Trajectory Optimization Procedures for Rotorcraft Vehicles, their Software Implementation and Applicability to Models of Varying Complexity

Carlo L. Bottasso

Associate Professor,

Politecnico di Milano, Dipartimento di Ingegneria Aerospaziale,

Milano, Italy

Giorgio Maisano

Ph.D. candidate,

Politecnico di Milano, Dipartimento di Ingegneria Aerospaziale,

Milano, Italy

Francesco Scorcelletti

Flight mechanics Engineer and Ph.D. candidate,

Agusta Westland,

Cascina Costa, Varese, Italy

Abstract

This paper describes numerical procedures for the solution of trajectory optimization problems in rotorcraft flight mechanics. We specifically consider procedures which can be easily interfaced with black-box flight simulators, with minimal assumptions on the functionalities of such third-party software components, and which can cater to a wide range of vehicle models of varying complexity. At first we identify in the direct approach to the solution of maneuver optimal control problems the

Presented at the American Helicopter Society 64th Annual Forum, Montréal, Canada, April 29 – May 1, 2008.

method of choice for this class of problems. Next, we formulate the direct transcription and the direct multiple shooting approaches, we discuss their characteristics and identify their respective optimal application areas. Finally, we describe the functionality and architecture of a general purpose code implementing both methods. The capabilities of the proposed procedures are demonstrated with the help of practical examples of industrial relevance, regarding both helicopters and tilt-rotors.

Introduction

The term *trajectory optimization* refers to the process of computing the optimal control inputs and the resulting response of a model of a vehicle, a rotorcraft in the present case, which minimize a cost function (or maximize and index of performance) while satisfying given constraints (which specify, for example, the vehicle flight envelope boundaries, and/or safety and procedural requirements for a maneuver of interest).

Notice that this problem differs significantly from the usual and more common problem of *forward simulation* starting from given initial conditions under the action of control inputs, both in the case when the control time histories are given a priori or when they are computed by a flight control system or tracking controller. In fact, for forward time simulation of rotorcraft vehicles, several general purpose codes are available and widely used by industry, as for example CAMRAD/JA [21], EUROPA [4], FLIGHTLAB [3], GenHel [26], etc. In order to use each one of these simulators, one has to specify a time history of control inputs, or a trajectory or sequence of trim conditions that the vehicle should track. In a trajectory optimization problem none of these two pieces of information, controls or trajectory, are known, so that standard forward simulators can not be used directly.

A typical example would be a continued take-off in one engine failure. FAA Category-A certification rules [1] describe in detail the maneuver and specify constraints that must be met for the vehicle to be safely flyable in such an event. An analyst might be interested in verifying the vehicle performance according to such rules, conducting trade studies on design parameters of interest or, for example, determining the maximum certifiable take-off weight. In all such cases, the sole use of a standard forward simulator would imply gross approximations and hypotheses on the necessary control inputs to fly that

maneuver or on the vehicle trajectory, both being unknown. Therefore, the analyst would be unable to determine with precision and confidence which is the maximum theoretical performance of the vehicle, and this would inevitably lead to conservative design choices or restrictions on the vehicle operation. Clearly, there are innumerable application areas other than Category-A certification which pose similar problems, such as optimal autorotation, landing procedures after tail-rotor loss, approach and departure from flight decks, definition of ADS-33 mission task elements [2], etc.

To answer these needs, trajectory optimization codes implement appropriate numerical methods which, interacting with forward simulators, augment their capabilities in order to compute the controls which fly the vehicle model in an optimal and constraint-satisfycing way. Software procedures for automatically and reliably solving trajectory optimization problems represent valuable tools in the design and development of a new product, and they find applicability in the vehicle preliminary design and certification phases, in the formulation of operational procedures that might abate noise, increase capacity or safety, reduce emissions, or improve other operational parameters in terminal areas.

Computer implementations of trajectory optimization procedures for rotorcraft flight mechanics vehicle models have been previously described by Okuno and Kawachi [28], Carlson and Zhao [19], and Bottasso et al. [13, 14]. The extension of such procedures to handle fine-scale aero-servo-elastic comprehensive vehicle models have been first described by Bottasso et al. [15, 16].

In the present work, we describe the Trajectory Optimization Program (TOP), developed at the Department of Aerospace Engineering of the Politecnico di Milano. TOP has been designed to solve efficiently trajectory optimization problems for rotorcraft vehicles, and has some unique features:

- *General applicability to an ample variety of maneuvers.* The user has total control on the specification of the optimization cost function and on the constraints, which collectively express the maneuver of interest in a clear and mathematically sound way as an optimal control problem.

- *General applicability to existing flight mechanics simulators.* TOP has both a built-in flight mechanics model and a general interface towards black-box external flight simulators. No assumptions are made on the black-box vehicle simulators, except that it must be possible to set the model initial conditions and to provide a time-history of control inputs on a certain time window. The interfacing

with external simulators is here demonstrated in the case of the industrial software FLIGHTLAB [3], although the code has also been successfully interfaced with EUROPA [4].

- *Multiple solution strategies.* The code implements two different approaches for the solution of optimal control problems, namely the direct transcription and the direct multiple shooting methods. The term *direct* refers to the fact that the optimal control equations need not be derived [8], which is crucial for complex or black-box models. In fact, the ability to readily use available vehicle models, given the large effort and cost in their development and validation, is considered here a fundamental prerequisite and it is also the only way that the new capabilities offered by trajectory optimization methods can be rapidly transitioned to industry. The two methods described here have complementary features, and the choice of one or the other helps to optimize efficiency, robustness and compatibility with external simulators. Each method has an optimal application area which depends on the model complexity, so that a hierarchy of models can be effectively supported with one single general purpose code.

- *Numerical efficiency and robustness.* The code is numerically efficient, and takes full advantage of the sparsity of discretized optimal control problems. Furthermore, TOP implements a number of grid adaption strategies which help minimize the growth of the problem dimension and hence the computational cost, automatic boot-strapping procedures for improving robustness and convergence, and scaling of the unknowns to improve conditioning of the numerical problem.

The paper is organized according to the following plan. At first we formulate in general terms the trajectory optimization problem. A discussion about the possible numerical solution strategies to solve this problem is given next; namely, we first describe the direct transcription approach and then we review the direct multiple shooting method. The paper continues describing the software architecture, including the graphical user interface, the definition of the maneuver optimal control problem through its cost function and constraints, and gives a brief review of the implementation and handling of some specific functionalities in the code. Finally, the capabilities of the new software procedures are demonstrated with the help of a number of maneuvering rotorcraft problems.

Formulation of Maneuvers as Optimal Control Problems

A maneuver can be defined as a finite-time transition between two trim conditions [22]*. Clearly, given a starting trim and an arrival trim, there is an infinite number of ways to transition between the two. A way to remove this arbitrariness is to formulate a maneuver as an optimal control problem [14, 15], where one minimizes a cost (time, altitude loss, control activity, fuel consumption, etc.) which in general is some given function of the vehicle states and control inputs. The solution of the optimization problem must satisfy the dynamic and kinematic equations of the vehicle, the initial and final conditions corresponding to the start and arrival trims, and all other equality and inequality constraints which need to be met in order to satisfy given performance and procedural requirements.

Consider a flight mechanics vehicle model $\mathcal{M}$, which includes structural and aerodynamic models of the vehicle components, possibly (but not necessarily) using a multibody approach [6]. The dynamics of model $\mathcal{M}$ can in general be described in terms of a set of non-linear index 1-3 differential algebraic equations written as

$$\boldsymbol{f}_{\mathrm{SD}}(\dot{\boldsymbol{x}}_{\mathrm{SD}}, \boldsymbol{x}_{\mathrm{SD}}, \boldsymbol{\lambda}, \boldsymbol{x}_{\mathrm{A}}, \boldsymbol{u}) = 0, \tag{1a}$$

$$\boldsymbol{c}(\boldsymbol{x}_{\mathrm{SD}}) = 0, \tag{1b}$$

$$\boldsymbol{M}\dot{\boldsymbol{x}}_{\mathrm{A}} + \boldsymbol{L}\boldsymbol{x}_{\mathrm{A}} - \boldsymbol{\tau}(\boldsymbol{x}_{\mathrm{SD}}, \boldsymbol{u}) = 0, \tag{1c}$$

where $\boldsymbol{x}_{\mathrm{SD}}$ are the structural dynamics states (including states which describe rigid and possibly flexible rotor(s), fuselage, engine, etc.), $\boldsymbol{\lambda}$ are constraint-enforcing Lagrange multipliers in a multibody vehicle model, $\boldsymbol{x}_{\mathrm{A}}$ are aerodynamic states (e.g. dynamic inflow variables), and $\boldsymbol{u}$ is the control input vector. Equations (1a) group together the equations of dynamic equilibrium and the kinematic equations. Equations (1b) represent mechanical joint constraint equations in a multibody vehicle model, while Eqs. (1c) are the aerodynamic state equations. Finally, the notation $\dot{(\cdot)} = \mathrm{d}(\cdot)/\mathrm{d}t$ indicates a derivative with respect to time t.

*Although this is the only rigorous definition of a maneuver, in the context of the present work it will be more useful to use the term maneuver more loosely, and we will often consider the case of terminal conditions which are not trimmed.

For the sake of simplicity, in the following we will consider that the Lagrange multipliers $\boldsymbol{\lambda}$ and redundant structural dynamics states can always be formally eliminated in favor of a minimal set of coordinates [23]. The interested reader can consult Ref. [12] for a description of the solution of optimal control problems for multibody models in redundant form; while this does not pose technical difficulties, it requires a heavier notation and complicates the discussion. Therefore, the governing equations will be assumed to be of the ordinary differential type and will be simply expressed as

$$\boldsymbol{f}_{\text{SD}}(\dot{\boldsymbol{x}}_{\text{SD}}, \boldsymbol{x}_{\text{SD}}, \boldsymbol{x}_{\text{A}}, \boldsymbol{u}) = 0, \tag{2a}$$

$$\boldsymbol{M}\dot{\boldsymbol{x}}_{\text{A}} + \boldsymbol{L}\boldsymbol{x}_{\text{A}} - \boldsymbol{\tau}(\boldsymbol{x}_{\text{SD}}, \boldsymbol{u}) = 0. \tag{2b}$$

When using quasi-steady aerodynamics, the aerodynamic model expressed by Eqs. (2b) and its associated aerodynamic states $\boldsymbol{x}_{\text{A}}$ are of an algebraic nature. The numerical solution is in that case performed by eliminating the algebraic aerodynamic variables, usually through a fixed point iteration. Therefore, even in that case, we can consider an ODE model with no loss of generality.

It will be convenient to use a more synthetical form of the above equations in the following pages, and hence we will write the vehicle model as

$$\boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{u}) = 0, \tag{3a}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}), \tag{3b}$$

where $\boldsymbol{x} = (\boldsymbol{x}_{\text{SD}}^T, \boldsymbol{x}_{\text{A}}^T)^T$ and $\boldsymbol{f}$ stacks together Eqs. (2a) and (2b). In addition, Eq. (3b) defines a vector of outputs $\boldsymbol{y}$. The outputs will typically represent some global vehicle states which describe its gross motion, such as position, orientation, linear and angular velocity of a vehicle-embedded frame with respect to an inertial frame of reference, or other quantities useful for formulating the maneuver optimal control problem.

Equations (3a) are solved for the forward simulation problem by providing a time history of control inputs $\boldsymbol{u}(t)$ and initial conditions on the states $\boldsymbol{x}(0) = \boldsymbol{x}_0$. Accordingly, one obtains also the associated values of the outputs through (3b).

The trajectory optimization problem is defined on the interval $\Omega = [0, T]$, $t \in \Omega$, where the final time T

is typically unknown and must be determined as part of the solution to the problem. Specific events might be associated with unknown time instants T_i, $0 < T_i < T$, as for example the reaching of specific values of certain states, the jettisoning of part of the cargo, etc. The present code implementation can handle multiple internal events, but we do not consider this case in the following for the sake of simplicity, since this does not pose any conceptual difficulty that is worth addressing in detail.

The maneuver optimization problem consists in finding the control function $\boldsymbol{u}(t)$, and hence through (3) the associated function $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$, which minimize the cost

$$J = \phi(\boldsymbol{y}, t)\big|_0^T + \int_0^T L(\boldsymbol{y}, \boldsymbol{u}, \dot{\boldsymbol{u}}, t)\,\mathrm{d}t. \tag{4}$$

The first term in the previous expression is a boundary term which accounts for values of the outputs at the initial and/or final instants, while the second term is an integral cost term.

The minimizing solution must satisfy the vehicle equations of motion (3), together with the boundary (initial and/or terminal) conditions on the states

$$\boldsymbol{g}_{\mathrm{bc}}\big(\boldsymbol{x}(0)\big) \in [\boldsymbol{g}_{\mathrm{bc},0}^{\min}, \boldsymbol{g}_{\mathrm{bc},0}^{\max}], \tag{5a}$$

$$\boldsymbol{g}_{\mathrm{bc}}\big(\boldsymbol{x}(T)\big) \in [\boldsymbol{g}_{\mathrm{bc},T}^{\min}, \boldsymbol{g}_{\mathrm{bc},T}^{\max}]. \tag{5b}$$

Notice that the initial and final values of the states $\boldsymbol{x}(0)$ and $\boldsymbol{x}(T)$ are typically determined by solving a separate trim problem off-line, whose details depend on the specifics of the vehicle model being considered. Often, the final conditions are not required to represent a trim state, in which case the exit conditions can be written in terms of the sole outputs as

$$\boldsymbol{g}_{\mathrm{bc}}\big(\boldsymbol{y}(T)\big) \in [\boldsymbol{g}_{\mathrm{bc},T}^{\min}, \boldsymbol{g}_{\mathrm{bc},T}^{\max}]. \tag{6}$$

Other maneuver-defining and/or envelope-protection constraints can be expressed as generic algebraic non-linear constraints of the form

$$\boldsymbol{g}_{\mathrm{gen}}(\boldsymbol{y}, \boldsymbol{u}, t) \in [\boldsymbol{g}_{\mathrm{gen}}^{\min}, \boldsymbol{g}_{\mathrm{gen}}^{\max}], \tag{7}$$

integral conditions

$$\frac{1}{T}\int_0^T \boldsymbol{g}_{\mathrm{int}}(\boldsymbol{y}, \boldsymbol{u}, t)\,\mathrm{d}t \in [\boldsymbol{g}_{\mathrm{int}}^{\min}, \boldsymbol{g}_{\mathrm{int}}^{\max}], \tag{8}$$

constraints at unknown internal events T_i

$$\boldsymbol{g}_{\text{event}}(\boldsymbol{y}, \boldsymbol{u}, T_i) \in [\boldsymbol{g}_{\text{event}}^{\min}, \boldsymbol{g}_{\text{event}}^{\max}], \tag{9}$$

or simple bounds

$$\boldsymbol{y} \in [\boldsymbol{y}^{\min}, \boldsymbol{y}^{\max}], \tag{10a}$$

$$\boldsymbol{u} \in [\boldsymbol{u}^{\min}, \boldsymbol{u}^{\max}]. \tag{10b}$$

In summary, the maneuver optimal control problem can be expressed as

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{u},T} \quad \text{Cost } J, \text{ Eq. (4)}, \tag{11a}$$

$$\text{s.t.:} \quad \text{ODE system (3)}, \tag{11b}$$

$$\text{Constraints (5–10)}. \tag{11c}$$

Direct Solution of Trajectory Optimization Problems

The indirect approach to the solution of the maneuver optimal control problem (11) amounts to augmenting the cost (4), by adjoining the governing equations (3) with a set of co-states, and adjoining all other constraints conditions (5–10) with Lagrange multipliers. By imposing the stationarity of the augmented cost, one derives a new set of equations with their associated boundary conditions, which govern the optimal control problem [18]. The resulting boundary value problem can then be discretized using a suitable numerical method defined on a computational grid. This transforms the infinite dimensional boundary value problem into a discrete problem, whose unknowns are the values of the variables (states, co-states, Lagrange multipliers, controls) on the computational grid.

It is clear that the implementation of this classical approach requires the manipulation of the governing equations (3) in order to derive the optimal control equations. This is a non-trivial task for complex and highly non-linear models, which might necessitate the use of symbolic manipulation or automatic differentiation tools to be carried out effectively. More importantly, this approach must be ruled out whenever

one does not have access to the analytical expression of Eqs. (3) or their software implementation, as it is the case whenever the flight simulator is a third-party black-box code.

To overcome these limitations of the indirect approach, one can use the direct method [8]. In this case, instead of first optimizing and then discretizing, one first discretizes the model equations (3) through a numerical method. This has the effect of discretizing also the cost function (4) and the constraints (5–10). This in turn defines a discrete parameter optimization or non-linear programming (NLP) problem [24], which can be written in general as

$$\min_{\boldsymbol{z}} \quad K(\boldsymbol{z}), \tag{12a}$$

$$\text{s.t.} \quad \boldsymbol{a}(\boldsymbol{z}) = \boldsymbol{0}, \tag{12b}$$

$$\boldsymbol{b}(\boldsymbol{z}) \in [\boldsymbol{b}^{\min}, \boldsymbol{b}^{\max}], \tag{12c}$$

where $\boldsymbol{z}$ is a set of algebraic unknowns, and K is a scalar objective function which represents an approximation of the cost J of Eq. (11a). The equality constraints (12b) are generated by the discretization of the equations of motion (11b), while the inequality constraints (12c) by all maneuver-defining constraints (11c). The specific form of the vector of algebraic unknowns and of the constraints depends on the method used for performing the discretization, as detailed below.

Necessary conditions for a constrained optimum for problem (12) are obtained, similarly to the optimal control case, by combining the objective K with the constraints through the use of Lagrange multipliers, and imposing the stationarity of the augmented objective function. By using the direct approach, one does not need to manipulate the equations of motion of the vehicle, since all that is required is the evaluation of the discretized equations on a step or sequence of steps, and this enables the interfacing to black-box vehicle simulators. For further possible advantages of the direct method, the interested reader is referred to Ref. [8].

We have implemented two different direct methods in the TOP code, namely direct transcription and direct multiple shooting, which are described in the next section. Both methods lead to sparse NLP problems (12), which in the present implementation are solved using sequential quadratic programming (SQP) [7].

We have found that, for the problems considered here, it is important for robustness to consider a

scaled version of the NLP problem, where the NLP variables $\boldsymbol{z}$ are replaced by scaled variables $\bar{\boldsymbol{z}} = \mathrm{diag}(\boldsymbol{w}_z)\boldsymbol{z}$, $\boldsymbol{w}_z$ being scaling coefficients chosen so that the new unknowns are $\bar{\boldsymbol{z}} \approx \mathcal{O}(1)$. Furthermore, since at convergence of the SQP solution procedure many constraints are active (at least all of the equality constraints (12b)), we have found that faster convergence of the SQP solver is obtained by using a slightly looser tolerance for the equality constraint feasibility than for the solution optimality tolerances. This in fact has the effect of avoiding repeated changes in the active set when trying to satisfy optimality at the end of the process, which eases the reaching of the exit condition of the NLP solver with basically no effect on the solution accuracy.

Direct transcription

We consider the partition of the time interval Ω as $0 = t_0 < t_1 < \ldots < t_N = T$, where the generic time element is $\Omega^n = [t_n, t_{n+1}]$, $n = (0, N-1)$, of time step size $h^n = t_{n+1} - t_n$. Here and in the following, quantities associated with the generic element vertex j are indicated using the notation $(\cdot)_j$, while quantities associated with the generic element k are labeled $(\cdot)^k$. Clearly, $h^n = h^n(T)$, i.e. the time step size is a function of the final time, when T is unknown.

In each time element Ω^n, the governing equations (3a) are discretized using a suitable numerical method. The resulting discrete equations are expressed here as

$$\boldsymbol{f}_h(\boldsymbol{x}_{n+1}, \boldsymbol{x}_n, \boldsymbol{u}^n, h^n) = 0, \qquad n = (0, N-1), \tag{13}$$

where $\boldsymbol{f}_h$ is an algorithmic approximation of function $\boldsymbol{f}$ of Eq. (3), $\boldsymbol{x}_n$, $\boldsymbol{x}_{n+1}$ are the values of the state vector at t_n and t_{n+1}, respectively, while $\boldsymbol{u}^n$ represents the value of the control vector within the step. In general there might be additional internal stages for both the state and the control variables, depending on the numerical method. For notational simplicity we do not consider that case here. With respect to this point, note further that in the case of higher order schemes with internal stages, Eqs. (13) might have been obtained by static elimination of these stages at the element level.

In the direct transcription case, the NLP problem (12) is defined as follows. First, the NLP variables are chosen as:

$$\boldsymbol{z} = (\boldsymbol{x}_{n=(0,N)}, \boldsymbol{u}^{n=(0,N-1)}, T)^T, \tag{14}$$

i.e. they are defined as the discrete states and control values on the computational grid, and the final time. Next, the cost J of Eq. (4) is discretized in terms of $\boldsymbol{z}$ as given by (14), obtaining the discrete cost K of Eq. (12a). Then, the discretized ODEs within each step, Eqs. (13), become the set of NLP equality constraints appearing in Eq. (12b). Finally, all other problem constraints and bounds, Eqs. (5–10), are expressed in terms of the NLP variables $\boldsymbol{z}$ and become the NLP inequality constraints of Eq. (12b).

The optimality conditions of the resulting discrete NLP problem converge to the optimality conditions of the optimal control problem (11) as the grid is refined and the number of discrete optimization variables goes to infinity [27].

The resulting problem is large but very sparse. In TOP, when using the internal flight mechanics model, the NLP problem Jacobian is evaluated using automatic differentiation with the ADOL-C code [25]. In the case of the black-box models (e.g. FLIGHTLAB and EUROPA), the Jacobian is obtained by sparse finite differencing.

Direct multiple shooting

We consider a partition of the time domain Ω given by $0 = t_0 < t_1 < \ldots < t_M = T$ with $\Omega^m = [t_m, t_{m+1}]$, $m = (0, M-1)$, where each Ω^m is a shooting segment. Here and in the following, quantities associated with the generic vertex between segments j are indicated using the notation $(\cdot)_j$, while quantities associated with the generic segment k are labeled $(\cdot)^k$. In each shooting segment Ω^m, the controls are discretized as $\boldsymbol{u}^m(t) = \sum_{i=1}^{N_c^m} s_i(t)\boldsymbol{u}_i^m$, where $s_i(t)$ are basis functions, in particular cubic splines in the present implementation, and $\boldsymbol{u}_i^m$ are N_c^m unknown discrete control values. Notice that we confine the control approximations on each shooting segment, instead of considering interpolations across segment boundaries; this has the effect of decreasing the computational cost of finite differencing by increasing the problem sparsity. Constraints are enforced at the shooting segment boundaries to enforce the continuity of the controls up to C^1.

In the case of direct multiple shooting, the NLP problem (12) is defined as follows. First, the set of NLP variables is chosen as:

$$\boldsymbol{z} = (\boldsymbol{x}_{m=(0,M)}, \boldsymbol{u}_{i=(1,N_c^m)}^{m=(0,M-1)}, T)^T, \tag{15}$$

i.e. they are defined as the discrete values of the states at the interfaces between shooting segments, the discrete values of the controls within each segment, and the final time.

Next, the governing ODEs (3) are marched in time within each shooting segment Ω^m, starting from the initial conditions provided by the values of the states $\boldsymbol{x}_m$ at the left boundary of the segment. The effect of the forward integration is to generate a discrete time history of states within Ω^m, which we label $\boldsymbol{x}_i^m$, $i = (1, N^m)$, where N^m is the number of steps taken in that segment. The last value of this sequence is named $\widetilde{\boldsymbol{x}}_{m+1} = \boldsymbol{x}_{N^m}^m$, and represents the new estimate of the state variables at the right boundary of the shooting segment. Segments are then glued together by imposing the following equality constraints

$$\boldsymbol{x}_m - \widetilde{\boldsymbol{x}}_m = 0, \qquad m = (1, M). \tag{16}$$

In the direct multiple shooting case, the cost J of Eq. (4) is discretized in terms of $\boldsymbol{z}$ as given by (15) and evaluated using the segment time histories $\boldsymbol{x}_i^m$; this yields the discrete cost K of Eq. (12a). Next, the gluing conditions (16) are used to express the set of NLP equality constraints appearing in Eq. (12b). All other problem constraints and bounds, Eqs. (5–10), are expressed in terms of the NLP variables $\boldsymbol{z}$ and become the NLP inequality constraints of Eq. (12b).

Multiple shooting segments are introduced for curing the well known instabilities of the single shooting method [5]. In fact, when using single shooting, small changes early in the trajectory can produce dramatic effects at the end of the trajectory itself, because the system non-linearities provide a mechanism for amplifying these changes; similar problems are found when analyzing unstable systems, which is often the case when considering rotorcraft vehicles. Consequently, convergence becomes very difficult if not impossible with single shooting. In most practical cases, the rather heuristic approach of breaking the problem domain into multiple segments alleviates these problems.

Considering the implementation of trajectory optimization solution procedures using third-party software, the direct multiple shooting method offers the advantage of a higher level interaction with the program as compared to the previous case. In fact, we just need to a) set the initial conditions at the beginning of each shooting segment and march in time till the end of the segment, under the action of given control inputs, and b) gather the solution of the forward simulation within and at the end of each segment. It is reasonable to assume that any black-box simulator will at least provide these minimum features.

The direct multiple shooting method is available in TOP for both the internal and external black-box vehicle models. The internal model is advanced in time using an explicit Runge-Kutta fourth-order scheme. In both cases, the NLP problem Jacobian is computed through sparse centered finite differencing by perturbation of the unknowns.

With multiple shooting, it is common to use adaptive step procedures to advance the solution in time within each segment, so that highly accurate solutions can be obtained. The size of the NLP problem clearly depends on the number of segments but, since the segments are consecutive, the resulting problem is sparse. Even more importantly, the size of the NLP problem does not depend on the number of internal steps taken in each segment. This is crucial if the vehicle model $\mathcal{M}$ has fast dynamic components which need small time steps to be resolved. In fact, treating such problems with the direct transcription approach will typically lead to very dense grids and hence to extremely large NLP problems, which would imply overwhelming computational costs.

Preferred application areas of trajectory optimization methods based on model complexity

The considerations of the previous pages regarding the different features of direct methods, lead to a classification of the solution procedures based on their applicability to rotorcraft models of varying complexity. In the context of the present discussion, a useful measure of complexity of a model is the computational cost necessary for advancing the solution forward in time of one physical time unit. Notice that we do not use the cost per time step, since time steps can be small or larger depending on the use of explicit or implicit methods, coupling procedures for the various physical fields, etc., so that covering a maneuver of some given duration might require a larger or smaller number of steps. Our proposed classification of optimal areas of applicability of the methods with varying model complexity is shown in Fig. 1.

With reference to rotorcraft vehicles, it is the opinion of the authors that simpler models (e.g. actuator disk-type rotor models with algebraic inflow variables [4, 30]) can be effectively solved with both direct transcription and multiple shooting; of the two, we typically recommend the former, since it is usually more robust, seems to enjoy a somewhat faster convergence in most cases and can handle more rigorously

and with fewer complications the presence of state inequalities (see the next section for a further discussion on this point). On the other hand, more refined rotor models (for example, using flapping blade states and dynamic inflow [3]) can at present be solved only with multiple shooting because of their fast solution components and the requirements they pose to the time step size for stability and/or accuracy.

Notice that there is an area of overlap of these two methods for the less complex models. This opens the way for a synergistic use of the two: one can solve the trajectory optimization problem first using direct transcription, which leads quickly and robustly to a solution, possibly of lower accuracy because computed on a rather coarse grid; next, the direct transcription solution can be used as an initial guess for a subsequent multiple shooting problem, whose role is to deliver a solution of greater accuracy using smaller time steps. Since multiple shooting is now initialized from a good starting guess, convergence is quick and not problematic.

Clearly, there is a limit to the complexity of the vehicle model which can be handled even in the case of multiple shooting. In fact, a converged solution will typically require of the order of thousands of forward dynamic simulations performed with the model. If such model has a high computational complexity according to the present definition (e.g. full FEM structural dynamics models [6] coupled with free wake [11], and more in general tightly coupled computational structural and fluid dynamics (CSD-CFD) [20]), the solution of the maneuver optimal control problem will incur into overwhelming costs. In order to make the problem computationally tractable, the multi-model steering algorithm (MMSA), a formulation which makes use of two models of the same vehicle, was fist proposed in Refs. [15] and [16].

Using MMSA, a coarse level flight mechanics model is used for solving the trajectory optimal control problem, termed in this context a planning problem. Being based on a reduced model of the vehicle with only a few degrees of freedom, the resulting non-linear multi-point boundary value problem can be solved at acceptable computational costs using the algorithms described in this paper. Next, the fine scale comprehensive model is steered in closed loop, following the trajectory computed at the flight mechanics level using a tracking controller. This amounts to a standard time marching problem for the comprehensive model, which is therefore also of reasonable computational cost. The flight mechanics model is iteratively updated for ensuring close matching of the trajectories flown by the two models, by resorting to system identification and model update techniques. This divide-and-conquer approach to the solution

of extremely large optimal control problems is not considered further in the rest of this document, whose focus is strictly on direct single model approaches using transcription and shooting.

We finally remark that these methods can be used in conjunction with a hierarchy of models of the same vehicle: solutions are computed starting from the crudest model and then used as initial guesses for initializing the computation on the next level model, using each time the most appropriate solution procedure.

Problems with state dependent inequality constraints, and the difference between the direct transcription and shooting methods

One possible issue with the multiple shooting approach is the solution of problems with state dependent inequality constraints [9]. In fact, only state variables at the shooting segment boundaries enter into the definition of the optimization unknowns (see the definition of $\boldsymbol{z}$ given in (15)). Therefore, one can not directly impose constraints or bounds on the values assumed by the states within the shooting segments; in other words, what happens to the solution within the segments is outside of the control of the optimization procedure, in marked contrast with the direct transcription method. This also implies that, contrary to the direct transcription case, it is not possible to prove that the numerical solution of the discrete problem converges to the optimal control solution when the number of time steps goes to infinity and state constraints are active.

This problem can be handled by looking more closely at the difference between the direct transcription and multiple shooting methods. In fact, consider the assembly of the two consecutive steps $n-1$ and n using the direct transcription method. This can be written as

$$\boldsymbol{f}_h(\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \boldsymbol{u}^{n-1}, h^{n-1}) = 0, \tag{17a}$$

$$\boldsymbol{f}_h(\boldsymbol{x}_{n+1}, \boldsymbol{x}_n, \boldsymbol{u}^{n}, h^{n}) = 0, \tag{17b}$$

which implies the boolean identification of the state vector at the end of the first step with the state vector

at the beginning of the second. Consider now the following alternative form of the assembly:

$$\boldsymbol{f}_h(\widetilde{\boldsymbol{x}}_n, \boldsymbol{x}_{n-1}, \boldsymbol{u}^{n-1}, h^{n-1}) = 0, \tag{18a}$$

$$\boldsymbol{f}_h(\widetilde{\boldsymbol{x}}_{n+1}, \boldsymbol{x}_n, \boldsymbol{u}^{n}, h^{n}) = 0, \tag{18b}$$

$$\boldsymbol{x}_n - \widetilde{\boldsymbol{x}}_n = 0. \tag{18c}$$

In this case the state vector at the end of the first step, $\widetilde{\boldsymbol{x}}_n$, is regarded as a separate variable from the state vector at the beginning of the second, $\boldsymbol{x}_n$, and the continuity of the states between the two steps is ensured by condition (18c). Clearly, the two forms of the assembly are perfectly equivalent, in the sense that at convergence they yield exactly the same solution. However, using the redundant state vector formulation of the problem, one can solve Eq. (18a) in terms of $\widetilde{\boldsymbol{x}}_n$, for example using a Newton-like iteration, and replace the result in Eq. (18c); the same procedure, repeated for all steps, effectively eliminates all the transcription constraints (18a), leaving only the gluing conditions (18c).

Consider now the multiple shooting case, with a time partition characterized by $N_m = 1$ for each shooting segment Ω^m, $m = (1, M-1)$. In other words, we march one single time step for shooting segment, ending up with many short segments instead of a few long ones, as in the classical multiple shooting method. Notice that the length of the segments will now have to be small, since we are taking a single step within each one of them; however, to avoid too stringent constraints on the time step length, we recommend in this case the use of an implicit time marching scheme. It is clear at this point that the resulting gluing conditions (16) for the multiple shooting approach with one step per segment are identical to the gluing conditions (18c) for the direct transcription approach after elimination of the transcription constraints.

In conclusion, an implementation of direct transcription using the redundant state vector form with static condensation of the transcription constraints is equivalent, at convergence, to a multiple shooting approach based on the same temporal discretization scheme using one time step per segment.

Therefore, it appears that a way to deal with state inequalities using a direct multiple shooting approach is as follows. At each time step within each segment, the state dependent inequality constraints are evaluated in terms of the segment-internal state values $\boldsymbol{x}_i^m$. If violations are detected within a shooting segment, that segment is broken into small segments in the violation area. On the new segments the

vehicle model will be advanced of one step of the length of the segment using an implicit scheme. Next, the NLP solution is restarted from the current solution projected onto the new grid. The procedure of segment update is repeated until convergence. As a matter of fact, this means that one is switching to a direct transcription treatment of the area where the violation was detected, which now enables the rigorous enforcement of the inequality constraints. Notice however that if the solution has fast solution components, e.g. in the case of detailed rotor models, this approach might not be always feasible because of the computational cost associated with a large number of direct transcription steps of small size.

An alternative approximate way of dealing with the problem is to simply break the segment where violations are detected into small ones while maintaining a shooting strategy on each of them, and continue with refinement until no more internal violations are found or when they become smaller than some given acceptable tolerance. This heuristic approach works in practice, was implemented in the TOP code and is demonstrated in the examples section.

Software Architecture

The user interacts with a pre-processing tool through a graphical user interface, which speeds up the work of the operator and reduces errors. This piece of software supports all phases of the definition of the maneuver optimal control problem, including the cost function and the maneuver-defining constraints, and of its numerical solution, including the definition of the solution strategies, of the time partition in segments and their meshes, of the data interpolation and associated quadrature rules, of the scaling factors for the solution variables, and of all the algorithmic parameters.

The cost function is defined as a weighed sum of powers of all unknowns and their rates, either in point or integral forms, all weights and exponents of each variable being selectable by the user. Simple bound constraints are defined through the graphical interface, while the code automatically generates the equality constraints (12b) stemming from the model dynamic equations according to the chosen solution strategy and time partition. More complex and problem-dependent additional constraint conditions are handled through user-defined constraint subroutines.

The vehicle model definition parameters are handled through a specialized sub-menu of the graphical

user interface for the internal flight mechanics model, while it must be handled externally in the case of third-party simulators which provide their own way of inputting the vehicle data. The pre-processor checks for inconsistencies in the input data, and generates a number of files which provide the input for the main optimization program.

The trajectory optimizer loads the data generated by the pre-processor, generates all necessary data structures and internal representations, and calls the NLP solver, which in turn evaluates the merit function and constraints at each step of the algorithm. The evaluation of cost and constraints necessitates of the interaction with the vehicle model according to the chosen solution strategy; this is handled through a generic interface which hides from the rest of the code whether the internal or the external vehicle models are used.

A maneuver can be covered by one or more phases separated by events. For each phase a different model can be specified, as long as the number and meaning of the states and control variables remains the same; more often than not, only different values of some of the parameters of the same vehicle model will be used in the different phases. Furthermore, in each phase different sets of constraints may be used.

For all phases, the grids which support the discretization of the problem are generated automatically from the user provided information, or from a mesh refinement module. Each phase time interval is partitioned into arcs, and each arc is associated with a grid. Since the total maneuver duration is usually unknown, partitions in phases and arcs and collocation of points is referred to the normalized interval $\overline{\Omega} = [0, 1]$ rather than the physical one Ω. The grid object contains all information about mesh points at which states, controls and constraints will be associated, the interpolations and quadrature methods for the unknowns, the solution procedure to be used for that arc (either transcription or multiple shooting), and the integration type (explicit or implicit) and specific scheme. The association of the discrete data (states and controls) with the grids and its evaluation at each point of interest is handled by means of an interpolation tool, which is based on the open-source GSL library [10]. Mesh points can be distributed uniformly, interactively through a graphical interface, or they can be generated automatically by a mesh refinement module. In the case of direct transcription, refinement is simply handled by repeating each step with a higher order scheme, and splitting that step if the percent difference between the original and new solutions is higher than a given threshold. Alternatively, simple uniform bisection of all time elements is

also available.

For flexibility and generality of use in an engineering context, several user-defined routines are available to define constraints or terms of the objective function. The routines have access to the problem unknowns and may or may not interact with the vehicle model, according to necessity. In the case of the multiple shooting approach, the routines have not only access to the state variables at the boundaries of the shooting segments, but also to the discrete time histories $\boldsymbol{x}_i^m$ generated by forward time marching. This allows for the evaluation of segment-internal quantities, or for integral constraints or integral terms of the objective through interpolation of the data points and the use of an appropriate quadrature rule.

Special care has been put into the definition of general purpose user-defined constraint routines which allow for the efficient exploitation of the problem sparsity, which is critical for efficiency. To handle this, it was decided to provide separate routines for the various types of constraints; currently, the constraint types which are supported by TOP are point-wise state and control constraints, control rate constraints, and integral state and control constraints.

When an optimization run is completed successfully, a refinement strategy module checks the acceptability of the solution and decides if another run is necessary or not. Finally the results are saved in ascii files for subsequent analysis, and plotted by a post-processing module.

Applications and Results

In this section we demonstrate some possible applications of the TOP program, considering both helicopter and tilt-rotor models. The choice of the examples shown here is meant to illustrate the applicability of trajectory optimization to problems in rotorcraft flight mechanics, but with no attempt at covering all possible areas or to give full and detailed descriptions and analyses of each problem.

The helicopter model represents a generic medium-size multi-engine four-bladed utility vehicle in the 9 ton class. The code-internal flight mechanics model is based on three-dimensional rigid body dynamics, where rotor forces and moments are computed by combining actuator disk and blade element theory, considering a uniform inflow [30]; the model also includes a power balance equation, which relates the rotor speed rate to the power available and to the main and tail rotor torques. The rotor attitude is evaluated

by means of quasi-steady flapping dynamics with a linear aerodynamic damping correction. Look-up tables are used for the quasi-steady aerodynamic coefficients of the vehicle lifting surfaces, and simple corrections for compressibility effects and for the downwash angle at the tail due to the main rotor are included in the model. The controls are defined as $\boldsymbol{u} = (\theta_{0_{MR}}, \theta_{0_{TR}}, A_1, B_1, P)^T$, where $\theta_{0_{MR}}$ is the main rotor collective, $\theta_{0_{TR}}$ is the tail rotor collective, A_1, B_1 are the lateral and longitudinal cyclics, respectively, and finally P is the power available. The FLIGHTLAB model is based on three-dimensional rigid body dynamics, where rotor forces and moments are computed by using an actuator disk model with uniform inflow. In order to simulate a one engine failure condition, a torque balance equation is added to the model. Look-up tables are used for the quasi-steady aerodynamic coefficients of the vehicle lifting surfaces.

The tilt-rotor model represents the ERICA vehicle [29], implemented with the FLIGHTLAB code. It is based on three-dimensional rigid body dynamics, with two four-bladed gimballed rotors with Peters-He dynamic inflow. Control inputs are the same as the helicopter model without power control. Here again look-up tables are used for the quasi-steady aerodynamic coefficients of the vehicle lifting surfaces.

Minimum time turn

We consider a minimum time turn, with cost function (4) taking the form

$$J = T + \frac{1}{T}\int_0^T \dot{\boldsymbol{u}} \cdot \boldsymbol{W}\dot{\boldsymbol{u}}\,\mathrm{d}t. \tag{19}$$

The first term enforces the minimum time condition, while $\boldsymbol{W} = \mathrm{diag}(\boldsymbol{w}_{\dot{u}})$ is a diagonal matrix of tunable weighting factors which penalize the control rates.

At first, we demonstrate the use of refinement procedures, both in the context of direct transcription and direct multiple shooting. To this end, we consider the FLIGHTLAB helicopter model performing a 90-deg turn, starting and returning to straight and level trimmed flight at 50 m/sec. Throughout the maneuver, we set a bound of $\pm$20 deg/sec for the body attached components of the angular velocity, to highlight the importance and the treatment of state constraints.

In Fig. 2 we show the time history of the roll rate obtained with the direct transcription method. The top diagram refers to the solution obtained on an initial uniform grid of 24 steps, while the bottom one presents the solution computed after 4 adaptive refinement iterations, which led to a grid having a total of

117 steps.

As previously mentioned, refinement was obtained by repeating each time step. More precisely, in this case transcription was performed using the implicit mid point rule, which has no step-internal state stages and uses a centered mid-step value of the control inputs. Nodal values of the controls were first obtained by recovery of the computed mid-step values, and then linearly interpolated within each time step. Next, using the re-interpolated control inputs, each step was repeated starting from the initial conditions provided by the solution at the left node of each time step, using a fourth-order Runge-Kutta scheme. The percent difference between the original solution at the right node of each time step and the newly obtained one was assumed as a local error indicator to drive time step refinement by simple bisection. Figure 3 shows the initial grid and a representation of its evolution throughout the local refinement iterations. Comparing this figure with the solution, it is clear that local refinement clusters the time steps where there are rapid variations in the helicopter response.

It appears that the overall behavior of the solution is reasonably captured also on the initial grid, although the local details are clearly different. The optimum time changes from the value $T^0_{\rm opt} = 12.0$ sec obtained on the initial grid to the value $T^4_{\rm opt} = 11.3$ sec of the final one. These results are somewhat typical of what can be achieved with direct transcription: reasonable answers can be obtained quickly even on rather coarse and simple grids, although, if one is interested in the details of the solution, then adaptive refinement must be used in order to cluster the time steps in the areas of sharp gradients, so as to achieve a greater accuracy while containing the computational cost. Note that in all cases, state constraints are straightforwardly enforced and rigorously satisfied by the solution.

Next, we solve the same problem using direct multiple shooting, covering the computational domain with 16 shooting arcs. The time history of the computed roll rate is shown in Fig. 4. It appears that the roll rate bounds are exceeded within some of the shooting arcs, as clearly shown in the zoomed detail of the plot reported in the bottom part of the same figure. In fact, as previously noticed, the state time history computed within the shooting arcs can not be constrained during optimization. On the other hand, the states variables at the arc interfaces, shown in the figure using $\circ$ symbols, do satisfy the required bounds, as shown in the figure, since they appear explicitly among the variables of the discrete optimization problem.

Here again, local adaptive refinement can help in improving the quality of the solution, in this case by

reducing the effect of the local constraint violations. Figure 5 shows the time history of roll rate after 2 refinement steps, which led to a total of 26 shooting arcs. In this case, the refinement strategy was based on checking if there were violations within each arc, and splitting the arc in two in case violations of the constraints were detected. It appears from the figure that violations are much reduced in this case, and the states constraints can be considered satisfied to a reasonable engineering tolerance. The optimum maneuver time was found to be $T_{\text{opt}} = 11.7$ sec, in reasonable agreement with the one computed using direct transcription.

In terms of computational cost, we have noticed that, when one considers rather coarse discretizations and loose accuracy, the direct multiple shooting strategy is typically significantly faster than multiple shooting. For example, but with no aim at giving general indications, in the present problem the ratio between the cpu time for the two methods on the initial respective grids was close to four in favor of direct transcription, although we have observed cases were this ratio can grow up to about ten. On the other hand, when more accurate solutions are required and the refinement procedures are activated, this difference tends to reduce, because the computational cost grows more rapidly in the case of direct transcription than in the case of direct multiple shooting, as previously explained.

Having highlighted some important characteristics of the two methods with the help of the 90-deg turn problem, we demonstrate the application of the proposed procedures to different vehicle models, namely helicopters and tilt-rotors. To this end, we consider a minimum time 180-deg turn, with the same cost function. In this case, the initial and final conditions correspond to a straight and level flight at 5 m/sec. The problem definition is complemented by suitable bounds on the vehicle states, controls and control rates, and is solved using the direct transcription method with 80 time steps.

Figure 6 shows the trajectory flown by the helicopter while Fig. 7 reports the trajectory of the tilt-rotor. As expected, given the different flight mechanics characteristics of the two vehicles, the same problem definition leads to two strikingly different optimal turns. In the helicopter case, the vehicle exhibits a rather standard behavior: it banks and turns and, after a slight altitude increase midway throughout the maneuver, eventually reaches the arrival trim condition about twenty meters to the right of the starting point. In the tilt-rotor case on the other hand, the vehicle goes through a flare to slow down, turns and eventually climbs with a very large side-slip angle, while loosing about fifteen meters in altitude midway

throughout the maneuver. The maximum lateral displacement is still of about twenty meters, but the arrival trim is located in this case only about five meters to the right of the starting one.

It is clear that the present solutions are only illustrative, and might not be desirable in a real operative situation. For example, the very pronounced side-slipping in the tilt-rotor case raises important visibility issues for the pilot, together with the rather significant altitude loss. On the other hand, it is clear that the maneuver could be "designed" in any desirable way, by simply enforcing suitable constraints, such as for example by limiting the side-slip angle so as not to exceed certain given values.

Category-A continued take-off after engine failure

We consider the vertical take-off from a confined area of a multi-engine helicopter under Category-A certification requirements, as described in Ref. [1]. The optimization cost function (4) in this case takes the form

$$J = h(T) + \frac{1}{T}\int_0^T \dot{\boldsymbol{u}} \cdot \boldsymbol{W}\dot{\boldsymbol{u}}\,\mathrm{d}t, \tag{20}$$

where $h(T)$ is the altitude loss at the lowest point in the maneuver, measured from the initial trimmed hovering condition. The maneuver definition is completed by the exit conditions, which specify null vertical velocity and pitch rate for $t = T$, and by bounds on the vehicle control inputs and their rates throughout the maneuver. This single-phase formulation of the problem considers only the first part of the maneuver, i.e. from the engine loss to the moment the lowest point in the trajectory is reached. A multi-phase formulation of the same problem covering also the climb part of the Category-A maneuver was considered in Refs. [13–15].

The problem was solved with the internal vehicle model, using the multiple shooting solution procedure. At first, we imposed a planar solution case, similarly to what done in Refs. [13, 14], by constraining the vehicle trajectory to be contained in a plane and by imposing a zero side-slip condition. Note that, in doing so, the equilibrium of the vehicle is still computed using the three-dimensional model, and therefore accounting for all force and moment components generated by the main and tail rotors and by the other aerodynamic surfaces of the vehicle.

The resulting control time histories are given in Fig. 8. Notice how, as expected, an increase in col-

lective is strictly correlated with corrections in the lateral cyclic and the pedal input, while a quick stick-forward input is applied in the initial phase of the maneuver to promote a fast forward acceleration of the vehicle to gain speed and hence reduce the required power.

Next, the problem was solved again, eliminating the planar solution constraints, all other problem parameters and definitions remaining the same. Figure 9 gives two views of the resulting trajectory in this case, which show a pronounced yaw of the vehicle in the first phases of the maneuver. It appears that this strategy has the effect of somewhat improving the cost function. In fact, the altitude loss was in this case of 11.6 m, compared to the 10.5 m of the planar case. This is not surprising, since helicopters are not symmetric vehicles and do not have a symmetric behavior. Here again a pronounced side-slipping might not always be desirable because of pilot and co-pilot visibility considerations or other issues. Nonetheless, the indication that better performance can be obtained by abandoning a strictly symmetric behavior can be of some interest, and has in fact been confirmed by test pilots.

ADS-33 mission task elements as optimal control problems

The ADS-33 specification [2] for military rotorcraft defines a series of Mission Task Elements (MTEs) which provide a basis for an overall assessment of a rotorcraft ability to perform certain critical tasks, and result in an assigned level of handling qualities according to the Cooper-Harper rating scale. Each MTE is related to a specific maneuver that shall be accomplished considering specific constraints, as described in Ref. [2]. In fact, it is possible to formulate each MTE as a constrained optimal control problem [17]. Hence, with a software implementation of the procedures discussed in this work, it is possible to readily compile a library of MTEs of interest in order to predict the handling qualities characteristics of a specific rotorcraft and work in parallel with the flight test trials.

To illustrate the possible use of trajectory optimization procedures in the context of ADS-33 handling qualities analysis, we consider here the simulation of two MTEs, namely the *Pirouette* and *Slalom* maneuvers. These applications are analyzed with the direct transcription method using a helicopter FLIGHTLAB model.

As described in Ref. [2], for the Pirouette MTE the helicopter should move along a reference circle of

radius $R = 100$ ft, pointing the nose towards the center of the trajectory, starting and arriving in the hover condition. The maneuver must be accomplished while satisfying constraints on the radial displacement, which must be less then ΔR $\pm$10 ft, on the heading misalignment, $\Delta\psi$ $\pm$10 deg, and with a total duration time not to exceed 45 sec. The maneuver is performed at an altitude of 10 ft ($\pm$10 ft), so that ground effects are not negligible and therefore were included in the FLIGHTLAB model.

One possible formulation of this MTE is to consider the minimum time cost function (19), and constrain the vehicle trajectory so as to express the MTE path requirements described above. With this formulation of the problem the time constraint is not explicitly enforced, but verified a posteriori. In other words, one tunes the weight parameters $\boldsymbol{W}$ in the merit function (19), this way controlling the aggressiveness of the maneuver. Then, once a solution has been computed, one verifies whether the maneuver was rapid enough and effectively completed within the maximum allotted time. Obviously there are limitations in the maneuver aggressiveness related to the vehicle capabilities and its flight envelope constraints.

The typical execution of the Pirouette maneuver can be divided into three phases. The first part is characterized by a transition from hover to a steady turn in lateral flight. The opposite transition is accomplished at the end of the maneuver when the hover state must be recovered. The cental part of the maneuver is characterized by a steady turn with an appropriate angular velocity. The simulation strategy reflects this description, and it is effectively the result of two optimized maneuvers with a trim condition interposed between them. The first and last transitions are formulated in exactly the same manner, except for the boundary conditions which are inverted; for the internal phase a steady turn in lateral flight at the velocity of 11 knots is assumed. Each transition is characterized by a Chebychev grid of 50 time elements and, according to the ADS-33 specification, the following additional constraints are enforced at each grid node:

$$|r(t_i) - R| \leq |\Delta R|, \quad |\psi(t_i) - \gamma(t_i)| \leq |\Delta\psi|, \quad i = 0 \ldots N, \tag{21}$$

where $r(t) \triangleq \sqrt{x(t)^2 + y(t)^2}$ and $\gamma(t) \triangleq tan^{-1}(y(t)/x(t))$, if x-y is the plane of the reference circle centered in the origin.

For the first phase of the MTE, the final position and heading angle are not imposed directly, but additional constraints are imposed so that at the end of the maneuver the vehicle is located in an unknown

point along the reference circle pointing its nose toward the center. The opposite approach is used for the last transition, where the initial position is unknown and constrained in an analogous way. Figure 10 shows the trajectory flown with snapshots of the vehicle. Lastly, Fig. 11 shows the roll and pitch angles, where one can notice the constant values corresponding to the central turning trim.

Next, we consider the Slalom MTE, shown in Fig. 12. The helicopter, starting from a stabilized forward flight condition lined up with the centerline of the test course, is supposed to turn around a series of obstacles and, finally to recover the initial steady state. The obstacles are located at 500 ft intervals and at ± 50 ft from the centerline; the obstacles should be passed with a maximum lateral error of 50 ft.

The maneuver must be accomplished below the reference altitude of 100 ft, so that, also in this case, ground effects must be included in the model. Furthermore, the maneuver definition does not assign a maximum time as in the previous case, but rather specifies a minimum speed of at least 60 knots throughout the slalom.

The simulation is characterized by a uniform grid of 100 steps and three obstacles. In order to introduce the presence of the obstacles, the bounds of Table 1 are enforced for the ith pylon, where x_j and y_j represent the position (in the x-y plane) at the jth mesh node.

Table 1. Slalom MTE: position bounds for the obstacles

	Lower	**Upper**
x_{1+i20}	i 500 ft	i 500 ft
y_{1+i20}	$-35+(-1)^i$ 75 ft	$+35+(-1)^i$ 75 ft

A specific lower bound is imposed for the flight speed in order to guarantee the satisfaction of the lower value of 60 knots imposed by the ADS-33 specification. The minimum time cost function is used again to reach a proper level of aggressiveness. Figure 13 reports the optimal control time histories; the

oscillations of the longitudinal and lateral cyclics is clearly related to the alternating left and right turns.

Conclusions

In this document we have described numerical procedures for the solution of trajectory optimization problems in rotorcraft flight mechanics, and their implementation in a general purpose software tool. The problem was cast within the framework of optimal control theory. Using this approach, each maneuver is viewed as the solution of an appropriate constrained minimization problem; by specifying cost function and constraints, one gives a mathematical definition of the maneuver which is then computed by solving the resulting optimal control problem.

Within this class of problems, we have discussed the issue of interfaceability to black-box vehicle simulators, identifying in the direct approach to the solution of optimal control problems the method of choice under this constraint. This enables the use of third-party vehicle models, which in turn is important for avoiding duplication of modeling and validation efforts, for reducing data incompatibility among different models, etc.

Among the direct solution strategies, we have considered the direct transcription and the direct multiple shooting methods, both having been implemented in the code. The experience gathered so far has lead to the following conclusions on the characteristics of these two methods:

- The direct transcription method is typically more efficient in terms of required computational resources, as long as one can use fairly large time steps using implicit numerical schemes. Previous experience has shown that the vehicle states converge quite rapidly with the mesh size, so that good results are obtained even on relatively coarse grids. Although control inputs converge more slowly and hence would require denser grids for convergence, these quantities are typically of a lesser interest to the analyst.

  The rigorous treatment of state inequality constraints, that for example are used for imposing the respect of flight envelope boundaries or other procedural and maneuver-defining constraints, is handled in a straightforward manner.

  The resulting numerical method is typically quite robust since the two-point boundary value treat-

ment of the problem can cope naturally with unstable systems as helicopters; furthermore, the method is not very sensitive to the initial guess, especially if suitable boot-strapping procedures from crude meshes are used [13].

If one has access to the model governing equations or their software implementation, the NLP problem Jacobian can be evaluated analytically or using automatic differentiation or symbolic manipulation programs, with possible substantial computational savings.

On the other hand, the implementation of direct transcription requires a level of interaction with the vehicle model which might not be available with all third-party flight simulators; contrary to this statement, we were able to implement this approach with the codes FLIGHTLAB and EUROPA.

- The direct multiple shooting method is easily interfaced with any third-party software which allows one to set the vehicle initial conditions, to integrate the model forward in time on a given temporal window under the action of given control inputs, and to gather the results of the integration; these seem to be minimum functionalities, which should be available in all general purpose simulators.

  Furthermore, the NLP problem size does not grow if the time step size for the forward time marching is reduced, as it is the case for the direct transcription approach. This may enable the use of explicit schemes instead of the implicit ones necessary for the direct transcription approach, but more importantly it makes it possible to use at affordable computational costs more complex vehicle models which, capturing fine-scale higher-frequency phenomena, require smaller time step sizes for resolving the physics of the model.

  On the other hand, the method might suffer when analyzing unstable systems, an ad hoc but rather heuristic cure being the split of the temporal domain in a suitable number of shooting segments.

  On the negative side, one should also mention the fact that the method can not deal directly with state inequality constraints, a possible solution being the use of adaptive refinement of the shooting arcs illustrated above.

The case of models of high complexity and elevated cost per physical time step was not covered in this work, but can be handled using the planning and steering algorithms described in Refs. [15, 16]. Notice

that a central component of those strategies is represented by the planning layer, which amounts to solving the very same maneuver optimal control problems described here.

Acknowledgments

The present research is supported by Agusta-Westland through a grant with the Politecnico di Milano (Marzio Preatoni and Marco Cicalè project monitors). The development of the interface with FLIGHTLAB and EUROPA was conducted at the Agusta-Westland headquarters in Cascina Costa, Italy, using Agusta-Westland licenses. The contribution of Andrea Ragazzi of Agusta-Westland is gratefully acknowledged. Further support is provided by the US Army Research Office, through a grant with the Georgia Institute of Technology and a sub-contract with the Politecnico di Milano, with Dr. Gary Anderson as technical monitor. The software described in this document is an evolution of the code described in Refs. [13, 14], and implemented by Domenico Leonello and Luca Riviello, whose valuable help in the development of the present version of the code is here gratefully acknowledged.

References

[1]*Advisory Circular 29-2C, Certification of Transport Category Rotorcraft*, Federal Aviation Administration, Department of Transportation, 1999.

[2]*Handling Qualities Requirements for Military Rotorcraft*, Aeronautical Design Standard, U.S. Army Aviation and Missile Command, Aviation Engineering Directorate, Rept. ADS-33E-PRF, Redstone Arsenal, AL, 2000.

[3]Advanced Rotorcraft Technology, Inc., 1685 Plymouth Street, Suite 250, Mountain View, CA 94043, http://www.flightlab.com.

[4]BRITE-EURAM, 2000, RESPECT, Rotorcraft Efficient and Safe Procedures for Critical Trajectories.

[5]Ascher, U.M., Mattheij, R.M.M. and Russell, R.D., *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Classics in Applied Mathematics, 13, SIAM, Philadelphia, 1995.

[6]Bauchau, O.A., Bottasso, C.L. and Nikishkov, Y.G., "Modeling Rotorcraft Dynamics with Finite Element Multibody Procedures," *Mathematics and Computer Modeling*, Vol. 33, 2001, pp. 1113–1137.

[7]Barclay, A., Gill, P.E. and Rosen, J.B., "SQP Methods and Their Application to Numerical Optimal Control," Report NA 97–3, Department of Mathematics, University of California, San Diego, CA, 1997.

[8]Betts, J.T., *Practical Methods for Optimal Control using Non-Linear Programming*, SIAM, Philadelphia, 2001.

[9]Betts, J.T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Controls and Dynamics*, Vol. 21(2), 1998, pp. 193–207.

[10]Booth, M., Davis, J., Galassi, M., Gough, B., Jungman, G., Rossi, F. and Theiler, J., *GNU Scientific Library Reference Manual*, http://www.gnu.org/software/gsl/, 2006.

[11]Bhagwat, M.J. and Leishman, J.G., "Stability, Consistency and Convergence of Time Marching Free-Vortex Rotor Wake Algorithms," *Journal of the American Helicopter Society*, Vol. 46, 2001, pp. 59–71.

[12]Bottasso, C.L. and Croce, A., "Optimal Control of Multibody Systems using an Energy Preserving Direct Transcription Method," *Multibody Systems Dynamics*, Vol. 12, 2004, pp. 17–45.

[13]Bottasso, C.L., Croce, A., Leonello, D. and Riviello, L., "Optimization of Critical Trajectories for Rotorcraft Vehicles," *Journal of the American Helicopter Society*, Vol. 50, 2005, pp. 165–177.

[14]Bottasso, C.L., Croce, A., Leonello, D. and Riviello, L., "Rotorcraft Trajectory Optimization with Realizability Considerations," *Journal of Aerospace Engineering*, Vol. 18, 2005, pp. 146–155.

[15]Bottasso, C.L., Chang, C.-S., Croce, A., Leonello, D. and Riviello, L. , "Adaptive Planning and Tracking of Trajectories for the Simulation of Maneuvers with Multibody Models," *Computer Methods in Applied Mechanics and Engineering*, Special Issue on Computational Multibody Dynamics, Vol. 195, 2006, pp. 7052–7072.

[16]Bottasso, C.L., Croce, A. and Leonello, D., "Neural-Augmented Planning and Tracking Pilots for Maneuvering Multibody Dynamics," *Multibody Dynamics. Computational Methods and Applications*, García Orden, J.C., Goicolea, J.M. and Cuadrado, J., Eds., Computational Methods in Applied Sciences, ISBN 1-4020-5683-4, Springer-Verlag, Dordrecht, The Netherlands, 2007.

[17]Bottasso, C.L., Maisano, G., Ragazzi, A. and Scorcelletti, F., "Trajectory Optimization Strategies for the Simulation of ADS-33 Mission Tasks Elements," European Rotorcraft Forum, Liverpool, UK, September 16–19, 2008.

[18]Bryson, A.E. and Ho, Y.C., *Applied Optimal Control*, Wiley, New York, 1975.

[19]Carlson, E.B. and Zhao, Y.J., “Optimal Short Takeoff of Tiltrotor Aircraft in One Engine Failure,” *Journal of Aircraft*, Vol. 39, 2002, pp. 280–289.

[20]DAtta, A. and Johnson, W.,“An Assessment of the State-of-the-Art in Multidisciplinary Aeromechanical Analyses,” Proceedings the AHS Specialists’ Conference on Aeromechanics, San Francisco, USA, January 23–25, 2008.

[21]Johnson, W., “CAMRAD/JA: A Comprehensive Analytical Model of Rotorcraft Aerodynamics and Dynamics, Volume I: Theory Manual,” Johnson Aeronautics, 1988.

[22]Frazzoli, E., “Robust Hybrid Control for Autonomous Vehicle Motion Planning,” Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001.

[23]Geradin, M. and Cardona, A., *Flexible Multibody Dynamics, a Finite Element Approach*, John Wiley & Sons, New York, NY, 2000.

[24]Gill, P.E., Murray, W. and Wright, M.H., *Practical Optimization*, Academic Press, London and New York, 1981.

[25]Griewank, A., Juedes, D., Mitev, H., Utke, J., Vogel, O. and Walther, A., “ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++,” ACM TOMS, Vol. 22(2), 1996, pp. 131–167.

[26]Howlett, J.J., “UH-60A Black Hawk Engineering Simulation Program: Volume I - Mathematical Model,” NASA Contractor Report 166309, National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California, USA, 1981.

[27]Hull, D.G., “Conversion of Optimal Control Problems into Parameter Optimization Problems,” *Journal of Guidance, Control and Dynamics*, Vol. 20, 1997, pp. 57–60.

[28]Okuno, Y. and Kawachi, K., “Optimal Takeoff Procedures for a Transport Category Tiltrotor,” *Journal of Aircraft*, Vol. 30, 1993, pp. 291–292.

[29]Nannoni, F., Giancamilli, G. and Cicalè, M., “ERICA: the European Advanced Tiltrotor,” 27th European Rotorcraft Forum, September 11–14, 2001.

[30]Prouty, R.W., *Helicopter Performance, Stability, and Control*, R.E. Krieger Publishing Co., Malabar, 1990.

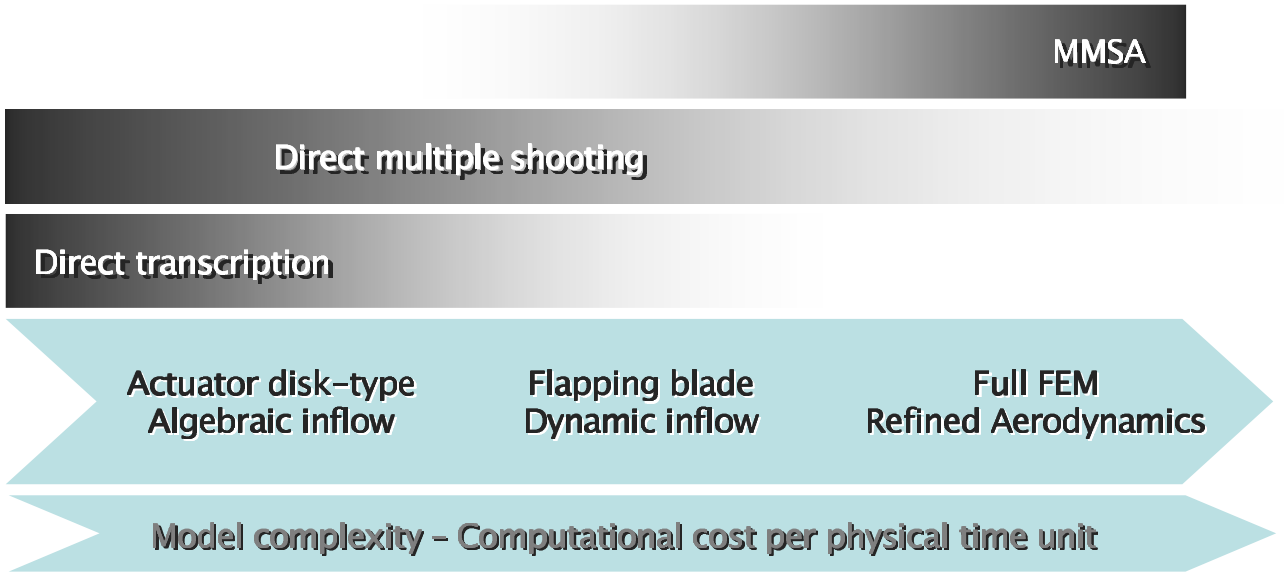

Fig. 1 Preferred trajectory optimization methods for rotorcraft models of increasing complexity as measured by the computational cost of advancing the solution of one physical time unit.

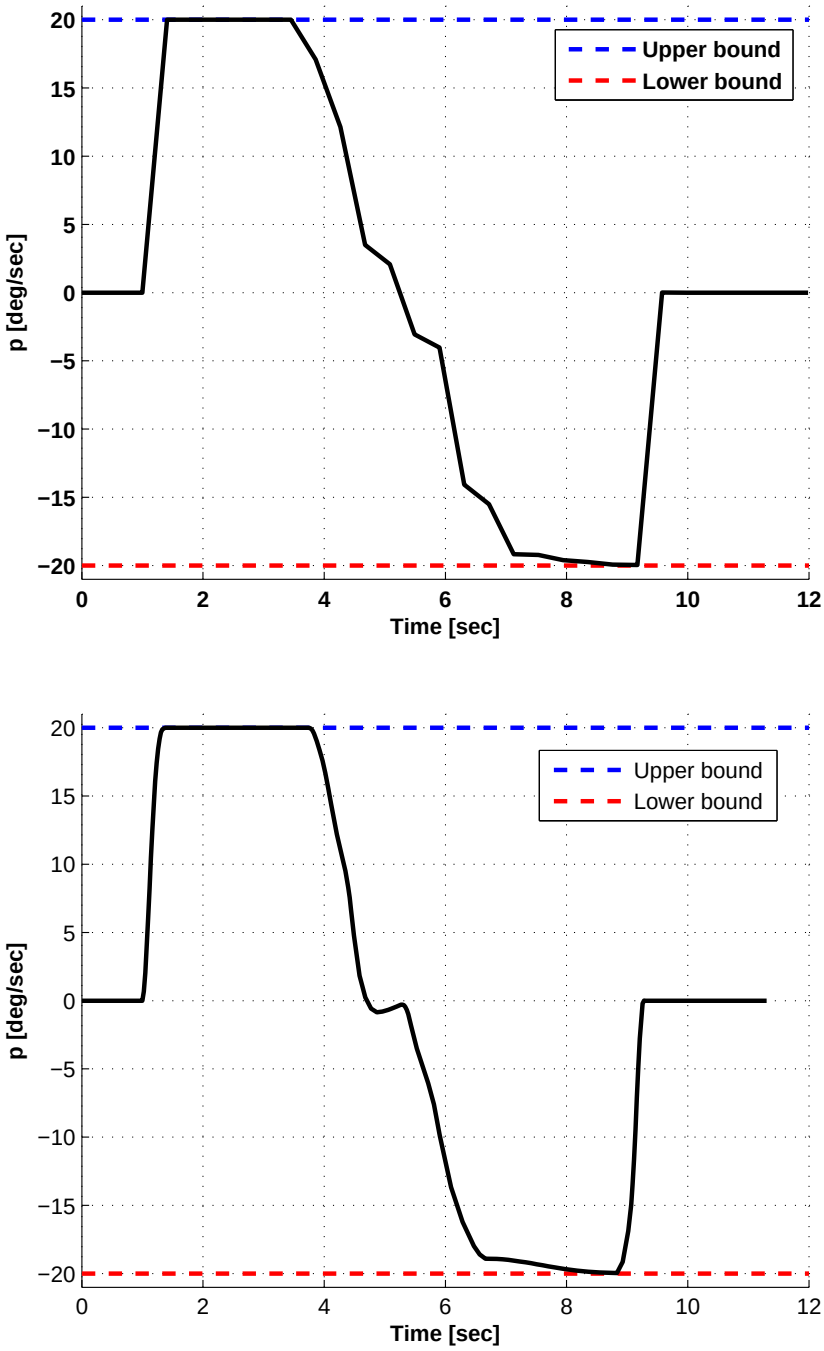

Fig. 2 Minimum time 90-deg turn, direct transcription method. Top: roll rate computed on initial uniform grid; bottom: solution for adaptively refined grid.

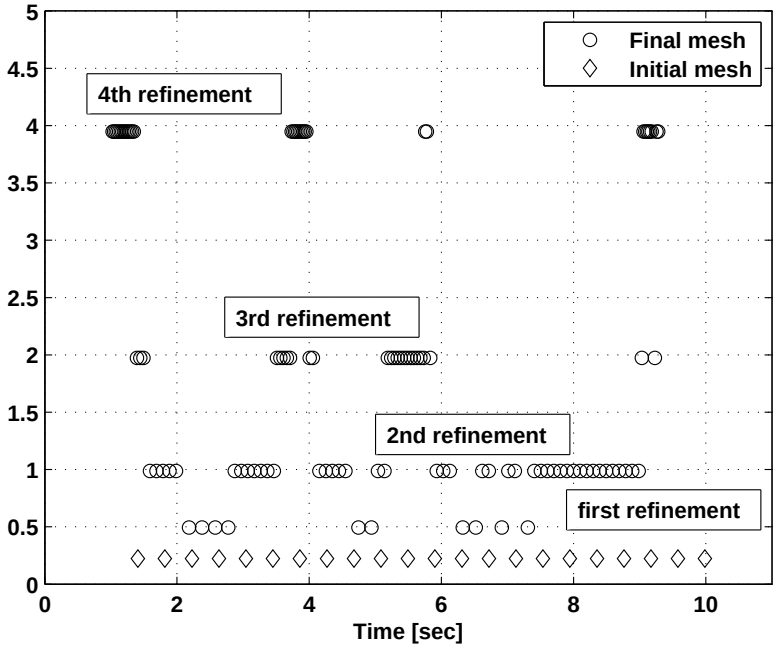

Fig. 3 Minimum time 90-deg turn, direct transcription method. Initial uniform grid and representation of the grid evolution throughout 4 local refinement iterations.

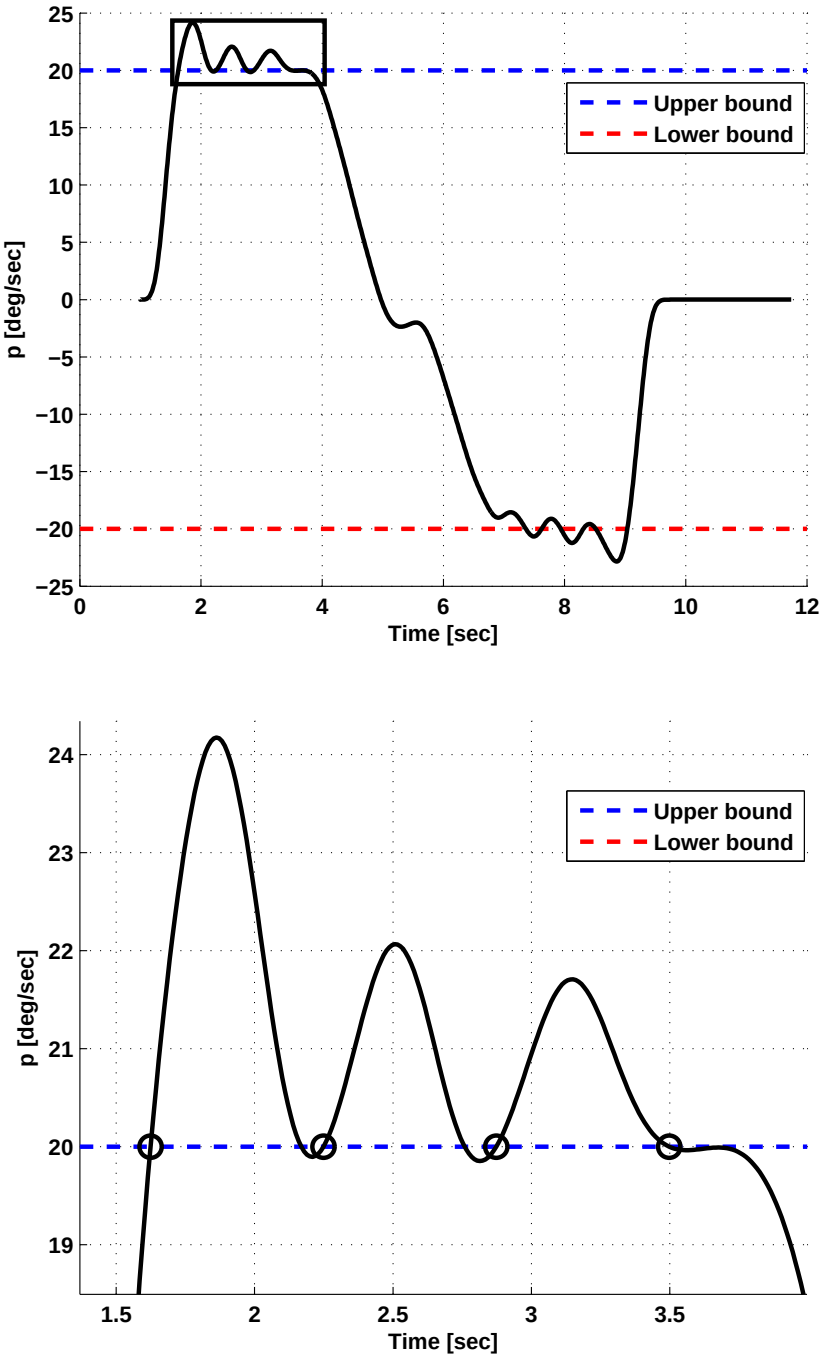

Fig. 4 Minimum time 90-deg, direct multiple shooting method. Time history of roll rate, and zoomed view in a region of local constraint violations (bottom).

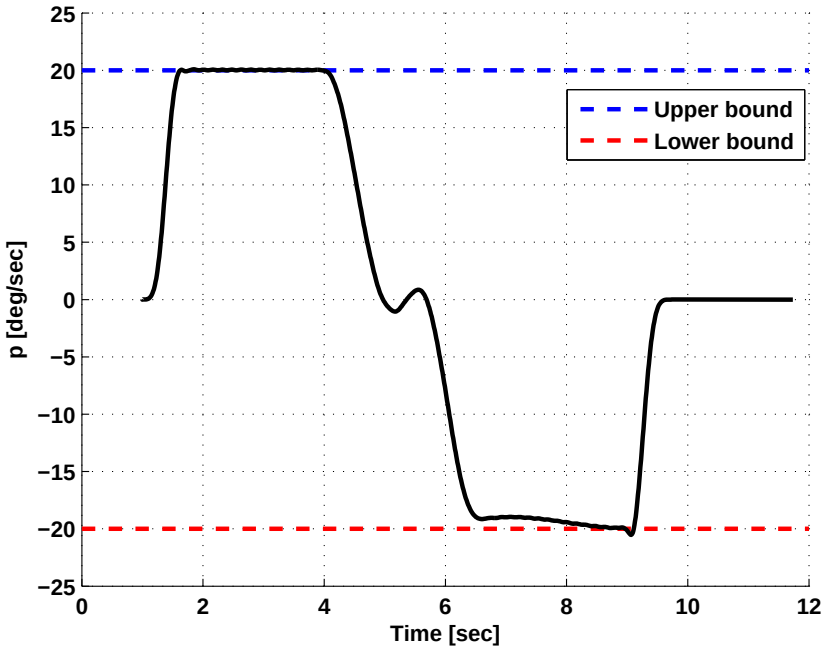

Fig. 5 Minimum time 90-deg, direct multiple shooting method. Time history of roll rate computed on the final adapted grid.

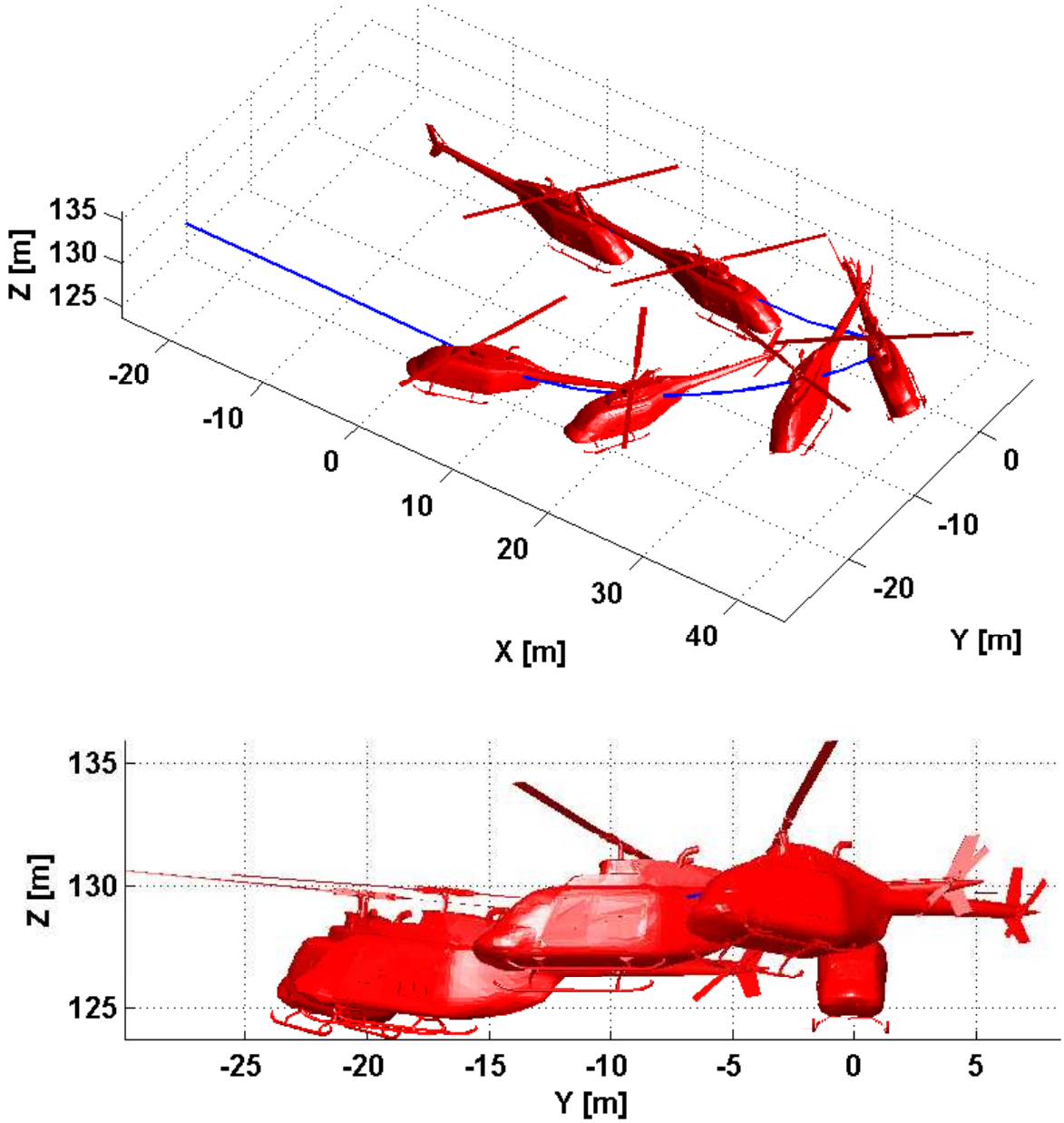

Fig. 6 Minimum time turn 180-deg: helicopter case.

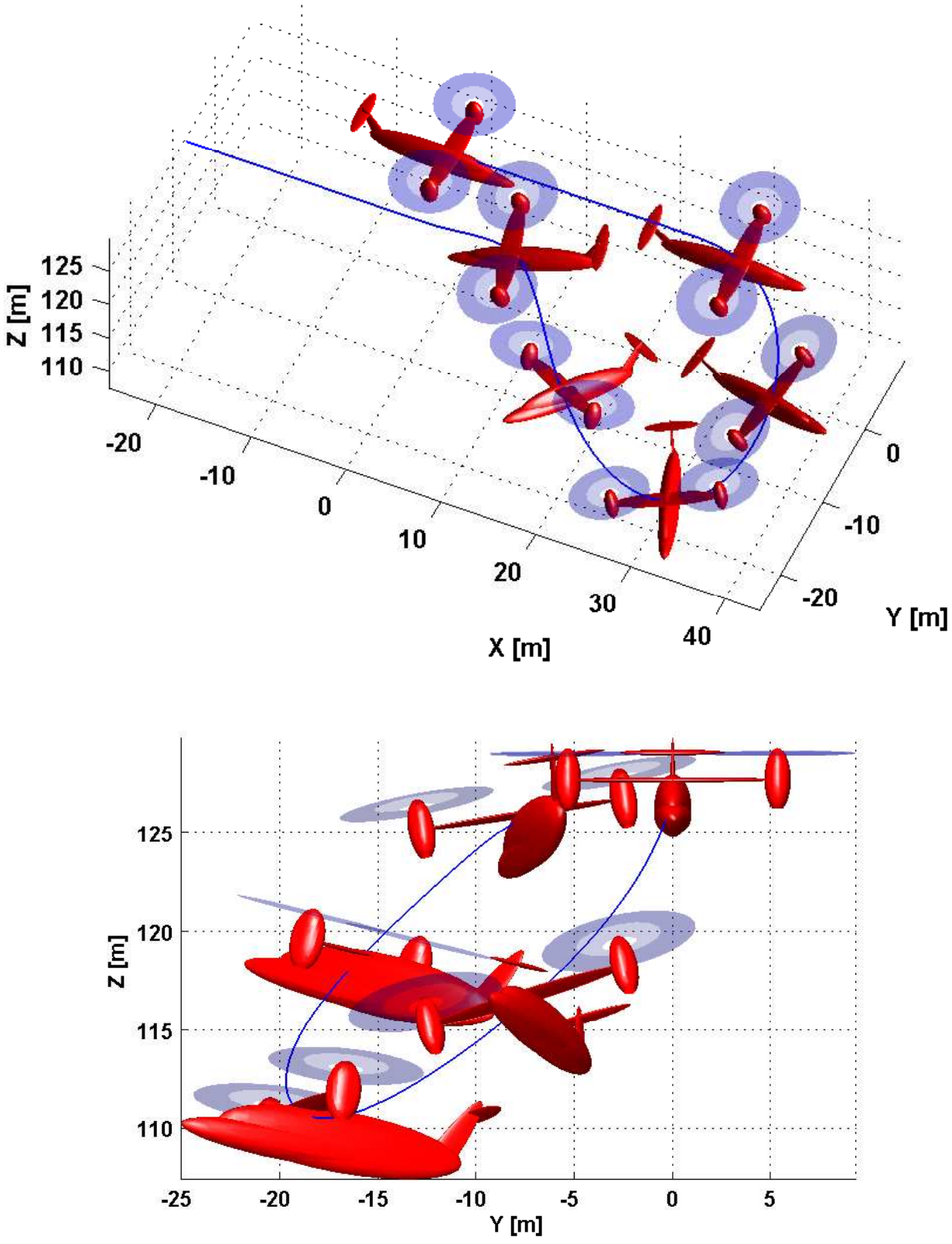

Fig. 7 Minimum time turn 180-deg: tilt-rotor case.

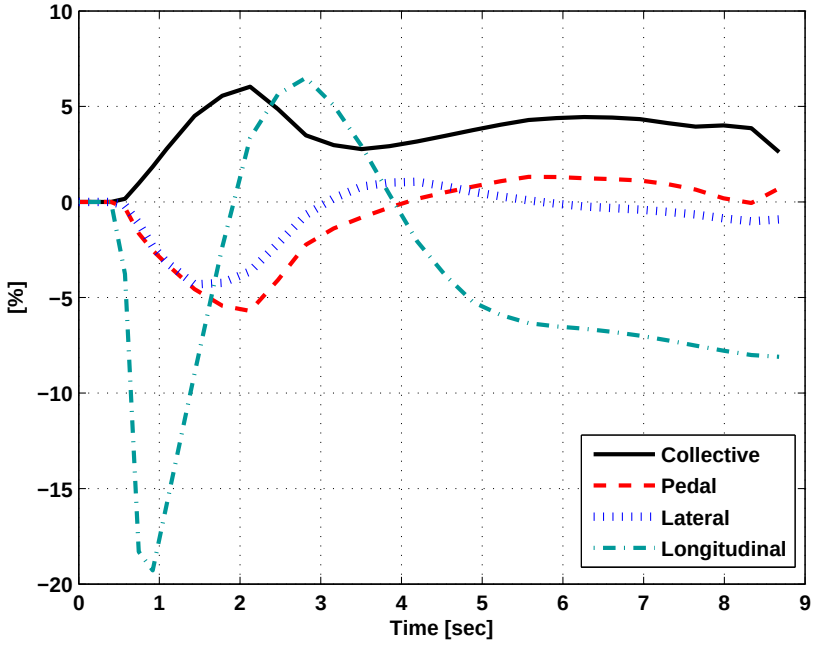

Fig. 8 Category-A continued take-off: time history of control inputs for the planar trajectory case.

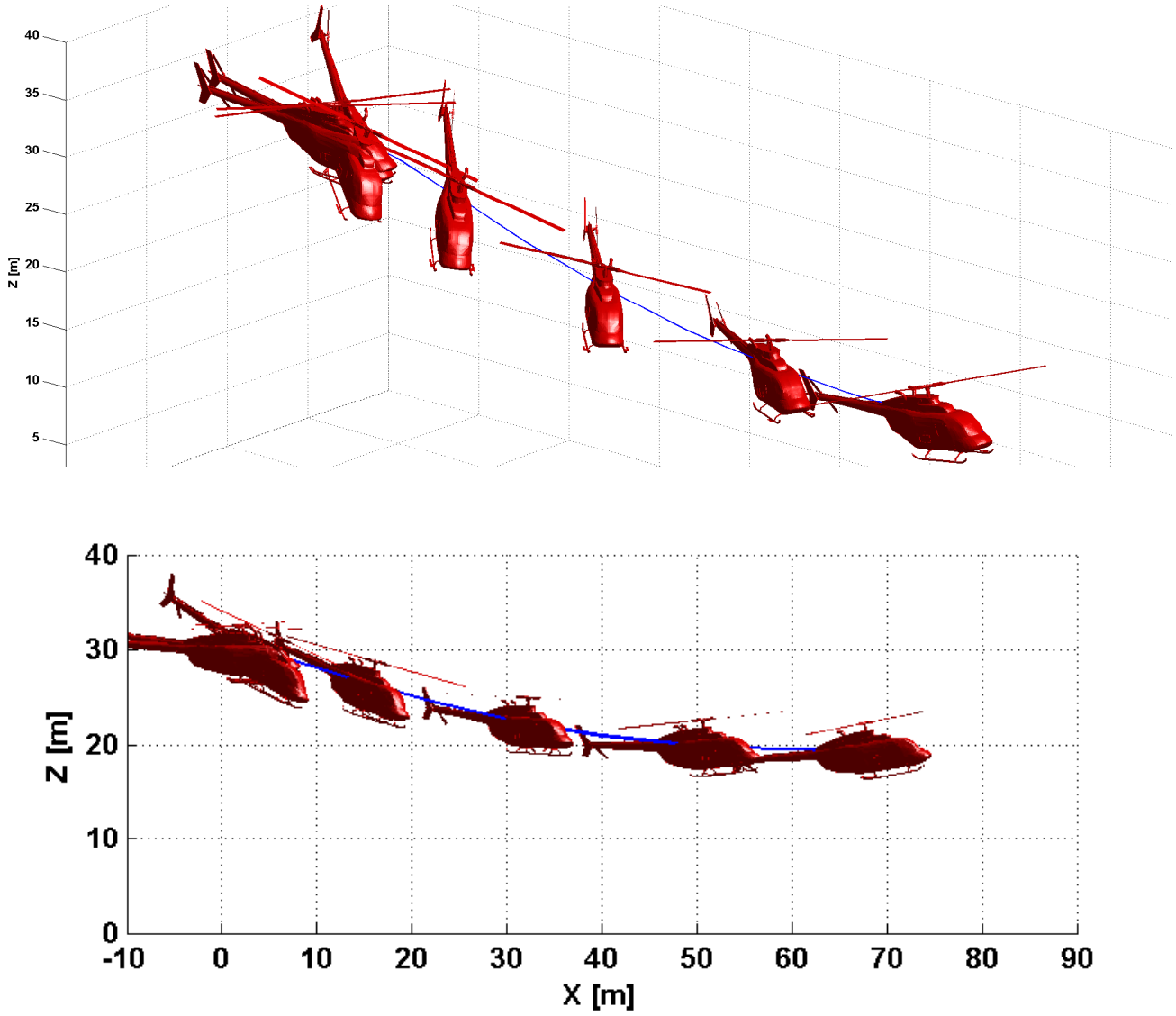

Fig. 9 Category-A continued take-off: three-dimensional trajectory case.

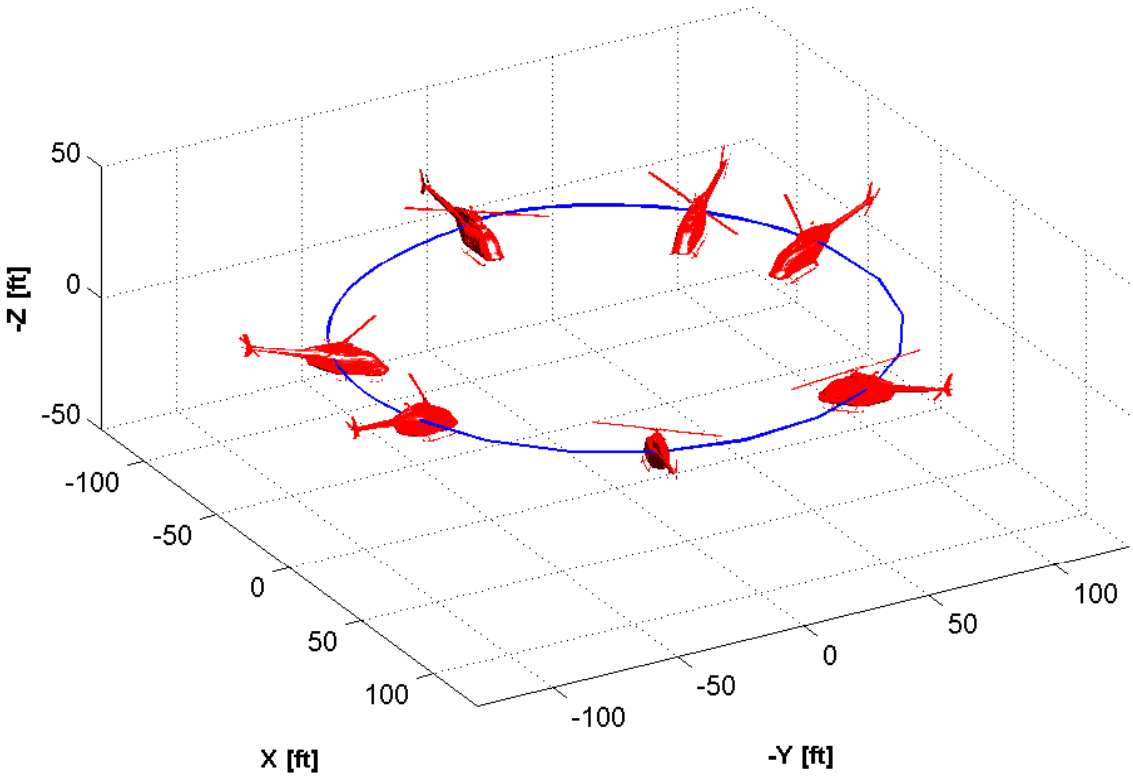

Fig. 10 Pirouette MTE: view of the trajectory.

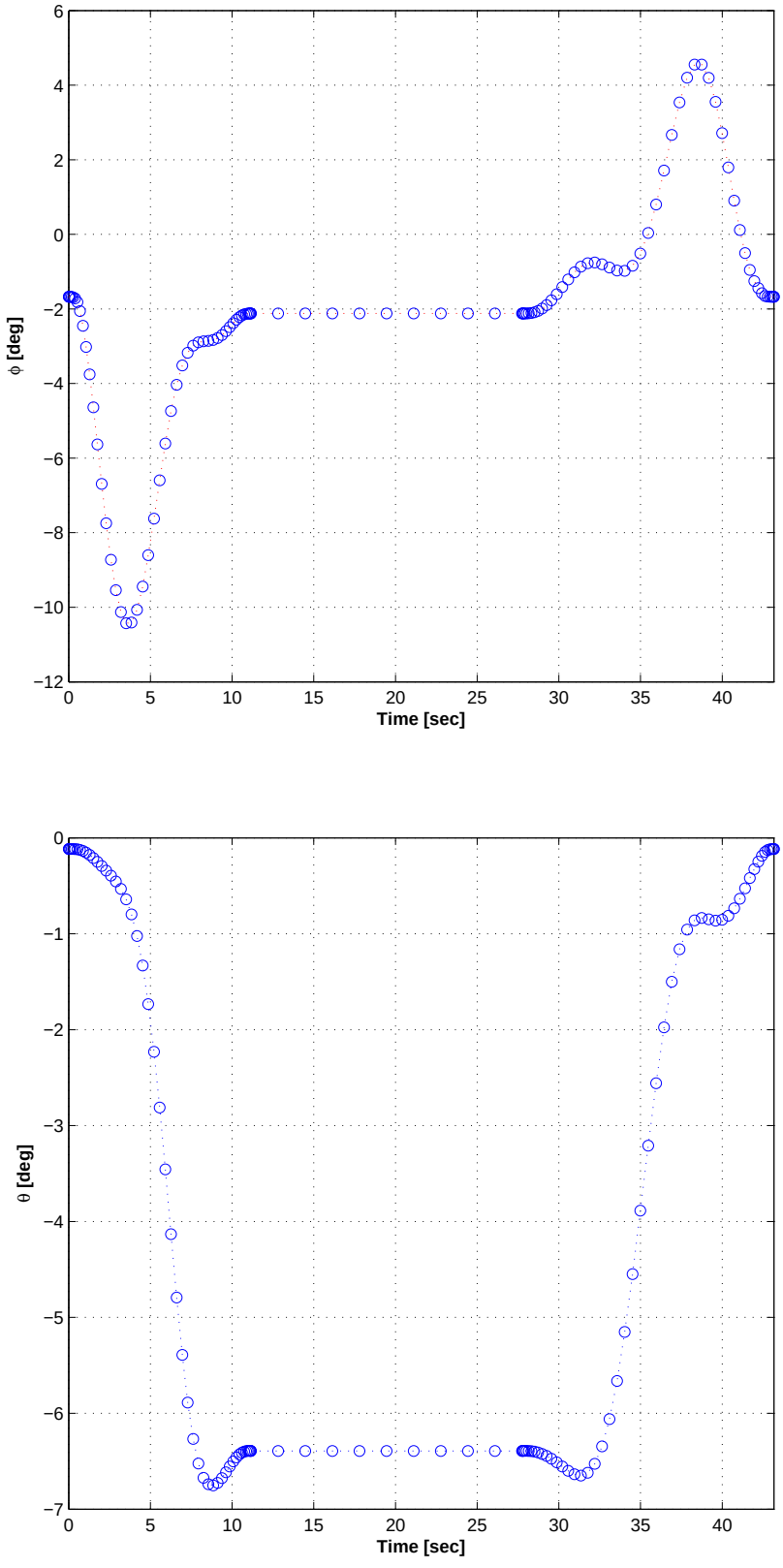

Fig. 11 Pirouette MTE. Top: roll angle; bottom: pitch angle.

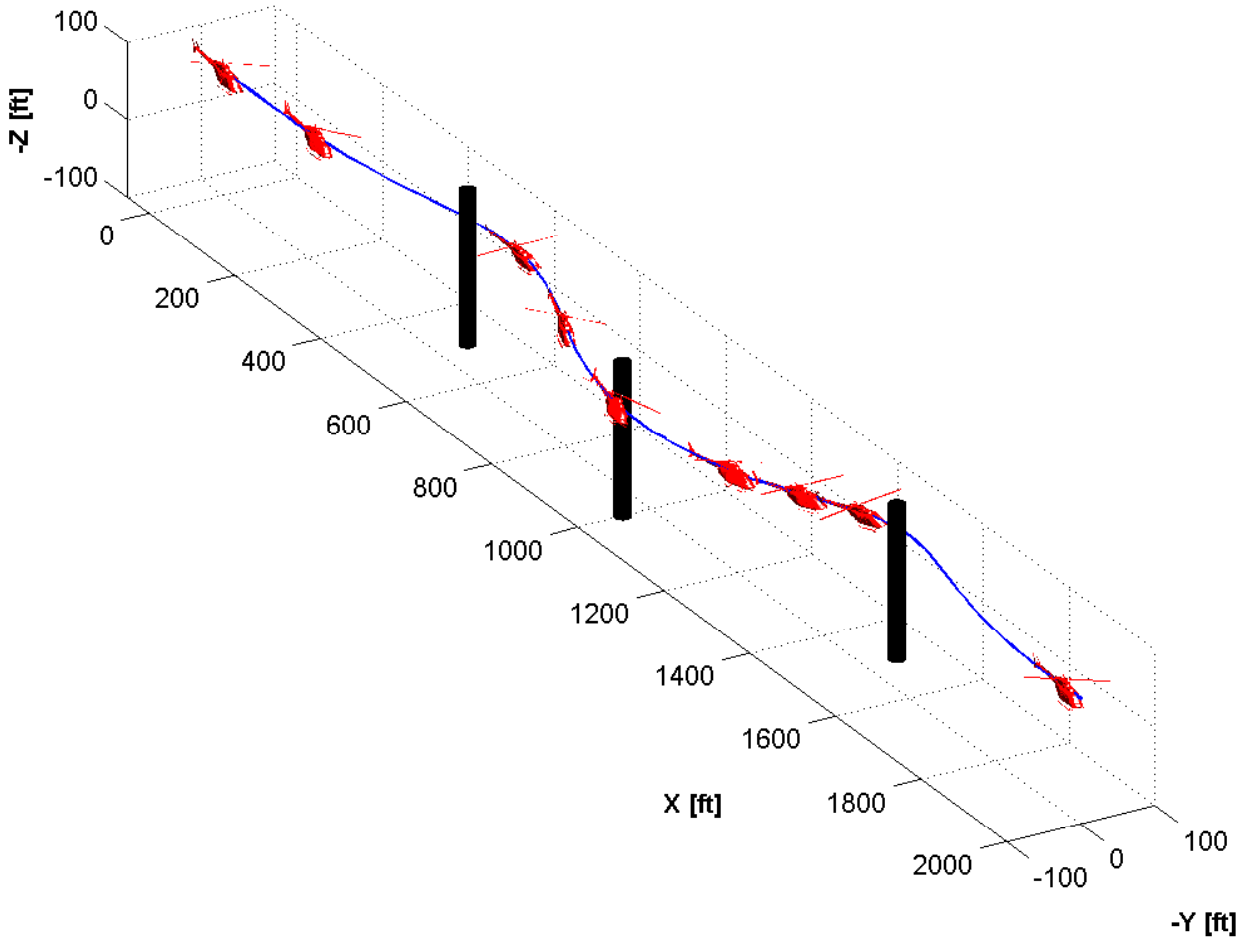

Fig. 12 Slalom MTE: view of the trajectory.

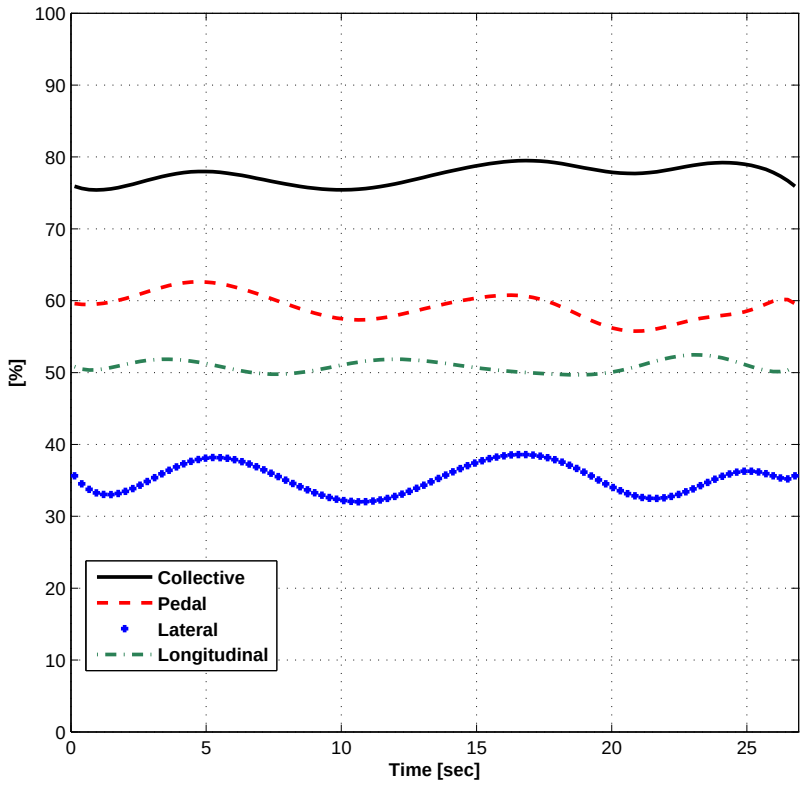

Fig. 13 Slalom MTE: time history of the control inputs.