An Implementation of the Shooting Algorithm for Solving Optimal Control Problems

J. Frederic Bonnans, S. Maurin

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

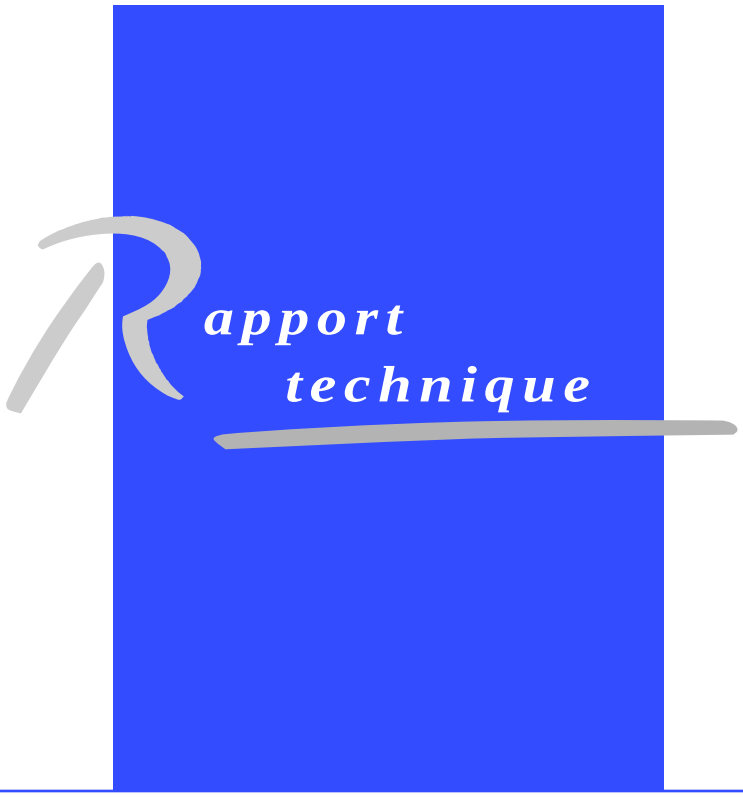
An implementation of the shooting algorithm for solving optimal control problems

F. Bonnans , S. Maurin

N ° 0240

May 2000

THÈME 4



ISSN 0249-0803

INRIA
ROCQUENCOURT

An implementation of the shooting algorithm for solving optimal control problems

F. Bonnans* , S. Maurin†

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet MOCOA

Rapport technique n ° 0240 — May 2000 — 24 pages

Abstract: We describe the software OCS-SA for solving optimal control problems. The method is essentially an implementation of Newton's method applied to an associated two points boundary value problems (TPBVP). The latter is reduced to a finite dimensional problem through the use of a shooting function. We use the MAPLE and SCILAB softwares. The differential equations to be solved by the shooting algorithm are automatically generated, as well as a latex report, including description of the problem and numerical results.

Key-words: Optimal control problems, Newton method, shooting algorithm, algebrico-differential system, two points boundary value problem.

(Résumé : tsvp)

* Action MOCOA, INRIA, B.P. 105, 78153 Rocquencourt, France. Email: Frederic.Bonnans@inria.fr.
† Action MOCOA, INRIA, B.P. 105, 78153 Rocquencourt, France. Email: Sady.Maurin@inria.fr.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Implementation de l'algorithme de tir pour la résolution des problèmes de commande optimale

Résumé : nous décrivons le logiciel OCS-SA pour résoudre les problèmes de commande optimale. Nous implémentons essentiellement la méthode de Newton appliquée au problème aux deux bouts associé (TPBVP). Ce dernier est réduit à un problème en dimension finie à l'aide de la fonction de tir. Nous utilisons les logiciels MAPLE et SCILAB. Les équations différentielles résolues par un algorithme de tir sont générées automatiquement, ainsi qu'un rapport latex, avec la description du problème et les résultats numériques.

Mots-clé : problème de contrôle optimal, méthode de Newton, algorithme de tir, système algébrico-différentiel, problème aux deux bouts.

AMS subject classifications. 49M15, 34B10, 65L10, 65K10.

1. Introduction. In this paper we describe some software tools for solving two related problems. The first is the following class of *optimal control problem*:

$$\min_{(y,u)} \quad \int_0^T \ell(t, y(t), u(t))\, dt + F(y_0, y_T), \tag{1.1}$$

subject to

$$\dot{y} = f(t, y(t), u(t)), \quad t \in [0, T], \tag{1.2}$$

$$0 = A(y_0, y_T), \tag{1.3}$$

$$y_i(0) = (y_0)_i, \quad i \in I \subset \{1, \ldots, n_y\}. \tag{1.4}$$

The variables y and u are called *state* and *control*, respectively, t is the time, and T is the horizon or final time. The functions $\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $F : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and $A : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$, where $q \leq n$, are called *distributed cost function*, *final cost function*, *dynamics*, and *initial-final constraints*, respectively. The latter may, for instance, represent periodicity conditions. Finally, (1.4) represents intial conditions on the components of the initial state that belong to some subset I of $\{1, \ldots, n_y\}$.

Equations (1.2), (1.3) and (1.4) are called the *state equation*, *initial and final state constraints*, and *initial conditions*, respectively.

The second problem, called the *two points boundary value problem* (TPBVP), has the following format:

$$\dot{x} = h(t, x(t), v(t)), \quad t \in [0, T], \tag{1.5}$$

$$0 = g(t, x(t), v(t)), \quad t \in [0, T], \tag{1.6}$$

$$0 = B(x(0), x(T)). \tag{1.7}$$

In this setting we call x (resp. v) the *differential* (resp. *algebraic*) variable; $x(t)$ and $v(t)$ have dimension n_x and n_v, respectively. The functions $h : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$, $g : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_v}$, and $B : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, are of class C^1.

The paper is organised as follows. Section 2 is devoted to numerical methods for solving the TPBVP (1.5)-(1.7), namely the simple and multiple shooting algorithms. Section 3 discusses the links between the optimal control problem (1.1)-(1.4) and TPBVP. Section 4 describes the software, while numerical examples are presented in section 5. Finally in section 6 we discuss some possible future direction of research.

2. Shooting algorithms for TPBVP. In this section we discuss numerical methods for solving the TPBVP (1.5)-(1.7).

The algebraic differential system (1.5)-(1.6) is said to be of *first order* if the partial derivative $D_v g(t, x, v)$ (that may be viewed as an $n_v \times n_v$ matrix depending on (t, x, v)) is

invertible for all (t,x,v). Otherwise (1.6) is said to be of *higher order*. In this paper we deal mainly with first order algebraic differential systems. In that case, the implicit function theorem says that (1.6) is locally equivalent to express locally v as a function of x and t, say $v = V(x,t)$. Additionally, differentiating the algebraic constraint, after substitution of v, with respect to t and x, we obtain the relations

$$0 = D_t g(t,x,V(x,t)) + D_v g(t,x,V(x,t)) D_t V(x,t), \tag{2.1}$$

$$0 = D_x g(t,x,V(x,t)) + D_v g(t,x,V(x,t)) D_x V(x,t), \tag{2.2}$$

that uniquely determine the partial derivatives of V, since $D_v g(t,x,V(x,t))$ is invertible. For first order algebraic differential systems, the algebraic differential system (1.5)-(1.6) is then equivalent to the *reduced differential system*

$$\dot{x} = h(t,x(t),V(x,t)), \quad t \in [0,T]. \tag{2.3}$$

Note that in general V depends on t even if g does not, since the equation $g(t,x,v) = 0$ may have several solutions.

The basic idea of the *shooting algorithm* is to express the final value $x(T)$ as a function of the initial value $x(0)$ through the algebraic differential system (1.5)-(1.6). Let us denote

$$x(T) = \varphi(x_0) \tag{2.4}$$

the mapping that with the initial value denoted x_0 associates the final value $x(T)$. For a first order algebraic differential system, this is well defined iff the solution of (2.3) with initial condition x_0 is. Therefore, φ is smooth over its domain of definition. It follows that the TPBVP is equivalent to the equation

$$B(x_0, \varphi(x_0)) = 0. \tag{2.5}$$

By *shooting method* we mean some algorithm applied to the problem of finding a zero to the function $F(x_0) := B(x_0, \varphi(x_0))$. The standard choice is NEWTON's method, that requires the evaluation of the Jacobian of F; clearly

$$DF(x_0) = D_{x_0} B(x_0, \varphi(x_0)) + D_{x_T} B(x_0, \varphi(x_0)) D\varphi(x_0), \tag{2.6}$$

so that the question is how to compute the Jacobian of φ. We denote

$$M_x(t) = \frac{\partial x(t)}{\partial x_0}, \qquad M_v(t) = \frac{\partial v(t)}{\partial x_0}. \tag{2.7}$$

The following theorem holds [2]:

THEOREM 2.1. *Let $x(t)$ and $v(t)$ be solutions of the algebrico-differential system (1.5)-(1.6) with initial condition x_0 and $v_0 = V(0,x_0)$. Assume this algebrico-differential system to*

be of first order. Then $x(t)$ and $v(t)$ are smooth functions of x_0, and there partial derivatives, denoted by $M_x(t)$ and $M_v(t)$, satisfy

$$\dot{M}_x(t) = D_x h(t, x(t), v(t)) M_x(t) + D_v h(t, x(t), v(t)) M_v(t), \quad t \in [0, T], \tag{2.8}$$

$$0 = D_x g(t, x(t), v(t)) M_x(t) + D_v g(t, x(t), v(t)) M_v(t), \quad t \in [0, T], \tag{2.9}$$

$$M_x(0) = Id. \tag{2.10}$$

Proof. We give a sketch of the proof of this classical result in order to make the paper self-contained. Let e_i be the ith element of natural basis of $\mathbb{R}^{n_x}$. Denote by $x_t(x_0)$ and $v_t(x_0)$ the solutions of the algebrico-differential system at instant t with initial condition x_0, and by $\delta(t)$ and $\Delta(t)$ their derivative with respect to e_i. Let $\varepsilon \in \mathbb{R}$. We have

$$\dot{x_t}(x_0 + \varepsilon\, e_i) = h(x_t(x_0 + \varepsilon\, e_i), v_t(x_0 + \varepsilon\, e_i)) \tag{2.11}$$

$$= h(x_t(x_0) + \varepsilon\, \delta(t), v_t(x_0) + \varepsilon\, \Delta(t)) \tag{2.12}$$

$$= h(x_t(x_0), v_t(x_0)) + \varepsilon\, D_x h(x_t(x_0), v_t(x_0))\, \delta(t) \tag{2.13}$$

$$+ \varepsilon\, D_v h(x_t(x_0), v_t(x_0))\, \Delta(t) + o(\varepsilon). \tag{2.14}$$

At the same time, we have that

$$\dot{x_t}(x_0 + \varepsilon\, e_i) = \dot{x_t}(x_0) + \varepsilon\, \dot{\delta}(t) + o(\varepsilon). \tag{2.15}$$

Identifying the terms in ε, we obtain

$$\dot{\delta}(t) = D_x h(x_t(x_0), v_t(x_0))\, \delta(t) + D_v h(x_t(x_0), v_t(x_0))\, \Delta(t), \tag{2.16}$$

and by the same argument applied to $g(x, v) = 0$, we have

$$D_x g(x_t(x_0), v_t(x_0))\, \delta(t) + D_v g(x_t(x_0), v_t(x_0))\, \Delta(t) = 0. \tag{2.17}$$

Since this holds for each i in $1, \ldots, n$, the result follows. □

2.1. Newton's method. In this section it is convenient to change slightly the notation, and to consider the problem of solving a nonlinear equation $G(x) = 0$ in $\mathbb{R}^n$, where G is a smooth function $\mathbb{R}^n \to \mathbb{R}^n$. Newton's method consists in computing a sequence satisfying

$$G(x^k) + DG(x^k)(x^{k+1} - x^k) = 0. \tag{2.18}$$

We consider a version of Newton's method with linesearch; namely, at step k, compute $d^k \in \mathbb{R}^n$ solution of

$$G(x^k) + DG(x^k) d^k = 0, \tag{2.19}$$

and then

$$x^{k+1} = x^k + \rho_k d^k, \tag{2.20}$$

for some $\rho_k > 0$; if $\rho_k = 1$, we recover the standard Newton's method. The usual way for choosing the stepsize ρ_k is based on the decrease of the least square criterion

$$\gamma(x) := \tfrac{1}{2}\|G(x)\|^2. \tag{2.21}$$

Due to (2.19), we have that

$$\frac{\mathrm{d}}{\mathrm{d}\rho}\gamma(x^k + \rho d^k) = -\|G(x^k)\|^2 = -2\gamma(x^k). \tag{2.22}$$

Given two parameters β and θ in $(0,1)$, it is therefore customary to take ρ_k following Armijo's rule [?]: the first value 1, β, β^2, ..., such that

$$\gamma(x^k + \rho_k d^k) \leq (1 - \rho_k\theta)\gamma(x^k). \tag{2.23}$$

If $\bar{x}$ is a regular zero of G, in the sense that $G(\bar{x}) = 0$ and $DG(\bar{x})$ is invertible, then if x^0 is close enough to $\bar{x}$, we have that Newton's method with linesearch is such that ρ_k is always 1, i.e., it generates the same sequence as the standard Newton's method.

2.2. The multiple shooting method. The multiple shooting method is the following variant of the (simple) shooting method. The interval $[0,T]$ is splitted into r subintervals $[t_{k-1}, t_k]$, for $k = 0, \ldots, r-1$, with

$$0 = t_0 < t_1 < \ldots t_{r-1} < t_r = T. \tag{2.24}$$

Consider the mappings φ_k, $k = 0, \ldots, r-1$, that with x^k (a candidate for the value of $x(t_k)$) associate the value $\varphi_k(x^k)$ obtained by integrating (1.5)-(1.7) over $[t_k, t_{k+1}]$, with initial condition $x(t_k) = x^k$. Then the TPBVP (1.5)-(1.7) is obviously equivalent to

$$\left\{ \begin{array}{lcl} x(t_{k+1}) & = & \varphi_k(x_k), \quad k = 0, \ldots, r-1, \\ B(x(0), x(T)) & = & 0. \end{array} \right. \tag{2.25}$$

We have now a system of $r\, n_x$ equations and unknowns, instead of n_x for the simple shooting method. This bigger system often appears to be better conditioned than the one obtained in simple shooting methods.

3. The optimal control problem as a TPBVP. In this section we recall how the optimal control problem (1.1)-(1.4) can be cast in the format of the TPBVP (1.5)-(1.7). Define the hamiltonian function H and the "initial-final Lagrangian" $\mathcal{H}$ as follows:

$$H(t, y, u, p) = L(t, y, u) + p^T \cdot f(t, y, u) \tag{3.1}$$

$$\mathcal{H}(y_0, y_T, \lambda) = F(y_0, y_T) + \lambda \cdot A(y_0, y_T) \tag{3.2}$$

We have then the theorem (see [1]):

THEOREM 3.1. *The fist order optimality system of the optimal control problem (1.1)-(1.4) is as follows: there exists* $\lambda \in \mathbb{R}^q$, $\mu \in \mathbb{R}^{|I|}$, *and* $p(t) \in W^{1,\infty}(0,T)$ *such that*

$$\dot{y} = D_p H(t, y(t), u(t), p(t)), \quad t \in [0,T], \tag{3.3}$$

$$-\dot{p} = D_y H(t, y(t), u(t), p(t)), \quad t \in [0,T], \tag{3.4}$$

$$0 = D_u H(t, y(t), u(t), p(t)), \quad t \in [0,T], \tag{3.5}$$

$$0 = A(y(0), y(T)), \tag{3.6}$$

$$y_i(0) = (y_0)_i, \quad \textit{for all} \quad i \in I, \tag{3.7}$$

$$D_{y_T} \mathcal{H}(y(0), y(T), \lambda) = p(T), \tag{3.8}$$

$$(D_{y_0} \mathcal{H}(y(0), y(T), \lambda))_j = -p_j(0), \quad \textit{for all} \quad j \notin I. \tag{3.9}$$

Proof. Denoting by μ the Lagrange multiplier associated with (1.4), we have that the Lagrangian of the optimal control problem (1.1) is

$$\begin{aligned} \mathcal{L}(y,u,p,\lambda,\mu) \quad := \quad & \int_0^T L(t, y(t), u(t))\, dt + F(y(0), y(T)) \\ & + \int_0^T p(t) \cdot (f(t, y(t), u(t)) - \dot{y}(t))\, dt \\ & + \lambda \cdot A(y(0), y(T)) + \sum_{i \in I} \mu_i (y_i(0) - (y_0)_i), \end{aligned} \tag{3.10}$$

or equivalently, using (3.1)-(3.2),

$$\begin{aligned} \mathcal{L}(y,u,p,\lambda,\mu) \quad := \quad & \int_0^T H(t, y(t), u(t), p(t))\, dt + \mathcal{H}(y(0), y(T), \lambda) \\ & - \int_0^T p(t) \cdot \dot{y}(t)\, dt + \sum_{i \in I} \mu_i (y_i(0) - (y_0)_i). \end{aligned} \tag{3.11}$$

Clearly then,

$$D_u \mathcal{L}(y,u,p,\lambda,\mu) v = \int_0^T D_u H(t, y(t), u(t), p(t))\, v(t)\, dt. \tag{3.12}$$

Also, after an integration by parts, we obtain

$$\begin{aligned} D_y \mathcal{L}(y,u,p,\lambda,\mu) z \quad = \quad & \int_0^T (D_y H(t, y(t), u(t), p(t)) + \dot{p}(t)) \cdot z(t)\, dt \\ & + (D_{y_0} \mathcal{H}(y(0), y(T), \lambda) + p(0)) z(0) + \sum_{i \in I} \mu_i\, z_i(0) \\ & + (D_{y_T} \mathcal{H}(y(0), y(T), \lambda) - p(T)) z(T). \end{aligned} \tag{3.13}$$

By classical arguments this amount is zero iff (3.4), (3.8), and (3.9) hold. The result follows. □

name	*function*	*language*
OCS_input.maple	declaration	MAPLE
*OCS_maple*1	generation	MAPLE
OCS_input.sci	initialization	SCILAB
*OCS_scilab*1*.sci*	simple shooting method	SCILAB
*OCS_scilab*2*.sci*	multiple shooting method	SCILAB
OCS_init.sci	initialization	SCILAB

TABLE 4.1
Table of the different softwares

4. Description of the software. The suffix .maple or .sci indicates whether the file is written in MAPLE or SCILAB language. For a given application, the user provides two files:

- *OCS_input.maple* defines all non numeric data of the application: names of constant parameters, state variables, and control variables, as well as the nonlinear functions: system dynamics, distributed cost, performance criterion, and constraints between initial state and final state.
- *OCS_input.sci* gives all needed numerical data: values of constant parameters, values or estimates for state, costate and control variables at initial time, and the constraints multiplier λ.

Here are the generic files:

- *OCS_maple*1, processes the user file *OCS_input.maple*, and generates the adjoint dynamics as well as some pieces of the LATEX report. Also, using the MACROFORT software [], it generates several FORTRAN routines allowing the numerical computation of the various nonlinear functions as well as their first and second order derivatives. (The computation of derivatives is required by the SCILAB integrators IMPL and DASSL.)
- *OCS_scilab*1*.sci* solves the optimal control problem by the shooting method.
- *OCS_scilab*2*.sci* solves the optimal control problem by the multiple shooting method.
- *OCS_init.sci* computes values of the controls and its time derivatives at initial time.

The softwares written in SCILAB are all included in one file *OCS_solver.sci* except *OCS_input.sci*. The softwares are run with version V release 5 of MAPLE (mapleV5) and version 2.5 of SCILAB (scilab-2.5) and were tested on three Operating Systems: SOLARIS, LINUX(on DEL), UNIX(on DEC).

5. Numerical study. The material of the two next subsections is generated automatically by OCS. Note that the symbol of costate variables are concatenation of a p with the name of the state variable. We give two examples of resolution of optimal control problems.

5.1. An example for TPBVP with periodicity: Oscillator example.

5.1.1. The optimal control problem and the differential equations.

Setting of the optimal control problem:

. - Constant parameters :
Ma : coefficient 1
$raid$: coefficient 2
A : amplitude
- State variables :
h : angle of deviation. Unit: rd
v : speed. Unit : rd per sec
tt : time Unit : sec
As the data depend on time t, one considers t as a third variable of state tt
- Control variables :
u : control.
Differential equations of the optimal control problem (1) :

$$\dot{h} = v \tag{5.1}$$

$$\dot{v} = \left(A \sin(7/6 \sqrt{\frac{raid}{Ma}} tt) + A \sin(5/6 \sqrt{\frac{raid}{Ma}} tt) - raid\, h \right) Ma^{-1} + u \tag{5.2}$$

$$\dot{tt} = 1 \tag{5.3}$$

Distributed cost :

$$1/2\, u^2 + 1/2\, h^2 \tag{5.4}$$

Final cost :

$$[0] \tag{5.5}$$

Final constraints :

$$h - h0 = 0 \tag{5.6}$$

$$v - v0 = 0 \tag{5.7}$$

Costate equations:

. Costate variables : ph , pv , ptt
Costate equations :

$$\dot{ph} = -h + \frac{pv\ raid}{Ma} \tag{5.8}$$

$$\dot{pv} = -ph \tag{5.9}$$

$$\dot{ptt} = -pv\left(7/6\,A\cos(7/6\sqrt{\frac{raid}{Ma}}tt)\sqrt{\frac{raid}{Ma}} + 5/6\,A\cos(5/6\sqrt{\frac{raid}{Ma}}tt)\sqrt{\frac{raid}{Ma}}\right)Ma^{-1} \tag{5.10}$$

Differential system after elimination of control:

. From the Pontryaguine's principle one deduces :

$$u = -pv \tag{5.11}$$

Resulting state-costate system (2) :

$$\dot{h} = v \tag{5.12}$$

$$\dot{v} = \left(A\sin(7/6\sqrt{\frac{raid}{Ma}}tt) + A\sin(5/6\sqrt{\frac{raid}{Ma}}tt) - raid\;h\right)Ma^{-1} - pv \tag{5.13}$$

$$\dot{tt} = 1 \tag{5.14}$$

$$\dot{ph} = -h + \frac{pv\;raid}{Ma} \tag{5.15}$$

$$\dot{pv} = -ph \tag{5.16}$$

$$\dot{ptt} = -pv\left(7/6\,A\cos(7/6\sqrt{\frac{raid}{Ma}}tt)\sqrt{\frac{raid}{Ma}} + 5/6\,A\cos(5/6\sqrt{\frac{raid}{Ma}}tt)\sqrt{\frac{raid}{Ma}}\right)Ma^{-1} \tag{5.17}$$

5.1.2. Numerical study.

Values of the constants:

. The different constants take the values:

$$Ma = 1 \tag{5.18}$$

$$raid = 1 \tag{5.19}$$

$$A = 0.5 \tag{5.20}$$

The solution is computed with the multiple shooting algorithm taking $r = 10$.

Behavior of the algorithm:
. Initial value of costate at initial time (found with successive tests):

$$ph_0 = -.6182200000000D+00$$
$$pv_0 = -.2202365000000D+00$$
$$ptt_0 = 0.0000000000000D+00$$

Initial value of shooting function f1 = 0.2247123316671D+01

Final value of shooting function f1 = 0.60307857D-06
Number of NEWTON steps N1 = 3
Final value of costate at initial time :

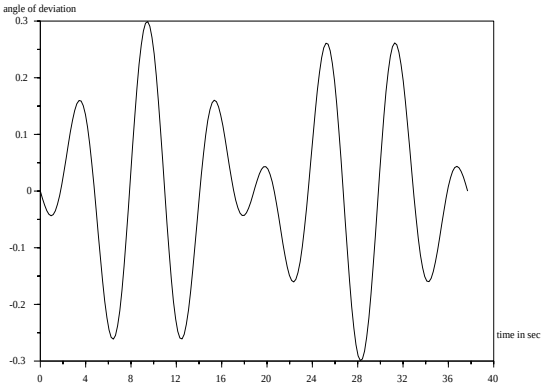
$$ph_0 = -.8971284605667D+00$$
$$pv_0 = 0.2044524307681D-06$$
$$ptt_0 = 0.7973904427432D-05$$

Display of the solutions:
. The evolution with respect to time of the state and control variables and the evolution of the log of the norm of shooting function and the evolution of ρ with respect to NEWTON's iterations are plotted on the following pictures:

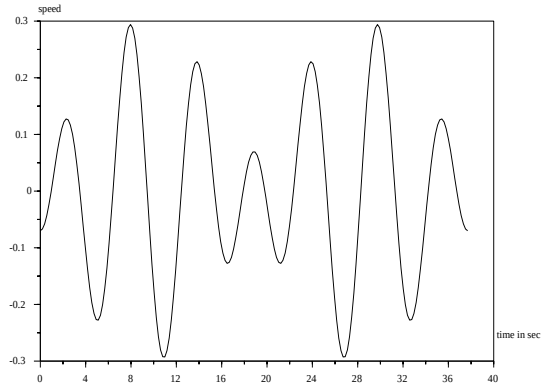
5.2. An example of optimal control problem: the flight of Apollo type vehicle.

5.2.1. The optimal control problem and the differential equations.

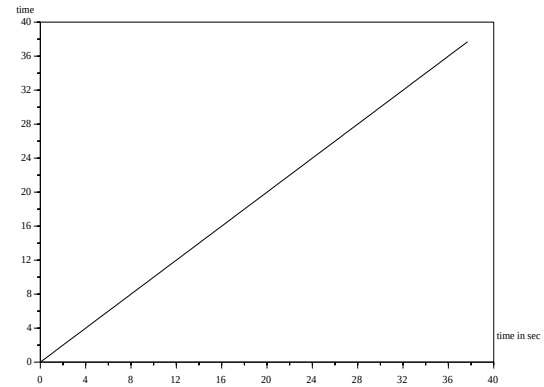
Setting of the optimal control problem:
. - Constant parameters :
cD : coefficient to aerodynamical drag coefficient
cL : coefficient to aerodynamical lift coefficient
g : gravitational acceleration
$rh1$: coefficient in cost function
$rh0$: atmospheric density at h=0
β : coeff beta
$c1$: coefficient1



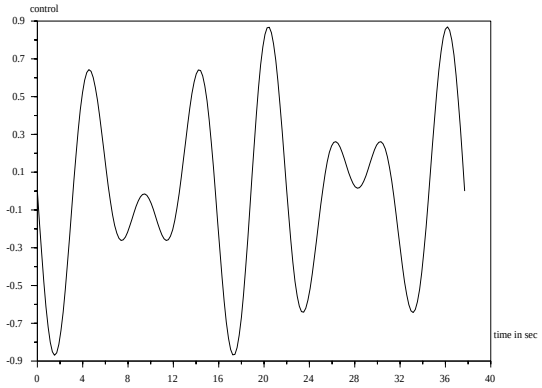
angle of deviation.



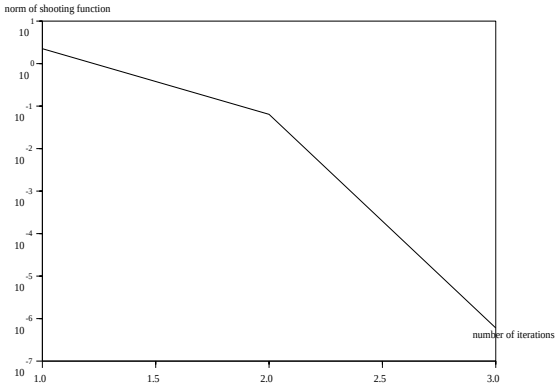
speed.



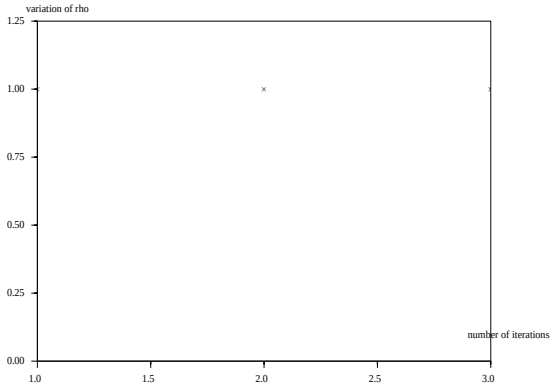
time.



control.



norm of shooting function w.r.t iterations.



rho function of iterations.

$c2$: coefficient2
- State variables :
v : speed. Unit: 10^5 ft per sec
γ : flight-path angle. Unit: rad
h : altitude. Unit: 10^5 ft
- Intermediate functions :
normalized atmospheric density

$$e^{-\beta\, h} \tag{5.21}$$

- Control variables :
u : angle of attack. Unit: rad
Differential equations of the optimal control problem (1) :

$$\dot{v} = -cD\ rh0\ e^{-\beta\, h}\,(c1\ -\ c2\ \cos(u))\, v^2 - g\sin(\gamma)\left(1+\frac{1}{209}\,h\right)^{-2} \tag{5.22}$$

$$\dot{\gamma} = cL\, rh0\, e^{-\beta\, h}\sin(u)v - g\cos(\gamma)v^{-1}\left(1+\frac{1}{209}\,h\right)^{-2} + \frac{v\cos(\gamma)}{209+h} \tag{5.23}$$

$$\dot{h} = v\sin(\gamma) \tag{5.24}$$

Distributed cost :

$$1/3\ rh1\ \sqrt{e^{-\beta\, h}}v^3 \tag{5.25}$$

Final cost :

$$[10\, v^2 + 1/2\,\gamma^2 + 1/2\, h^2 + 1/2\, v0^2 + 1/2\,\gamma 0^2] \tag{5.26}$$

Final constraints :

$$\gamma = 0 \tag{5.27}$$

$$h = 0 \tag{5.28}$$

5.2.2. Numerical study.

Values of the constants:

. The different constants take the values:

$$cD = 26.6 \tag{5.29}$$

$$cL = 15.96 \tag{5.30}$$

$$g = 0.32172d - 3 \tag{5.31}$$

$$rh1 = 1.56 \tag{5.32}$$

$$rh0 = 2.704d - 3 \tag{5.33}$$

$$\beta = 4.26 \tag{5.34}$$

$$c1 = 1.174 \tag{5.35}$$

$$c2 = 0.9 \tag{5.36}$$

The solution is computed with the multiple shooting algorithm taking $r = 20$.

Behavior of the algorithm:

. Initial value of costate at initial time (found with successive tests):

$$pv_0 = 0.1007378600000D + 02$$
$$pgama_0 = 0.5680274000000D + 01$$
$$ph_0 = 0.1139820000000D + 00$$

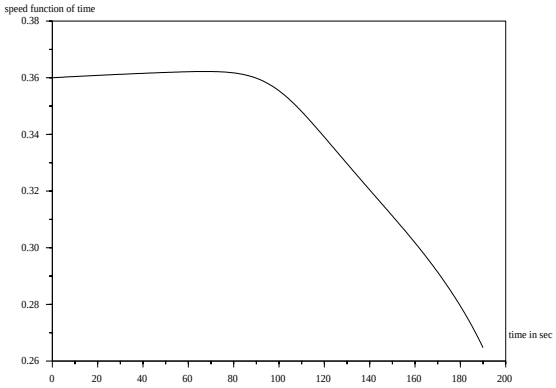
Initial value of shooting function f1 = 0.1646604586143D+01

Final value of shooting function f1 = 0.55584789D-06
Number of NEWTON steps N1 = 21
Final value of costate at initial time :

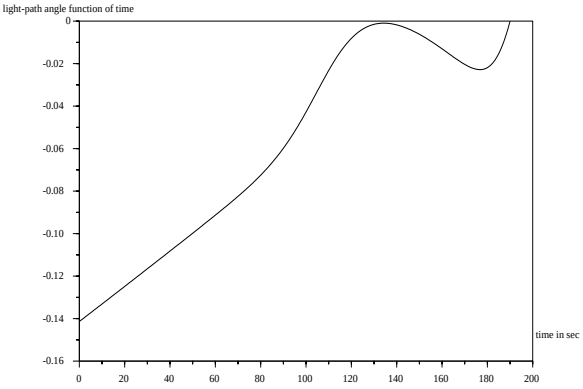
$$pv_0 = 0.1448946316506D + 02$$
$$pgama_0 = 0.4047872758437D + 01$$
$$ph_0 = 0.0000000000000D + 00$$

Display of the solutions:

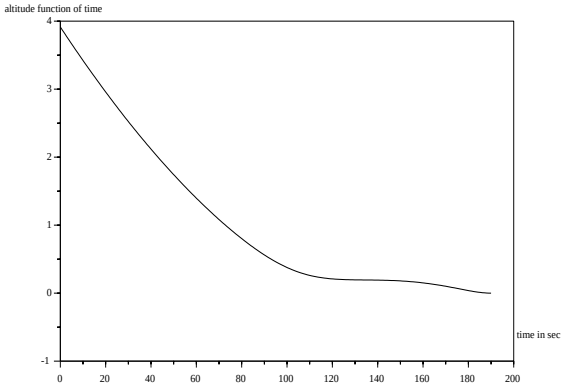
. The evolution with respect to time of the state and control variables and the evolution of the log of the norm of shooting function and the evolution of ρ with respect to NEWTON's iterations are plotted on the following pictures:



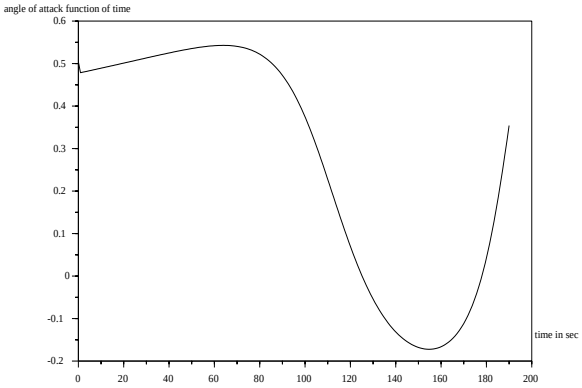
speed.



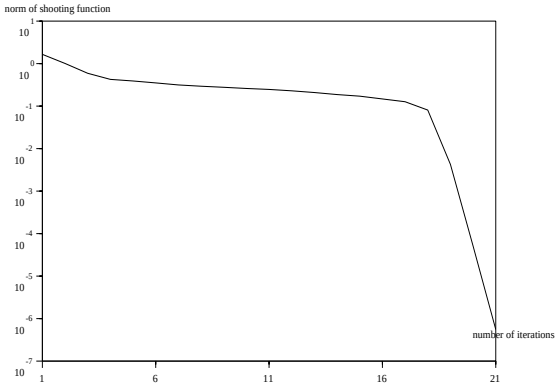
flight-path angle.



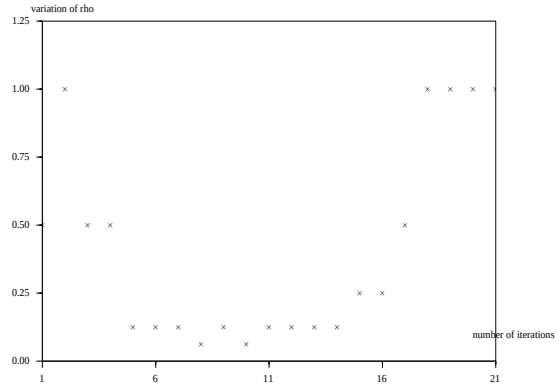
altitude.



angle of attack.



norm of shooting function w.r.t iterations.



rho function of iterations.

6. Getting the software. The software is available on the web site www-rocq.inria.fr/mocoa. Comments are welcome.

Appendix A. Structure of the software.

A.1. An example of input file in scilab. As an example of input file in scilab let us print the file *OCS_input.sci* of Apollo flight case. It shows how are initialized the different data as the constants,the state variables, the costate variables etc...and how are given the values of the different algorithmic parameters.

```
//                      FILE OCS_input.sci
// INPUT FILE (NUMERICAL DATA) FOR OCS SOLVER : problem flight0

//DATA
//-------

//VALUE OF CONSTANT PARAMETERS
//-----------------------------
// constants : cD, cL, g1, rh1, rh0, beta, c1, c2
cD=26.6d0;
cL=26.6*0.6d0;
g1=0.32172d-3;
rh1=1.56;
rh0=2.704d-3;
beta=4.26;
c1=1.174;
c2=0.9;
tr2=[cD;cL;g1;rh1;rh0;beta;c1;c2];

// INITIAL AND FINAL TIME (t0 and tf)
//-----------------------------------
t0=0.0;
tf=190;


// DATA AND ALGORITHMIC INITIALIZATIONS
//-------------------------------------

// INITIAL STATE
//---------------

//v0 initial speed
v0=0.36D+00;
//gam0 initial flight path angle
gam0=-0.1413716694D+00;
//h0 initial altitude
h0=1.0D+00;
```

```
y0=[v0;gam0;h0];

// INITIAL COSTATE
//----------------

//pv0 first component of costate
pv0=10.073786;
//pgam0 second component of costate
pgam0=5.680274;
//ph0 third component of costate
ph0=0.113982;

p0=[pv0;pgam0;ph0];

// INITIALIZATION OF CONTROL
//--------------------------
//u1 : vector of control variables
u1(1)=1.0d0

// INITIALIZATION OF LAGRANGE MULTIPLIERS
//---------------------------------------

lambda=[1;1];

// ALGORITHMIC PARAMETERS
//-----------------------

//r number of intervals in multiple shooting algorithm
//-----------------------------------------------------
//r=0 simple shooting algorithm
r=20;

// number of points for plotting
//------------------------------
//(optional,default value=200)
//----------------------------

nsteps=200;

// stopping criterion of value of norm of shooting function
//---------------------------------------------------------
```

```
//(optional,default value=1.0d-10)
//-------------------------------

epsf=1.0d-11;

// maximum number of shooting (Newton) iterations
//-----------------------------------------------
//(optional, default value=40)
//----------------------------

N=40;


// VALUE OF THE PARAMETER DEFINING THE METHOD
//-------------------------------------------
// USED TO COMPUTE THE JACOBIAN OF THE FINAL
//-----------------------------------------
// STATE AND COSTATE WITH RESPECT TO THE INITIAL
//----------------------------------------------
// VALUES OF INITIAL STATE AND COSTATE
//-----------------------------------
//(compjac=1 exact jacobian,
//---------------------------
// compjac=2 finite differences)
//------------------------------

compjac=1;

// VALUE OF THE CHARACTER STRING DEFINING
//---------------------------------------
// THE CHOICE OF ALGEBRAIC DIFFERENTIAL SOLVER
//--------------------------------------------
// (ads='dassl' DASSL SOLVER,
//---------------------------
//  ads='impl' IMPL SOLVER)
//------------------------

ads='dassl';

// VALUE OF THE CHARACTER STRING DEFINING
//---------------------------------------
// THE METHOD USED TO INVERT HESSIAN OF HAMILTONIAN
```

```
//-----------------------------------------------
// w.r.t CONTROL
//--------------
//(invhes='formal' the inverse is formal
//---------------------------------------
// invhes='numerical' inverse is numerical)
//---------------------------------------

invhes='numerical';
```

A.2. An example of input file in maple. As an example of input file in maple let us print the file *OCS_input.maple* of Apollo flight case. It shows how the different names are given to the state variables, the control variables, how are described the dynamics, the distributed cost, the final cost etc...

```
#                          FILE OCS_input.maple
# INPUT FILE (SYMBOLIC DATA) FOR OCS SOLVER : problem flight0
#################################################################
#
#Frederic Bonnans ; Sady Maurin ; INRIA december 1999
#####################################################
#
#
#
#Constants
##########
#
OCS_cons:=[cD,cL,g,rh1,rh0,beta,c1,c2]:
#
OCS_cons_names:=[`coefficient to aerodynamical drag coefficient`,
                 `coefficient to aerodynamical lift coefficient`,
                 `gravitational acceleration`,
                 `coefficient in cost function`,
                 `atmospheric density at h=0`,
                 `coeff beta`,`coefficient1`,`coefficient2`]:
#
#
#State variables
################
#
OCS_state:=[v,gama,h]:
#
OCS_state_names:=[`speed. Unit: ft per sec`,
```

```
                 'flight-path angle. Unit: rad',
                 'altitude. Unit: ft']:
#
#
#legend of figures for state variables
#####################################
#
OCS_figy_leg:=['speed function of time',
               'flight-path angle function of time',
               'altitude function of time']:
#
#
#Control variables
##################
#
OCS_control:=[u]:
#
OCS_control_names:=['angle of attack. Unit: rad']:
#
#
#legend of figures for control variables
########################################
#
OCS_figu_leg:=['angle of attack function of time']:
#
#
#Intermediate functions :
#########################
#
OCS_fint:=[exp(-beta*h)];
#
rho:=OCS_fint[1]:
#
OCS_fint_names:=['normalized atmospheric density']:
#
#
#Dynamics :
###########
#
OCS_dynamics2:=[ -cD*rh0*rho*(c1-c2*cos(u))*v**2
                 -g*sin(gama)/(1+h/209)**2,
                 cL*rh0*rho*sin(u)*v-g*cos(gama)/(v*(1+h/209)**2)
```

```
                +v*cos(gama)/(209*(1+h/209)),
                v*sin(gama)]:
#
#
#Distributed cost:
###################
#
OCS_cost:=rh1*sqrt(rho)*v**3/3:
#
#
#
#final cost :
#############
#
OCS_fcost:=array([10*v^2+1/2*gama^2+1/2*h^2+1/2*v0^2+1/2*gama0^2]);
#
#
#Initial vector of state variables
###################################
#
y0:=[v0,gama0,h0];
#
#
#Final state constraints
#########################
#
OCS_fcont:=[gama,h];
#
#
#Vector indicating which components of initial
###############################################
#state are fixed or free : 0 if fixed, 1 if free
###############################################
#
lindm:=[0,0,1];
```

REFERENCES

[1] Bryson A.E. and Ho Y.C. *Applied optimal control.* Hemisphere Publishing, New-York, 1975.
[2] Stoer J. and Bulirsch R. *Introduction to Numerical Analysis.* Springer-Verlag, New-York Heidelberg Berlin, 1983.
[3] Guilbaud Thérèse. *Interior point method and Shooting method.* PhD thesis, Paris VI University, 1999.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399