VIATO—Visual Interactive Aircraft Trajectory Optimization

Kai Virtanen, Harri Ehtamo, Tuomas Raivio, and Raimo P. Hämäläinen, *Member, IEEE*

***Abstract*— An approach toward the automated solution of aircraft trajectory optimization problems is introduced and implemented in an interactive program called visual interactive aircraft trajectory optimization (VIATO). This MS Windows-compatible software produces minimum time trajectories to a fixed or moving target. It is easy to use by nonexperts as no previous knowledge of the methods of optimal control theory or mathematical modeling are needed. VIATO consists of a graphical user interface, an optimization server, and a model server. In VIATO, different aircraft types are represented by a set of parameters. The equations of motion and state as well as control constraints are fixed in advance. Since the objective function is also specified, the user avoids the modeling and explicit formulation of optimal control problems. Reliable convergence to an approximate optimal solution is achieved by converting the original optimal control problem into a finite dimensional optimization problem. The parameterized problem is solved using nonlinear programming.**

***Index Terms*—Aircraft control, interactive computing, nonlinear programming, optimal control.**

I. Introduction

IN this paper, a way to automatically solve aircraft trajectory optimization problems is proposed and implemented in the visual interactive aircraft trajectory optimization (VIATO) software. Its current version, operated by the research team, solves minimum time climb problems, minimum time trajectories to a fixed or a moving target on the vertical plane, and three-dimensional interception problems.

The well-known theory of optimal control (e.g., [7]) provides a framework for solving optimal trajectories. Nevertheless, in practice, generating numerical solutions is a laborious and time consuming task even for experts in system modeling and optimal control theory [4], [27], [34]. As the models are nonlinear, analytical solutions can be obtained only in some special cases (e.g., [18]). Automating the numerical solution process would make optimal trajectories easily accessible for engineers, pilots, instructors, and other relevant parties, but advances in this direction have mainly served mathematically literate researchers (see Section II).

Off-line computation of optimal trajectories has a variety of applications in different fields of aviation. In flight engineering, optimal flight paths provide a commensurable way to analyze and compare the performance of different aircraft and technologies [30]. Analysis of safe takeoff and landing in the presence of windshear [8], [25], produce valuable instructions for pilots encountering such a phenomenon. On the other hand, minimum fuel trajectories (e.g., [36]) aim at cost-effectiveness and help in reducing air traffic pollution.

Manuscript received January 5, 1998; revised July, 1998.

The authors are with the Systems Analysis Laboratory, Helsinki University of Technology, FIN-02015 HUT, Espoo, Finland (e-mail: kai.virtanen@hut.fi; harri.ehtamo@hut.fi; tuomas.raivio@hut.fi; raimo@hut.fi).

Publisher Item Identifier S 1094-6977(99)05323-7.

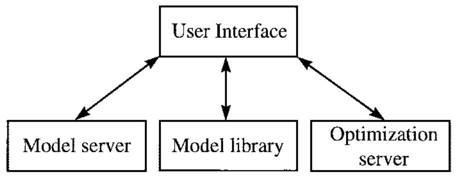


Fig. 1. Structure of VIATO software.

In military aviation, knowledge of different minimum time maneuvers is of fundamental importance (e.g., [12], [13], and [32]). This fact already guided the seminal studies [6] and is still acknowledged in pilot training and education. Analysis of optimal trajectories can also reveal new ways to complete given tasks. For example, mission planning in a hostile environment can be based on estimated risk minimization [37].

When solving a particular trajectory optimization problem, the user of VIATO needs only to choose the aircraft model type from the model library, select the control criterion, and specify the initial and terminal states of the aircraft. VIATO then creates the corresponding optimization problem and solves it automatically. After the optimization, VIATO visualizes the results and allows for sensitivity analysis of the solution with respect to different model parameters.

The underlying mathematical theory is hidden behind the user interface, and thus, pilots and other decision makers can study and compare different scenarios independently. Versatile visualization possibilities facilitate clear interpretation of results, and sensitivity analysis enhances the user's understanding of the nature of the problem. VIATO software could therefore be considered as an off-line support tool for decision making and design problems of flight.

VIATO has been implemented as a client-server application. It consists of three separate blocks: a user interface (the client), an optimization server, and a model server (see Fig. 1). It is possible to distribute the client and the servers to separate computers and exchange data through a suitable network.

The graphical user interface operates under MS Windows 3.x, 95, and NT. It is implemented with Delphi [1], which is a component-based application development environment. VIATO's mouse-driven interface has a limited set of commands which makes it easy to manage. It provides the optimization server with the required information and displays optimal trajectories graphically. From the interface, optimal solutions can be exported into other applications as well.

Aircraft of a certain type, like fighters, obey structurally similar mathematical models that differ only in parameters (e.g., [24]). In VIATO, the equations of motion and the structure of possible constraints for control and state variables are fixed in advance, and different aircraft are characterized by a set of parameters which are saved in an aircraft model library. VIATO's model server maintains the library and automatically creates continuous and smooth approximations for the thrust force and the drag coefficients. Approximations can also be formed interactively.

In VIATO's optimization server, the original optimal control problem is converted into a finite dimensional optimization problem that is solved using nonlinear programming [19], [28], [33]. For a recent review of different discretization schemes, see [21]. Current nonlinear programming methods provide reliable convergence and require moderate computational effort. Hence, the optimization can be carried out on a PC, which is a common, widespread, cost-effective, and flexible platform.

The paper is organized as follows. First, currently existing optimal control software is briefly surveyed. In Section III, the class of aircraft trajectory optimization problems, whose solution process is to be automated, is defined. VIATO's optimization server and the solution method are described in Section IV. The description of the model server is given in Section V. The graphical user interface and the use of the software are demonstrated by example runs in Section VI. In Section VII, an example of sensitivity analysis is carried out. Concluding remarks appear in Section VIII.

II. Review of Related Optimal Control Software

The nonlinear programming for direct optimization of trajectories (NPDOT) package [19] discretizes the original optimal control problem by direct collocation and uses nonlinear programming to solve the finite dimensional problem. Its successor, optimal trajectories by implicit simulation (OTIS) [20], solves aerospace problems and produces optimal point-mass trajectories for different flight vehicles. The model and the objective are specified via Fortran 77 subroutines which have to be compiled and linked with the main program by the user.

The variational trajectory optimization tool set (VTOTS) software package [5] is aimed at solving arbitrary optimal control problems. It consists of a finite element and multiple shooting algorithms for solving the necessary conditions for optimality. VTOTS uses three computer languages and compilers.

The OptiA system [11] is an interactive environment for the definition, solution, and analysis of nonlinear mathematical programming problems. For optimal control problems, OptiA uses control parameterization (e.g., [21]). Although operated through a graphical interface, the models and objectives are given as source code procedures.

The direct collocation program (DIRCOL) [34] discretizes optimal control problems using direct collocation. The resulting finite dimensional problem is solved by the NPSOL subroutine [16] which utilizes sequential quadratic programming. DIRCOL offers many sophisticated features, like the treatment of multiple-stage models, discretization error estimation, and adaptive allocation of discretization points. Also, in DIRCOL the problem is defined by Fortran 77 subroutines.

The recursive integration optimal trajectory solver (RIOTS) [31] provides an interactive environment for solving optimal control problems via Matlab [2]. The user of RIOTS supplies the objective, the constraints, and the dynamics as well as their derivatives as M-files of Matlab. RIOTS consists of three different programs that discretize original optimal control problems and solve the finite dimensional problem. It can adaptively refine the discretization mesh and compute estimates of integration errors.

Trajectory optimization by mathematical programming (TOMP) [23] is a Fortran module for optimal control calculations. It uses control parametrization. The user must write three subroutines and link them with the main program of TOMP.

The user of this software is expected to be able to formulate the optimal control problem to be solved, and thus, he or she must have expert knowledge of mathematical modeling and optimal control theory. The software requires source code modification and recompilation every time the problem is altered, which complicates the solution process and calls for some knowledge of programming. In VIATO, the problem class is fixed in advance, and the user specifies the necessary information for the problem formulation via the graphical interface.

III. Aircraft Trajectory Optimization Problems

In an optimal control problem, the objective is to find an admissible control function that transforms a dynamical system from its initial state to its final state so that the given objective is minimized, and state and control constraints are satisfied.

Identifying the state and control variables of a dynamical system and specifying the relations between them can be a difficult and complex task. It is obviously impossible to translate a real-world problem into an optimal control problem without experience in mathematical modeling.

In aircraft trajectory optimization problems, the modeling can be avoided by specifying the equations of motion and the constraints for control and state variables in advance. This can be done because structurally identical differential equations represent most point-mass aircraft models.

A. Aircraft Model

In VIATO, the dynamics of an aircraft are described by a system of differential equations that constitutes the state equation of the optimization problem. The equations of motion

TABLE I
THE PARAMETERS THAT CHARACTERIZE AN AIRCRAFT IN VIATO

Parameter		Independent variables
Maximum thrust force	T_{max}	Altitude, Mach number
Zero drag coefficient	C_{D_0}	Mach number
Induced drag coefficient	C_{D_I}	Mach number
Initial mass	m_0	
Drag polar coefficient	a	
Wing area	S	
Fuel consumption coefficient	c	
Load factor bounds	n_{min} , n_{max}	Altitude, Mach number
Dynamic pressure bound	q_{max}	

for the 3-D point-mass model of an aircraft are

$$\begin{aligned}
\dot{x} &= v\cos\gamma\cos\chi \\
\dot{y} &= v\cos\gamma\sin\chi \\
\dot{h} &= v\sin\gamma \\
\dot{v} &= \frac{1}{m}\left\{ uT_{\max}(h, M(h,v)) - \left(C_{D_0}(M(h,v)) \right.\right. \\
&\quad \left.\left. + \, C_{D_I}(M(h,v)) \left(\frac{nmg}{S\frac{1}{2}\varrho(h)v^2} - a \right)^2 \right) S\frac{1}{2}\varrho(h)v^2 \right\} \\
&\quad - g\sin\gamma \\
\dot{\gamma} &= \frac{g}{v}(n\cos\mu - \cos\gamma) \\
\dot{\chi} &= \frac{g}{v}\frac{n\sin\mu}{\cos\gamma} \\
\dot{m} &= -cuT_{\max}(h, M(h,v)) \qquad (1)
\end{aligned}$$

(e.g., [24]). The state variables x, y, h, v, γ, χ, and m refer to the x-range, the y-range, altitude, velocity, flight path angle, heading angle, and mass of the aircraft, respectively. The normal acceleration of the aircraft is controlled with the loadfactor n and the tangential acceleration with the throttle setting $u \in [0, 1]$. The loadfactor is the ratio of the lift force and the gravitational force that affect the aircraft. In 3-D flight, the loadfactor can be directed with the bank angle $\mu \in [-\pi, \pi]$. The gravitational acceleration g and the specific fuel consumption coefficient c are assumed constant. The aircraft mass is also included as a state variable because flight times of several hundred seconds may reduce the mass considerably. $T_{\max}(h, M(h,v))$ denotes the maximum available thrust force $C_{D_0}(M(h,v))$, and $C_{D_I}(M(h,v))$ denotes zero-lift and induced drag coefficients, respectively. S refers to the reference wing area of the aircraft. Modern flight control systems are able to reduce the induced drag (e.g., [12], [29]). The effect can be approximated by a positive constant a.

The Mach number $M(h,v)$ and the air density $\varrho(h)$ are computed on the basis of the ISA standard atmosphere. The equations of motion in the vertical x, h-plane are similar to the equations above, but the y-range, the heading angle, and the bank angle are not present.

The pilot and the aircraft itself set some constraints for the state and the control variables. The loadfactor n has lower and upper limits which in general depend upon the altitude and the Mach number of the aircraft

$$n_{\min}(h, M(h,v)) \le n \le n_{\max}(h, M(h,v)). \qquad (2)$$

The feasible region of stationary flight is described by the minimum altitude constraint

$$H(h) = h_{\min} - h \le 0 \qquad (3)$$

the minimum velocity constraint

$$V(h,v) = v_{\min}(h) - v \le 0 \qquad (4)$$

and the maximum dynamic pressure constraint

$$Q(h,v) = \frac{1}{2}\varrho(h)v^2 - q_{\max} \le 0. \qquad (5)$$

In addition to the constraints above, the problem definition includes the initial and terminal conditions for the state variables. For example, in 3-D minimum time flight to a given moving target, the terminal condition of the aircraft is defined by

$$\begin{aligned}
\Psi(h(T), x(T), y(T)) &= \alpha_1(h(T) - h_{tar}(T))^2 \\
&\quad + \alpha_2(x(T) - x_{\text{tar}})^2 \\
&\quad + \alpha_3(y(T) - y_{\text{tar}}(T))^2 \\
&= d^2 \qquad (6)
\end{aligned}$$

where $h_{\text{tar}}(T), x_{\text{tar}}(T)$, and $y_{\text{tar}}(T)$ refer to the position of the target at the capture instant T. For a target, these quantities are uniquely determined by the target's initial position, velocity, flight path angle, and heading angle. The positive constant d refers to the capture radius. Different scales of the state variables are balanced by the coefficients α_1, α_2, and α_3. If the flight state of the interceptor must coincide exactly with the flight state of the target at the capture instant, then the interceptor's velocity, heading angle, and flight path angle must also be included in the terminal conditions.

Different aircraft types can be distinguished from each other by a set of parameters that are present in the dynamic system or in the constraints. In VIATO, the aircraft is characterized by ten parameters that are shown in Table I.

B. Objective Function

With the current implementation of VIATO, one can solve various minimum time problems including minimum time climb, minimum time trajectories to a fixed or a moving target on the vertical plane, and 3-D interception. A moving target is specified by its initial position, velocity, and flight path angle. In minimum time problems, the objective function is the total flight time to the terminal state

$$J(n,\mu,u)=\int_0^{T(n,\mu,u)} dt, \qquad T(n,\mu,u)>0. \tag{7}$$

Other appropriate optimization tasks such as those related to fuel consumption and tracking could be implemented as well. In the minimum fuel consumption problem, the objective function to be maximized is the final mass of an aircraft

$$J(n,\mu,u)=m(T(n,\mu,u)). \tag{8}$$

The final time is free or fixed. In tracking problems, the objective is to maintain the aircraft as close as possible to a given reference flight path by minimizing the average squared deviation from it.

IV. Optimization Server

The optimization server is an independent program that is implemented with Fortran 77. The server discretizes the problems using direct collocation [19] or a scheme based on differential inclusions [28], [33]. The finite dimensional approximation of the original optimal control problem is solved by utilizing a nonlinear programming package NPSOL [16], an implementation of sequential quadratic programming (SQP), (e.g., [3]).

Another way to solve an optimal control problem is to use indirect methods that solve the necessary conditions for optimality. The necessary conditions constitute a multipoint boundary value problem that can be solved mainly by using three approaches: shooting methods, indirect collocation, and finite difference methods. These methods produce accurate results, but they are not always useful for an automated solution process because their convergence domain is small. The initial guess for both the state and adjoint variables must lie close to the optimal solution. In addition, the switching structure, i.e., the sequence of the constrained, unconstrained, and singular arcs of the solution must be specified in advance.

A. Discretization Methods in VIATO

Several methods exist for converting optimal control problems into finite dimensional optimization problems. For a review, see [21]. A suitable approach for automated solutions is to discretize both the control and the state variables. In this way, the explicit numerical integration of the state equations and the objective function is avoided. Furthermore, the state and the control constraints of the problem are treated as usual constraints of nonlinear optimization. In VIATO, the state and control discretization is carried out using direct collocation [19] or a scheme based on differential inclusions [33].

The direct collocation method seeks the solution of the problem among piecewise-defined polynomials that have to satisfy the state equations pointwisely. For simplicity, assume that the state x and the control u are scalars. Let $0=t_1<\cdots<t_N=T$ be an equidistant division of the solution interval $[0,T]$. The state trajectory is represented in each subinterval $[t_i,t_{i+1}]$ by cubic of the form

$$p(\tau)=\alpha_1+\alpha_2\tau+\alpha_3\tau^2+\alpha_4\tau^3 \tag{9}$$

where $\tau\in[0,1]$ is the normalized time. The control variable is approximated piecewise linearly.

Define the defect at the center of each subinterval as

$$\Delta=\dot{p}(1/2)-f(p(1/2),u(1/2)) \tag{10}$$

where f denotes the state equation of the original optimal control problem. The quantities $\dot{p}(1/2), p(1/2)$, and $u(1/2)$ can be expressed using $p(0), p(1)$, $u(0)$, and $u(1)$ (see [19]). Therefore, the state and the control variables at each time point become decision variables of the finite dimensional optimization problem. Now the controls at the time points can be selected freely within their bounds to minimize the objective function subject to $\Delta=0$ and the state and control constraints. The resulting solution satisfies the state equation of the original problem at the center of each subinterval and at each time point t_i.

The scheme based on differential inclusions replaces the differential equation constraint with a requirement that each state must lie in the approximate set of attainability of the previous state. A set of attainability is defined as the collection of the states that can be reached from a given state in some finite time interval with admissible controls. To identify the set of attainability, the hodograph of the dynamical system is employed. It is defined as the set of possible state rates that can be achieved in the current state with an admissible control [33]. If the state equation and the constraints of the optimal control problem are given by

$$\dot{x}=f(x,u), \quad \psi(x)\le 0, \quad \theta(x,u)\le 0 \tag{11}$$

the hodograph of the dynamic system at state x can be written as

$$\mathcal{H}(x)=\{\dot{x}\in R\,|\,\dot{x}=f(x,u),\psi(x)\le 0,\theta(x,u)\le 0\}. \tag{12}$$

Let us assume that there are smooth functions p and q such that the hodograph can be expressed as

$$\mathcal{H}(x)=\{\dot{x}\in R\,|\,p(\dot{x},x)=0,q(\dot{x},x)\le 0\}. \tag{13}$$

In practice, p and q are formed by eliminating the control variables from the state equations f and the constraints ψ and θ.

After identifying the hodograph, the first-order approximation of the set of attainability at state x_0 within the time interval $[t_0,t_1]=\Delta t$ is given by

$$\hat{K}(x_0,t_0,t_0+\Delta t)=\{x\in R^n\,|\,x=x_0+\Delta t\cdot\mathcal{H}(x_0)\} \tag{14}$$

where

$$\Delta t\cdot\mathcal{H}(x_0)=\{\Delta t\cdot\dot{x}\,|\,\dot{x}\in\mathcal{H}(x_0)\}. \tag{15}$$

Let $0=t_1<\cdots<t_N=T$ again be a subdivision of the solution interval. Using the first-order approximation of the

state derivative, the approximate set of attainability (14) can be replaced by the conditions

$$p\left(\frac{x_i - x_{i-1}}{t_i - t_{i-1}}, x_{i-1}\right) = 0,$$
$$q\left(\frac{x_i - x_{i-1}}{t_i - t_{i-1}}, x_{i-1}\right) \leq 0, \qquad i = 2, \dots, N. \tag{16}$$

In this way, the constraints (11) are replaced with the approximate relations (16) that can be treated as constraints of a finite dimensional optimization problem.

In the optimization tasks which are implemented in VIATO's optimization server, the controls can be eliminated analytically. In problems where the explicit elimination is not possible, the controls must be solved using some iterative method every time the constraints (16) are evaluated. The elimination of the control variables might be beneficial in problems involving singular control arcs [33]. The direct collocation and differential inclusion schemes are compared in [28].

Define vector X of the decision variables by $X = [x_1', \dots, x_N', u_1', \dots, u_N', T]'$ for the direct collocation method and $X = [x_1', \dots, x_N', T]'$ for the differential inclusion scheme, respectively. Here, $[\cdot]'$ means the transpose, T is the final time, and x_i and $u_i, i = 1, \dots, N$ refer to the state and the control vectors at time t_i. The dimension of X depends upon the dimension of the state and control vectors. The discretization methods replace the optimal control problem with the finite dimensional approximation

$$\min f(X)$$

subject to

$$c(X) = \bar{0}$$
$$g(X) \leq \bar{0}$$

where f is the objective function, and c and g define the appropriate equality and inequality constraints.

The selection of the discretization grid affects the accuracy of the finite dimensional solution. In principle, the accuracy can be increased by careful allocation of the discretization points. For example, if solutions obtained by equidistant points show rapid state transitions in some time interval, then the problem can be solved again such that the points are placed more densely in the intervals of rapid state transitions and more sparsely in intervals with slow state rates. In practice, the discretization points must be updated adaptively. In this process, the problem is first solved using the given gridpoints. Then the grid is refined by adding and relocating the gridpoints. The refinement strategies are based mainly on either the local approximation error [4] or the equidistribution of the approximation error [22]. The adaptive grid refinement is useful if the initial grid is sparse. Nevertheless, even personal computers are so powerful today, that the optimization problems can be solved using a sufficiently dense grid.

VIATO utilizes equidistant discretization and continuation with respect to the number of the discretization points. When solving an optimization problem, the user can choose the discretization method and the initial number of the discretization points. The problem can first be solved with a sparse discretization grid to get a rough initial estimate of the solution. Then the accuracy of the approximation is increased by adding discretization points. While the next solution is being computed, the user may already study the previous solution. The new solutions are produced such that the previous solution is used as the initial guess. In this way, a solution with a dense discretization grid is often obtained faster than by solving the problem only once using a dense grid.

B. Nonlinear Programming by SQP

SQP is widely used in solving discretized optimal control problems, and several versatile implementations of SQP are available (e.g., [16], [17], [35]). SQP is a method in which the objective function is approximated with a quadratic function at each iteration step, and nonlinear constraints are linearized. The resulting linear-quadratic problem is solved by a suitable quadratic programming method, and a line search between the iteration point and the solution of the quadratic problem is carried out. The SQP method converges rapidly and reliably when applied to a discretized optimal control problem [17].

Implementations that treat matrices as sparse [9], [17], [26] can be used, because the Jacobian of the constraints and the Hessian of the Lagrangian are almost block-diagonal. Nevertheless, the special treatment of sparsity is beneficial only if the number of discretization points is large. VIATO's optimization server is implemented with the NPSOL subroutine [16] that treats the matrices as dense.

In NPSOL, SQP is implemented using an augmented Lagrangian merit function [15]. During the solution process, the step length of the line search is chosen such that the value of the merit function is decreased properly. Convexity of the Lagrangian is ensured by a quadratic augmentation term. Therefore, the iteration point will converge from any reasonable initial guess to a point that satisfies the first-order necessary conditions for optimality.

In optimization routines, the values of the objective and the constraints and their first derivatives with respect to the decision variables are needed at arbitrary points. The derivatives could be computed by finite differences. In VIATO's optimization server, faster and more reliable convergence is achieved by using analytical expressions.

C. Initial Guess for Decision Variables

Although nonlinear programming methods have a large convergence domain, the initial guess for decision variables cannot be chosen arbitrarily. VIATO's optimization server determines the initial guess $x_{i,\text{ini}}^j$ from a straight line connecting the initial and the terminal conditions

$$x_{i,\text{ini}}^j = x_1^j + (x_f^j - x_1^j)\frac{(i-1)}{(N-1)}, \qquad i = 1, \dots, N \tag{17}$$

where N is the number of the discretization points, x_1^j is the initial value, and x_f^j is the terminal value of the particular state j. If the initial or terminal value of a variable is not fixed, an initial guess is generated using heuristic rules that utilize the given initial and terminal states. An initial guess for the final time is produced by estimating the average velocity of the aircraft and computing the distance from the initial state to the terminal or predicted capture position.

Computational experience with VIATO has shown that in most cases, this initial guess leads to convergence. Furthermore, convergence from this nonfeasible initial guess seems to be faster than from a feasible initial guess that is obtained by integrating the state equations with appropriate feasible controls.

D. Scaling of Variables and Constraints

An essential part of the solution is the scaling of the variables and the constraints. Optimization routines generally assume that the decision variables and the constraints are roughly of the same magnitude. Badly scaled decision variables and constraints may cause numerical errors in the iteration. A suitable scaling is an affine transformation [15]. In VIATO, each state and control variable is normalized by the absolute maximum of the initial guess

$$x_{i,\text{sca}}^j = \frac{x_i^j}{\max\{|x_{1,\text{ini}}^j|, \ldots, |x_{N,\text{ini}}^j|\}}, \qquad i = 1, \ldots, N. \tag{18}$$

Here N is the number of discretization points, $x_{i,\text{sca}}^j$ is the scaled variable with j specified as above, and $x_{1,\text{ini}}^j, \ldots, x_{N,\text{ini}}^j$ are the initial guesses for the particular state or control variables. The final time is scaled as

$$T_{\text{sca}} = \frac{T}{U} \tag{19}$$

where U is the estimated upper bound for the final time T. The components of the constraints c and g are scaled by their values at the initial guess.

V. Model Server

In VIATO, different aircraft types are characterized by a set of parameters which are stored in a model library. The values of the induced drag and the zero drag coefficient, as well as the maximum available thrust force, are known only at certain Mach numbers and altitudes as tabular data. The optimization routine needs the values of the parameters at arbitrary states, and therefore, the data must be approximated or interpolated using some continuous functions.

An interpolation fits the data exactly, but it might fluctuate between the data points. Furthermore, the data often consists of measurements which may include noise, and thus, exact fitting is not appropriate. For these reasons, approximation is chosen instead of interpolation in VIATO.

For optimization purposes, the approximation has to be smooth and continuous. Furthermore, the evaluation of the approximation must not require considerable computational effort, because the optimization routine calls the approximation numerous times during the optimization.

The model server of VIATO automatically creates continuous and smooth approximations which can be affected interactively. The user can make a tradeoff between the accuracy and the smoothness of the approximation (see Fig. 2).

The drag coefficients are approximated using a rational polynomial or piecewise cubic splines. The form of the rational approximation is

$$f(M) = \frac{\sum_{i=0}^{n} c_i M^i}{1 + \sum_{i=1}^{m} d_i M^i} \tag{20}$$

where the coefficients c_i and d_i are determined by least squares fitting. When the approximation is determined interactively, the user can test different degrees of the denominator and the numerator and select the most suitable ones. The coefficients d_i must be chosen such that no poles arise in the range of the argument M. If this happens, the user interface gives an error message. A rigorous treatment of the poles would lead to a constrained and nonconvex optimization problem that most likely cannot be solved automatically.

In a spline approximation (see [10]), a cubic polynomial is created at the same time for every subinterval of the knots $0 = M_1 < \cdots < M_n = M_{\max}$. Note that the knots are not necessarily the same as the data points. The form of the cubic spline for every interval $[M_i, M_{i+1}]$ is

$$s_i(M) = a_i + b_i M + c_i M^2 + d_i M^3 \tag{21}$$

where $M \in [M_i, M_{i+1}]$. The splines are fitted to the data by least squares fitting and by requiring that the approximation and its first and second derivative are continuous. The user can add new knots and decide the tradeoff between the accuracy and the smoothness of the spline approximation.

The maximum thrust force is approximated by a 2-D polynomial or a B-spline. The 2-D polynomial approximation is

$$P(M, h) = \sum_{i=0}^{p} \sum_{j=0}^{q} c_{i,j} M^i h^j. \tag{22}$$

The coefficients $c_{i,j}$ are determined by least squares fitting. In the interactive process, the user can test different degrees p and q. A disadvantage of a polynomial approximation is the fluctuation that appears with high-order polynomials in particular. It can be avoided using piecewisely-defined 2-D polynomials, e.g., a B-spline approximation [10]. This approximation is formed using the knots $0 = M_1 < \cdots < M_m = M_{\max}$, $0 = h_1 < \cdots < h_l = h_{\max}$ and base function B_i, given by

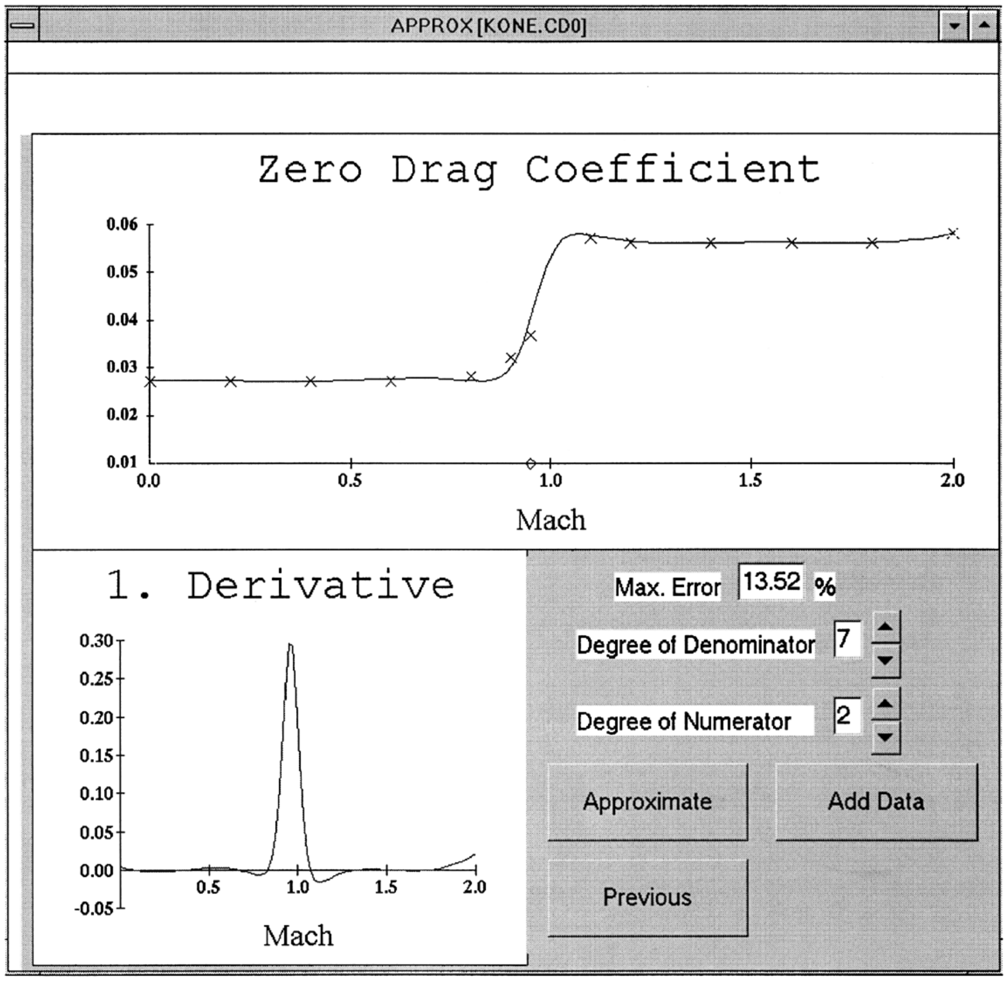
$$B_i(x) = (x_{i+4} - x_i) \sum_{j=i}^{i+4} \frac{[(x_j - x)^3]_+}{\Phi_i'(x_j)} \tag{23}$$

where x_i is a knot and

$$\Phi_i'(x) = \sum_{j=i}^{i+4} \prod_{\substack{k=i \\ k \neq j}}^{i+4} (x - x_k). \tag{24}$$

The function $[x]_+$ is defined as

$$[x]_+ = \begin{cases} 0, & \text{when } x < 0 \\ x, & \text{when } x \geq 0. \end{cases}$$



Max. Error 13.52 %
Degree of Denominator 7
Degree of Numerator 2

Fig. 2 Visualization of the approximations. A tradeoff can be made between the accuracy and the smoothness of the approximation.

The 2-D continuous and smooth B-spline approximation is formed by the sum of the base functions

$$S(M,h) = \sum_{i=1}^{l-4} \sum_{j=1}^{m-4} \beta_{i,j} B_j(M) B_i(h). \tag{25}$$

The coefficients $\beta_{i,j}$ are determined by fitting the function (25) to data points in the least-squares sense. If the user forms the approximation interactively, he or she can make the tradeoff between accuracy and smoothness of the approximation by adding or removing knots.

The continuous approximations for the drag coefficients and the thrust force are formed and displayed together with the wing area and the mass of the aircraft by the model server. A typical aircraft model is presented in Fig. 3.

If necessary, a similar approach can be adopted for the load-factor upper and lower bounds $n_{\min}(h,M)$ and $n_{\max}(h,M)$. At this stage, however, bounds are assumed to be constant.

VI. Example Runs with VIATO

The software is demonstrated by solving two example aircraft trajectory optimization tasks. The example problems are a minimum time climb, where the total time needed in achieving a certain altitude and velocity is minimized, and a 3-D minimum time flight to a fixed target. The software is operated in a PC equipped with a 200 MHz PentiumPro processor and Windows NT operating system.

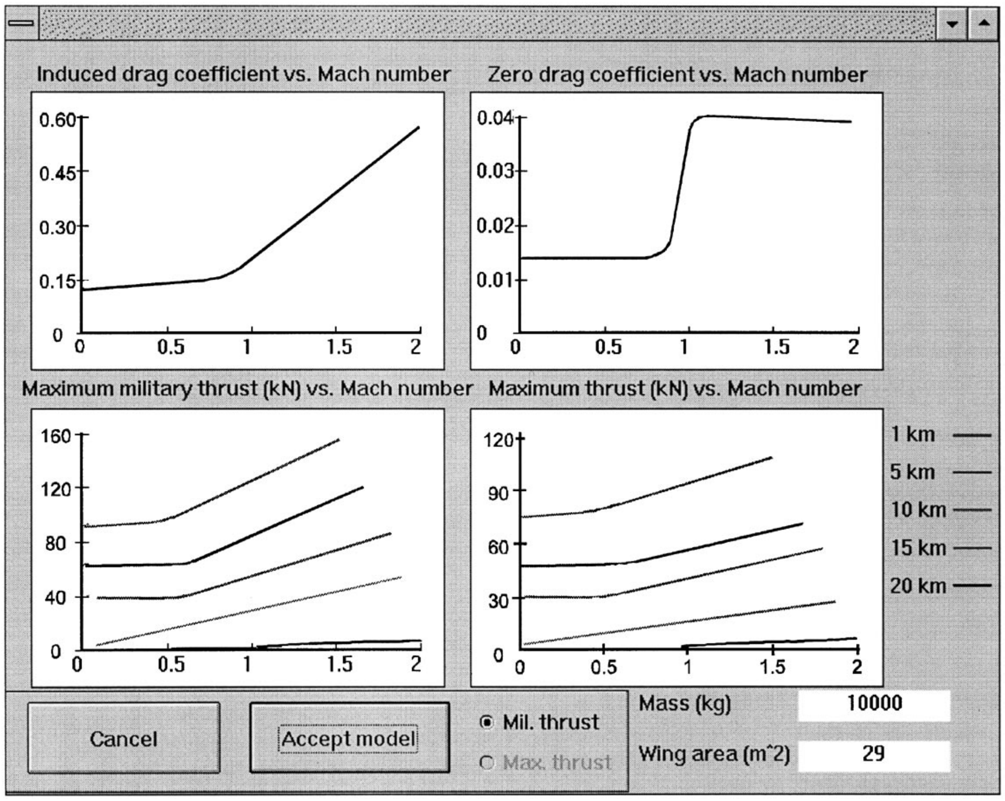
The user first chooses the aircraft model from the model library. In the examples, we use a generic modern fighter aircraft. After the aircraft model has been selected, the user can choose the problem type from among four possible optimization tasks mentioned earlier.

A. Minimum Time Climb on a Vertical Plane

After the problem type has been chosen, the initial and the terminal conditions are specified (see Fig. 4). The user can also choose the method of discretization (either direct collocation or differential inclusion) and the number of discretization points. By default, the server uses the collocation method and ten discretization points.

In the example, the initial and the terminal conditions of the aircraft are

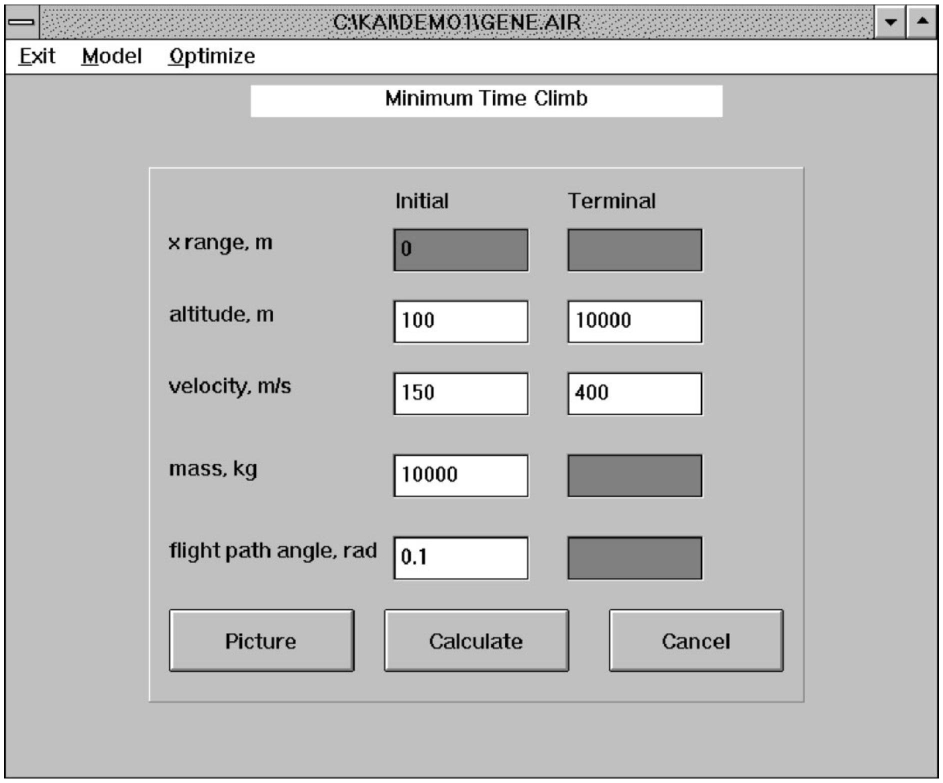
$$h(0) = 100 \text{ m}, \quad \gamma(0) = 0.1 \text{ rad}, \quad v(0) = 150 \text{ m/s},$$
$$m(0) = 10\,000 \text{ kg}, \quad h(T) = 10\,000 \text{ m},$$
$$v(T) = 400 \text{ m/s}.$$


Induced drag coefficient vs. Mach number
Zero drag coefficient vs. Mach number
Maximum military thrust (kN) vs. Mach number
Maximum thrust (kN) vs. Mach number
1 km
5 km
10 km
15 km
20 km
Cancel
Accept model
Mil. thrust
Max. thrust

Mil. thrust [✓]

Mass (kg)	10000
Wing area (m^2)	29

Fig. 3. Typical aircraft model.



Minimum Time Climb

	Initial	Terminal
x range, m	0	
altitude, m	100	10000
velocity, m/s	150	400
mass, kg	10000	
flight path angle, rad	0.1	

Fig. 4. Specification of the initial and the terminal conditions.

The initial conditions correspond to a flight situation just after a takeoff.

Once the user interface has received the required information, it calls the optimization server. The server solves the

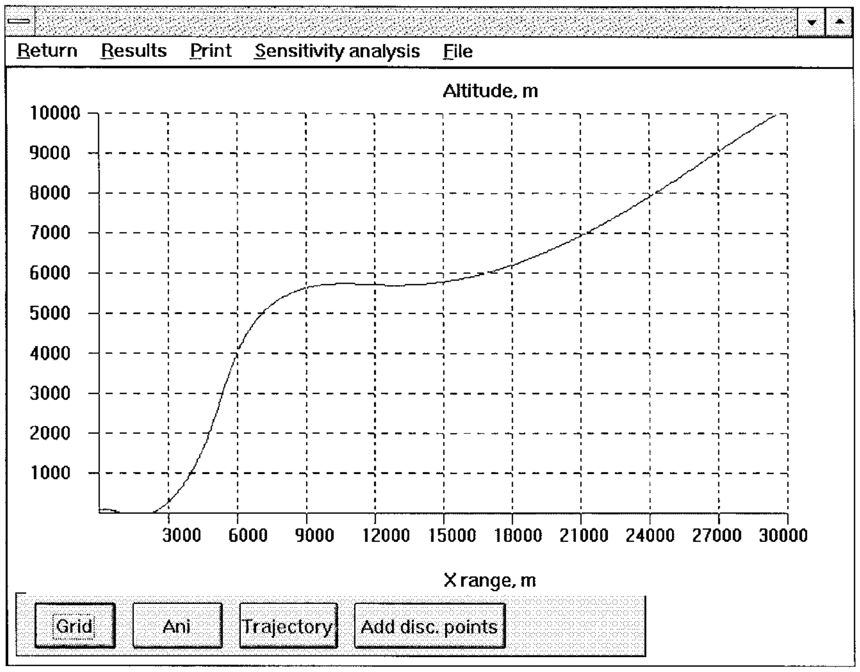


Fig. 5. Visualization of 2-D trajectories in VIATO.

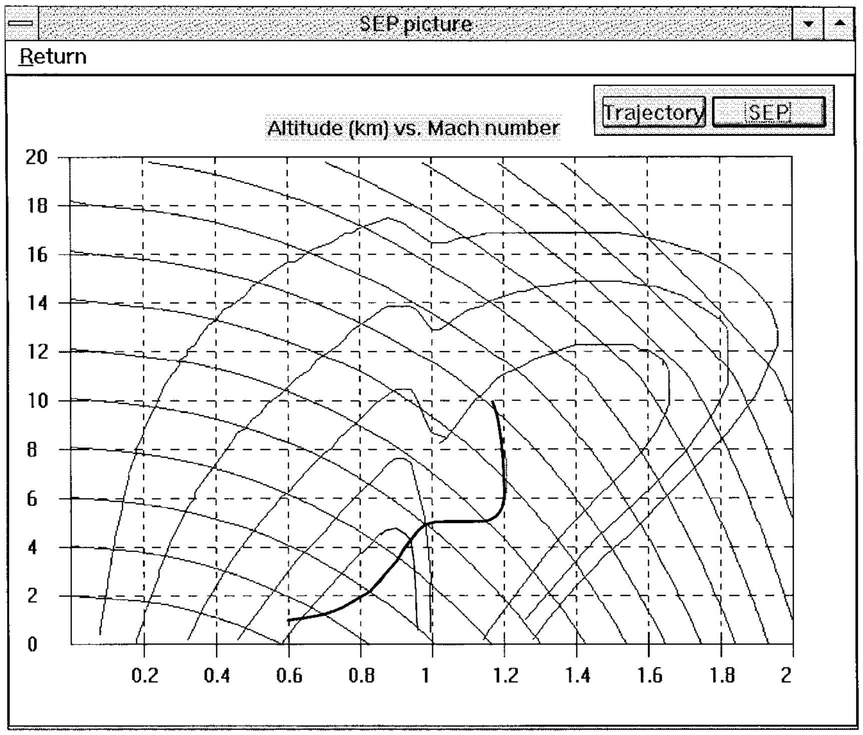


Fig. 6. Optimal solution shown with the SEP-plot.

problem and returns the optimal solution to the user interface. Optimal solutions are presented graphically, and results can be saved in a file. Thus, it is possible to read the solutions in other applications as well.

Solutions obtained with a 2-D point-mass model are shown in 2-D plots (see Fig. 5). The optimal state and control variables can be plotted as a function of some other state or control variable or the total flight time. The optimal trajectories can further be shown together with constant energy contours and contours of the time derivative of the aircraft's total energy, also called specific excess power (SEP) in the Mach number-altitude plane (see Fig. 6). In addition to the plots, trajectories to a moving target can be animated.

The example minimum time climb problem is solved with 10, 20, 30, and 40 discretization points. The computation times are 1, 4, 15, and 40 s, respectively. The optimal flight

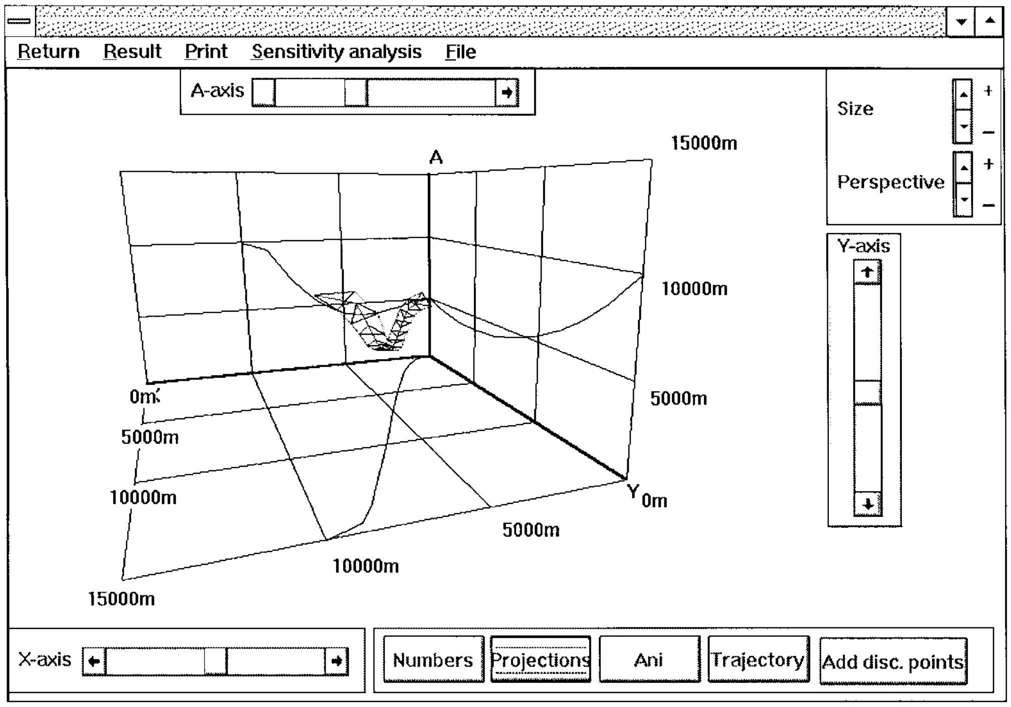


Fig. 7. Visualization of the time optimal trajectory to the fixed target in 3-D space.

time is approximately 97 s. The optimal trajectory with 30 discretization points is shown in Fig. 5.

B. Three-Dimensional Interception

In the second example, a 3-D interception problem is solved. The objective of the interception is to fly in minimum time from the initial state

$$x(0) = 0 \text{ m}, \quad y(0) = 0 \text{ m}, \quad h(0) = 5000 \text{ m},$$
$$v(0) = 200 \text{ m/s}, \quad \gamma(0) = 0.1 \text{ rad}, \quad \chi(0) = 0.1 \text{ rad}$$

to the terminal point

$$x(T) = 10000 \text{ m}, \quad y(T) = 10000 \text{ m}, \quad h(T) = 10000 \text{ m},$$
$$\gamma(T) = 0 \text{ rad}, \quad \chi(T) = 0 \text{ rad}.$$

The terminal velocity $v(T)$ of the aircraft is free. The initial conditions correspond to a subsonic cruise situation.

In addition to the 2-D plots, the optimal trajectory can now be plotted in a 3-D picture where the bank angle of the aircraft is also illustrated (see Fig. 7). The user can change the perspective, the size, and the view angle of the 3-D picture. It is also possible to draw the projections of the optimal trajectory on the $x, y-, x, h-$, and y, h-planes.

The example is solved using 10, 20, 30, and 40 discretization points and the direct collocation method. The computation times are 3, 15, 50, and 120 s, respectively. The 3-D time optimal trajectory with 20 discretization points is shown in Fig. 7. The dimension of the problem is larger, and there are more constraints than in the minimum time-climb task. For these reasons, the computation times increase but are still rather reasonable. The flight takes 63 s.

VII. Sensitivity Analysis

Solutions of optimal control problems depend upon model parameters. Their impact can be studied using basic sensitivity analysis. In VIATO, the effect of the maximum thrust force or the drag coefficients, as well as the initial or terminal states, can be studied. As an example, we consider the minimum time-climb of the previous section and study its sensitivity with respect to the initial mass m_0.

We write the discretized problem as

$$\min T$$

subject to

$$\begin{aligned} c(X) &= \bar{0} \\ g(X) &\leq \bar{0} \\ m_1 - m_0 &= 0 \end{aligned} \tag{26}$$

where the initial condition for the mass of the aircraft has been written separately.

Let us assume that the optimal solution of the above problem is X^*. Denote by $\tilde{c}$ the equality and the binding inequality constraints at X^*, and let $\tilde{\lambda}^*$ be the corresponding Lagrange multiplier. Further, let $\lambda^*_{m_0}$ correspond to the constraint (26), and assume that the solution satisfies the usually assumed conditions for basic sensitivity theorems (e.g., [14, Th. 3.2.2]).

The Lagrange multipliers represent the first-order sensitivity or the rate of change of the optimal objective function as the parameter that is associated to the particular constraint changes. The derivative of the optimal objective function at T^* with respect to the parameter m_0 is

$$\frac{\partial T^*}{\partial m_0} = \lambda^*_{m_0} \tag{27}$$

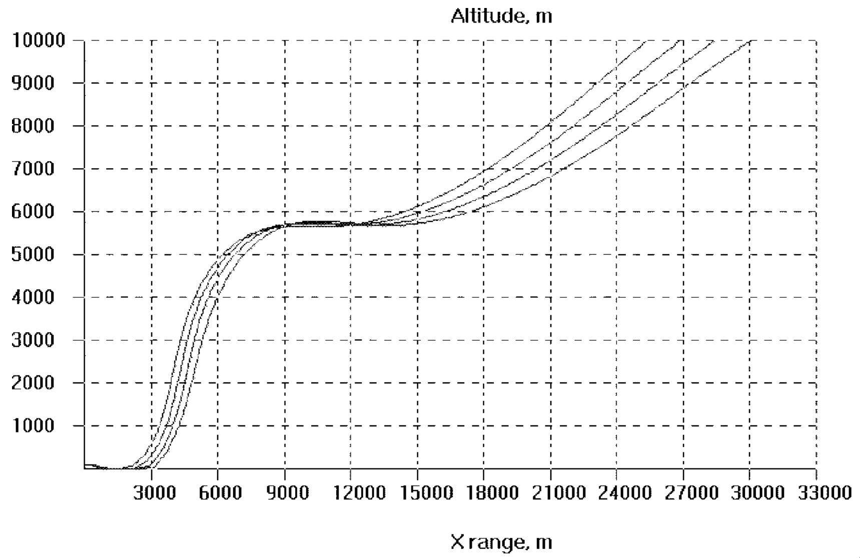


Fig. 8. Family of minimum time climb trajectories obtained with 20 discretization points. The initial mass of the aircraft is changed in 500 kg intervals from 9000 to 10 500 kg.

so that the first-order approximation for the change of the optimal flight time is given by

$$\Delta T^* \approx \lambda^*_{m_0} \Delta m_0. \qquad (28)$$

In the first example run, the value of $\lambda^*_{m_0}$ is approximately

$$\lambda^*_{m_0} \approx 0.012672.$$

Thus, a change of 100 kg in the initial mass changes the total flight time approximately 1.3 s.

The sensitivity of the optimal solution $(X^{*\prime}, \lambda^{*\prime})$, $\lambda^{*\prime} = (\tilde{\lambda}^{*\prime}, \lambda^*_{m_0})$ with respect to m_0 can be studied by using the relation

$$\frac{\partial}{\partial m_0}\begin{pmatrix} X^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix}^{-1}_{\Big| \substack{X=X^* \\ \lambda=\lambda^*}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \qquad (29)$$

(for the general formula, see [14], Ch. 4) where A is the Jacobian of the constraint $(\tilde{c}(x)', m_1 - m_0)$ and H is the Hessian of the Lagrangian L defined by

$$L = T + \tilde{\lambda}^{*\prime}\tilde{c}(x) + \lambda^*_{m_0}(m_1 - m_0). \qquad (30)$$

However, for the presented aircraft trajectory optimization problems, the use of the sensitivity results proved unreliable. The accuracy of the solution depends on the feasibility and optimality tolerances that are used in the optimization algorithm. In practice, the tolerances cannot be arbitrarily tight, because the approximate solution should converge in moderate time. The approximate nature of the solution and the Lagrange multipliers cause numerical errors in the Jacobian and especially in the Hessian, which makes (29) ill-conditioned. Another problem is that the set of binding constraints must be unchanged as the sensitivity parameter changes.

In VIATO, the sensitivity analysis also can be carried out by solving the problem with a new value of the parameter after the problem has been solved once. The user chooses the varying parameter and its new value, and the problem is solved such that the nominal solution is used as the initial guess. In this way, the perturbed problem can be solved quickly. The solutions of the perturbed problems are visualized together with the nominal solutions.

As an example, the first problem of Section VI is now solved such that the initial mass is changed in 500 kg intervals from 9000 to 10 500 kg. The number of the discretization points is 20. The optimal flight times are 87, 91, 96, and 100 s, respectively. The family of the optimal solutions is shown in Fig. 8. The computation times of the perturbed problems varied between 3 and 5 s. The results are in agreement with (28).

VIII. Conclusion

We have introduced an approach for the automated solution of optimal flight trajectories. The structure of the aircraft models and the objectives of the problems are specified in advance, and different aircraft types are characterized by sets of parameters which are stored in a model library. Restricting to a class of problems facilitates the efficient automatization of the solution process. The approach is implemented in the VIATO software which consists of an optimization server, a model server, and an intuitive, menu-driven, graphical user interface. As far as the authors know, VIATO is the only software that can be run automatically by a nonexpert user. Preliminary tests have shown that after a short introduction, aircraft engineers without any specific background in optimization are able to use the software independently. Due to the efficient checking of the initial and terminal states, numerical problems are unlikely to occur.

The implementation efficiently hides the mathematics and practical problems related to the formulation and solution of optimal control problems. In addition, much of the laborious work concerning model management has been automatized. The use of discretization and nonlinear programming in the optimization server enlarges the convergence domain and ensures the convergence from the automatically generated initial guesses. The treatment of the necessary conditions, as well as the estimation of the switching structure, is avoided.

Due to discretization, the solutions are approximate. The accuracy can be improved either by increasing the number of discretization points or by reallocating the current points adaptively. Computational experience with VIATO has shown that the present computational power seems to favor the former alternative even in interactive use.

Aside from flight path optimization, a similar approach also could be tailored to other application areas where repeated optimal trajectories are needed, and global solutions cannot be obtained. Such an area might be, for example, industrial robot manipulator trajectory planning.

ACKNOWLEDGMENT

The authors would like to thank the Finnish Air Force for their support and collaboration.

REFERENCES

[1] *Borland Delphi for Windows User's Guide*. Scott's Valley, CA: Borland, 1995.

[2] *Matlab 5 User's Guide*. Natick, MA: MathWorks, 1997.

[3] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.

[4] J. T. Betts and W. P. Huffman, "Path-constrained trajectory optimization using sparse sequential quadratic programming," *J. Guid., Contr., Navigation*, vol. 16, pp. 59–68, Jan./Feb. 1993.

[5] R. R. Bless, E. M. Queen, M. D. Cavanaugh, T. A. Wetzel, and D. D. Moerder, "Variational trajectory optimization tool set, technical description, and user's manual," NASA Langley Res. Center, Hampton, VA, NASA Tech. Memo. 4442, 1993.

[6] A. E. Bryson and W. F. Denham, "A steepest-ascent method for solving optimum programming problems," *J. Appl. Mech.*, vol. 29, pp. 247–257, June 1962.

[7] A. E. Bryson and Y. Ho, *Applied Optimal Control*. Washington, DC: Hemisphere, 1975.

[8] R. Bulirsch, F. Montrone, and H. Pesch, "Abort landing in the presence of a windshear as a minimax optimal control problem," *J. Optimiz. Theory Applicat.*, vol. 70, no. 1, pp. 1–23, 1991.

[9] A. R. Conn, N. I. M. Gould, and L. Toint, "LANCELOT: A fortran package for large-scale nonlinear optimization (Release A)," in *Computational Mathematics 17* (Springer Series), R. L. Graham, J. Stoer, and R. Varga, Eds. Berlin, Germany: Springer-Verlag, 1992.

[10] C. De Boor, *A Practical Guide to Splines*. New York: Springer-Verlag, 1978.

[11] J. Dolezal and J. Fidler, "Numerical solution of dynamic optimization problems using parametrization and OptiA software," *Appl. Math. Comput.*, vol. 78, no. 2–3, pp. 111–121, 1996.

[12] H. Ehtamo, T. Raivio, and R. P. Hämäläinen, "A method to generate trajectories for minimum time climb," in *Proc. 6th Int. Symp. Dynamic Games and Applications*, St. Jovite, P.Q., Canada, 1994, pp. 175–183.

[13] Y. Fan, F. Lutze, and E. Cliff, "Time-optimal lateral maneuvers of an aircraft," *J. Guid., Contr., Dynam.*, vol. 18, pp. 1106–1112, Sept./Oct. 1995.

[14] A. V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. New York: Academic, 1983.

[15] P. E. Gill and W. Murray, "Model building and practical aspects of nonlinear programming," in *Computational Mathematical Programming*, (NATO ASI Series, vol. F15), K. Schittkowski, Ed. Berlin, Germany: Springer-Verlag, 1985, pp. 209–247.

[16] P. E. Gill, W. Murray, and M. H. Wright, "User's guide for NPSOL (Version 4.0)," Dept. Oper. Res., Stanford Univ., Stanford, CA, Report SOL 86-2, 1986.

[17] P. E. Gill, W. Murray, and M. A. Saunders, "Large-scale SQP methods and their application in trajectory optimization," in *Computational Optimal Contr.*, R. Bulirsch and D. Kraft, Eds. Basel, Germany: Birkhäuser, 1994, pp. 29–42.

[18] V. Y. Glitzer, "Optimal planar interception with fixed end conditions: Closed-form solution," *J. Optimiz. Theory Applicat.*, vol. 88, pp. 503–539, Mar. 1996.

[19] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *J. Guid., Contr., Dynam.*, vol. 10, pp. 338–342, Jul./Aug. 1987.

[20] ——, "Optimal multiple vehicle trajectories," in *Proc. 8th AIAA Atmospheric Flight Mechanics Conf.*, Portland, OR, 1990, 161–167.

[21] D. G. Hull, "Conversion of optimal control problems into parameter optimization problems," *J. Guid., Contr., Dynam.*, vol. 20, pp. 57–60, Jan./Feb. 1997.

[22] D. Kraft, "On converting optimal control problems into nonlinear programming problems," in *Computational Mathematical Programming*, (NATO ASI Series, vol. F15), K. Schittkowski, Ed. Berlin, Germany: Springer, 1985, pp. 261–280.

[23] ——, "Algorithm 733: TOMP—Fortran modules for optimal control calculations," *ACM Trans. Math. Softw.*, vol. 20, pp. 262–281, Sept. 1994.

[24] A. Miele, *Flight Mechanics*, vol. 1. Reading, MA: Addison-Wesley, 1962.

[25] A. Miele, T. Wang, and W. Melvin, "Optimal abort landing trajectories in the presence of windshear," *J. Optimiz. Theory Applicat.*, vol. 55, no. 3, pp. 165–202, 1987.

[26] B. A. Murtagh and M. A. Saunders, "MINOS 5.0 user's guide," Dept. Oper. Res., Stanford Univ., Stanford, CA, Report SOL 83-20, 1983.

[27] H. Pesch, "A practical guide to the solution of real-life optimal control problems," *Contr. Cybern.*, vol. 23, no. 1–2, 1994.

[28] T. Raivio, H. Ehtamo, and R. P. Hämäläinen, "Aircraft trajectory optimization using nonlinear programming," in *Syst. Model. Optimiz.*, J. Dolezal and J. Fidler, Eds. London, U.K.: Chapman & Hall, 1996, pp. 435–441.

[29] R. M. Rennie and E. J. Jumper, "Dynamic leading-edge flap scheduling," *J. Aircraft*, vol. 34, pp. 606–611, Sept./Oct. 1997.

[30] G. Ryan and D. Downing, "Functional agility metrics and optimal trajectory analysis," *J. Guid., Contr., Dynam.*, vol. 17, pp. 1234–1240, Nov./Dec. 1994.

[31] A. Schwartz, "Theory and implementation of numerical methods based on Runge-Kutta integration for solving optimal control problems," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, 1996.

[32] H. Seywald, "Range optimal trajectories for an aircraft flying in the vertical plane," *J. Guid., Contr., Dynam.*, vol. 17, pp. 389–398, Mar./Apr. 1994.

[33] ——, "Trajectory optimization based on differential inclusion," *J. Guid., Contr., Dynam.*, vol. 17, pp. 480–487, May/June 1994.

[34] O. von Stryk, "Numerical solution of optimal control problems by direct collocation," in *Optimal Control*, (International Series in Numerical Mathematics 111), R. Bulirsch, A. Miele, J. Stoer, and K. H. Well, Eds. Basel, Germany: Birkhäuser, 1993, pp. 129–143.

[35] A. L. Tits and J. L. Zhou, "User's guide for FFSQP version 3.5: A FORTRAN code for solving constrained nonlinear optimization problems, generating iterates satisfying all inequality and linear constraints," Elect. Eng. Dept., Univ. Maryland, College Park, Tech. Rep. SRC-TR-92-107r5, 1995.

[36] Y. Tsujikawa, "Analysis of a flight path of subsonic transport aircraft for fuel saving," *Int. J. Turbo Jet Engines*, vol. 12, pp. 189–199, 1995.

[37] J. Vian and J. Moore, "Trajectory optimization with risk minimization for military aircraft," *J. Guid., Control, Dynam.*, vol. 12, pp. 311–317, May/June 1989.

Kai Virtanen received the M.Sc. degree in systems and operations research from the Helsinki University of Technology (HUT), Espoo, Finland, in 1996. He is currently pursuing the Ph.D. degree at HUT.

Currently, he is a Researcher at the Systems Analysis Laboratory, HUT. His research interests cover optimization and game as well as decision theoretical aerospace applications.

Harri Ehtamo received the M.S. and Lic.Phil. degrees in theoretical physics from the University of Helsinki, Finland, in 1978 and 1980, respectively, and the Dr.Tech. degree in systems and operations research from the Helsinki University of Technology (HUT), Espoo, Finland, in 1989.

He was a Teaching Assistant in the Department of Theoretical Physics, University of Helsinki, from 1977 to 1980, and from 1981 to 1983 he was a Research Fellow in the Research Institute for Theoretical Physics in the Department of High Energy Physics, University of Helsinki. Since 1983, he has been with the Systems Analysis Laboratory, HUT, and is currently an Associate Professor. His research interests include dynamic game theory with aerospace applications, negotiation analysis, and optimization theory and algorithms.

Tuomas Raivio received the M.Sc. degree in systems and operations research with a minor in control engineering in 1993 from the Helsinki University of Technology (HUT), Espoo, Finland.

He is currently a Researcher and a Ph.D. student at the Systems Analysis Laboratory, HUT. His research interests include optimization, optimal control, and dynamic games with particular attention to pursuit-evasion games.

Raimo P. Hämäläinen (M'82) received the M.S. and Dr.Tech. degrees in systems theory in 1972 and 1976, respectively, from the Helsinki University of Technology (HUT), Espoo, Finland.

He was Professor of Mathematics at the Vaasa School of Economics and Business Administration, Vaasa, Finland, from 1979 to 1981. In 1980, he worked at the University of California, Los Angeles (UCLA), under a senior Fulbright Research Grant, and he returned to UCLA from 1985 to 1986 and 1991 to 1992. He is currently Professor of Applied Mathematics and Operations Research and Director of the Systems Analysis Laboratory, HUT. His research interests include dynamic optimization, game and decision theory including aerospace applications, distributed AI and agent based simulation, electricity pricing and markets analysis, and environmental decision making. His special interest is in increasing the use of decision models in public policy. He is the designer of the HIPRE 3+ and web–HIPRE decision support software.