, Association for Computing Machinery, New York, 2002, pp. 483–490.

[19]Kovar, L., Gleicher, M., and Pighin, F., "Motion Graphs," *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, ACM SIGGRAPH, Association for Computing Machinery, New York, 2002, pp. 473–482.

[20]Popović, J., Seitz, S. M., and Erdmann, M., "Motion Sketching for Control of Rigid Body Simulations," *ACM Transactions on Graphics*, Vol. 22, No. 4, 2003, pp. 1034–1054.

[21]Witken, A., and Popović, Z., "Motion Warping," *Proceedings of 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1995)*, edited by R. Cooke, ACM SIGGRAPH, Association for Computing Machinery, New York, 1995, pp. 105–108.

[22]Amit, R., and Matarić, M. J., "Parametric Primitives for Motor Representation and Control," *Proceedings of the International Conference on Robotics and Automation (ICRA-2002)*, IEEE Publications, Piscataway, NJ, 2002, pp. 863–868.

[23]Amit, R., and Matarić, M. J., "Learning Movement Sequences from Demonstration," *Proceedings of the International Conference Development and Learning (ICDL-2002)*, Inst. of Electrical and Electronics Engineers Computer Society, Los Alamitos, CA, 2002, pp. 302–306.

[24]Del Vecchio, D., Murray, R. M., and Perona, P., "Primitives for Human Motion: A Dynamical Approach," International Federation of Automatic Control, World Congress, CDS TR 01-009, 2002.

[25]Fod, A., Matarić, M. J., and Jenkins, O. C., "Automated Derivation of Primitives for Movement Classification," *Autonomous Robots*, Vol. 12, No. 1, 2002, pp. 39–54.

[26]Jenkins, O. C., and Matarić, M. J., "Deriving Action and Behavior Primitives from Human Motion Data," *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2002)*, IEEE Publications, Piscataway, NJ, 2002, pp. 2551–2556.

[27]Hauser, J., and Meyer, D., "Trajectory Morphing for Nonlinear Systems," *Proceedings of the American Control Conference*, American Automatic Control Council, Evanston, IL, 1998, pp. 2065–2070.

[28]Gavrilets, V., Frazzoli, E., Mettler, B., Piedmonte, M., and Feron, E., "Aggressive Maneuvering of Small Autonomous Helicopters: A Human-Centered Approach," *International Journal of Robotics Research*, Vol. 20, No. 10, 2001, pp. 795–807.

[29]Gavrilets, V., Mettler, B., and Feron, E., "Human-Inspired Control Logic for Automated Maneuvering of Miniature Helicopter," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 752–759.

[30]Piedmonte, M., and Feron, E., "Aggressive Maneuvering of Aerial Vehicles: A Human-Centered Approach," *Robotics Research: The Ninth International Symposium*, edited by J. M. Hollerbach and D. E. Koditschek, New York, Springer, 2000, pp. 413–420.

[31]Allgower, E. L., and Georg, K., *Numerical Continuation Methods, An Introduction*, Springer-Verlag, Berlin, 1990, pp. 37–74.

[32]Rheinboldt, W. C., *Numerical Analysis of Parametrized Nonlinear Equations*, University of Arkansas Lecture Notes in the Mathematical Sciences, Vol. 7, Wiley, New York, 1986, pp. 113–139.

[33]Roweis, S., and Saul, L., "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, Vol. 290, No. 5500, 2000, pp. 2323–2326.

[34]Tenenbaum, J. B., de Silva, V., and Langford, J. C., "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, Vol. 290, No. 5500, 2000, pp. 2319–2323.

[35]Thomson, D. G., and Bradley, R., "The Mathematical Definition of Helicopter Maneuvers," *Journal of the American Helicopter Society*, Vol. 42, No. 4, 1997, pp. 307–309.

[36]Bemporad, A., and Morari, M., "Control of Systems Integrating Logic, Dynamics, and Constraints," *Automatica*, Vol. 35, No. 3, 1999, pp. 407–427.

[37]Borrelli, F., *Constrained Optimal Control of Linear and Hybrid Systems*, Springer-Verlag, Berlin, 2003, pp. 25–62.

[38]Brockett, R. W., "Hybrid Models for Motion Control Systems," *Essays on Control: Perspectives in the Theory and its Applications*, edited by H. L. Trentelman and J. C. Willems, Birkhäuser, Boston, 1993, pp. 29–53.

[39]Jongen, H. Th., and Weber, G.-W., "On Parametric Nonlinear Programming," *Annals of Operations Research*, Vol. 27, No. 1–4, 1990, pp. 253–284.

[40]Lundberg, B. N., and Poore, A. B., "Numerical Continuation and Singularity Detection Methods for Parametric Nonlinear Programming," *SIAM Journal on Optimization*, Vol. 3, No. 1, 1993, pp. 134–154.

[41]Rakowska, J., Haftka, R. T., and Watson, L. T., "An Active Set Algorithm for Tracing Parametrized Optima," *Structural Optimization*, Vol. 3, 1991, pp. 29–44.

[42]Golub, G. H., and Van Loan, C. F., *Matrix Computations*, 3rd ed., John Hopkins Univ. Press, Baltimore, MD, 1996, pp. 256–264.

[43]Spiteri, R. J., Pai, D. K., and Ascher, U. M., "Programming and Control of Robots by Means of Differential Algebraic Inequalities," *IEEE Transactions on Robotics and Animation*, Vol. 16, No. 2, 2000, pp. 135–145.

[44]Spiteri, R. J., Ascher, U. M., and Pai, D. K., "Numerical Solution of Differential Systems with Algebraic Inequalities Arising in Robot Programming," *Proceedings of the 1995 International Conference on Robotics and Automation*, Vol. 3, Robotics and Automation Society, Danvers, MA, 1995, pp. 2373–2380.

[45]"3D Helicopter System (with Active Disturbance)," User's Manual, Quanser Consulting, Markham, ON, Canada, 2003.

[46]de Boor, C., *A Practical Guide to Splines*, revised edition, Applied Mathematical Sciences, Vol. 27, edited by J. E. Marsden and L. Sirovich, Springer-Verlag, New York, 2001, pp. 109–144.

[47]Leishman, J. G., *Principles of Helicopter Aerodynamics*, Cambridge Univ. Press, Cambridge, England, U.K., 2000, pp. 43, 44, 66–68.

[48]Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129.

[49]Mettler, B., Valenti, M., Schouwenaars, T., Frazzoli, E., and Feron, E., "Rotorcraft Motion Planning for Agile Maneuvering," *Proceedings of the 58th Forum of the American Helicopter Society*, Alexandria, VA, 2002.

[50]Bellingham, J., Richards, A., and How, J. P., "Receding Horizon Control of Autonomous Aerial Vehicles," *Proceedings of the American Control Conference*, American Automatic Control Council, Evanston, IL, 2002, pp. 3741–3746.

[51]Richards, A., and How, J. P., "Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming," *Proceedings of the American Control Conference*, American Automatic Control Council, Evanston, IL, 2002.

[52]Schouwenaars, T., Mettler, B., Feron, E., and How, J., "Hybrid Architecture for Full-Envelope Autonomous Rotorcraft Guidance," American Helicopter Society 59th Annual Forum, Alexandria, VA, 2003.

[53]Schouwenaars, T., Mettler, B., Feron, E., and How, J., "Hybrid Model for Receding Horizon Guidance of Agile Autonomous Rotorcraft," 16th IFAC Symposium on Automatic Control in Aerospace, 2004.

[54]Dever, C. W., "Parametrized Maneuvers for Autonomous Vehicles," Ph.D. Dissertation, Massachusetts Inst. of Technology, Cambridge, MA, Sept. 2004.

[55]Dever, C., Mettler, B., Feron, E., Popović, J., and McConley, M., "Trajectory Interpolation for Parametrized Maneuvering and Flexible Motion Planning of Autonomous Vehicles," AIAA Paper 2004-5143, Aug. 2004.

[56]Tischler, M. B., and Cauffman, M. G., "Comprehensive Identification from Frequency Responses: Flight Applications to BO-105 Coupled Rotor/Fuselage Dynamics," *Journal of the American Helicopter Society*, Vol. 37, No. 3, 1992, pp. 3–17.