Resolution numerique de problemes de controle optimal par une methode homotopique simpliciale

Pierre Martinon

Numerical resolution of optimal control problems by a Piecewise Linear continuation method.

Pierre Martinon

November 2005

Contents

1 Shooting method and continuation **7**
- 1.1 Principle and first applications . . . 7
- 1.2 Some convergence results . . . 20

2 PL continuation - Simplicial method **25**
- 2.1 Simplicial methods principle . . . 25
- 2.2 Junction homotopy . . . 32
- 2.3 Adaptive triangulation . . . 37
- 2.4 Solution refining . . . 51

3 Orbital Transfer problems **55**
- 3.1 Problem statement . . . 56
- 3.2 Resolution approach . . . 59
- 3.3 Comparison of different integrators . . . 63
- 3.4 Adaptive triangulation study . . . 72

4 Singular Arcs problems **85**
- 4.1 Problems statement . . . 86
- 4.2 Quadratic perturbation . . . 90
- 4.3 Discretized BVP approach . . . 99
- 4.4 Structured shooting . . . 109
- 4.5 Discretized Control formulation . . . 119
- 4.6 Direct method approach . . . 124

A Simplicial package overview **131**
- A.1 Using the Simplicial package . . . 133
- A.2 General code structure . . . 142

B Main options **145**
- B.1 Problem class . . . 145
- B.2 Labelings . . . 145
- B.3 Integrators . . . 146

C Sample files for some studied problems **151**
C.1 Demonstration problem 151
C.2 Orbital transfer problems 155
C.3 Optimal harvesting in fishery 158
C.4 Quadratic regulator 160

Introduction

This thesis work deals with the numerical resolution of optimal control problems with a low regularity. Indirect methods, based on the Pontryagin's Maximum Principle, are known to be a fast and accurate way of treating optimal control problems. However, their practical application can encounter some difficulties, especially the critical choice of the initial point, when the control structure is bang-bang or presents singular arcs. We handle here these difficulties with a continuation approach, and we choose more specifically to use simplicial methods due to their robustness.

We begin in chapter I with a presentation of the general principle of indirect methods, and more precisely the single shooting, that consists in transforming the optimal control problem into finding a zero of the associated shooting function. This method is illustrated on a simple example, which allows us to highlight some of the difficulties expected in the case of a bang-bang structure. We introduce then a continuation approach, whose broad lines we recall (namely following the zero path of a certain homotopy), and pursue the resolution of the previous demonstration problem. We conclude by recalling some convergence results concerning the continuation, that also hold in the multi-valued case (for singular arcs for instance).

Chapter II is devoted to the description of the simplicial method, whose main advantage is its low requirements concerning the path to follow. We recall first the general principal of a simplicial method, then present some algorithmic points we introduced in this context, such as adaptive meshsize and solution refining mechanisms.

We focus in chapter III on the study of an orbital transfer problem provided by the CNES, for which the results tie up with those obtained by a differential continuation (see [26]). We also compare on this problem several variable step integrators, as well as different settings for the adaptive meshsize mechanism.

In chapter IV we study two singular arcs problems in parallel, aiming for a better generality of the methods employed. The first phase consists

here in gathering sufficient information about the control structure, by the use of relevant continuations. More precisely, we try to determine both the number (and localization) of singular arcs, and the nature of remaining bang-bang arcs. Then we move on to the precise resolution of the problems, by methods derived from multiple shooting and adapted to the singular arcs case. To finish with, we present some preliminary numerical results obtained with a new formulation that considers the discretized control as part of the unknown (more in the spirit of direct methods).

After the conclusion and future perspectives of this work, we briefly detail in the appendix the various integrators used, as well as the Simplicial code, in which we have implemented all the continuations and formulations used in the numerical experiments.

Chapter 1

Shooting method and continuation

To begin with, we recall the general principle of the shooting method, which transforms the original optimal control problem into the resolution of a nonlinear equation. We then detail the practical application of the shooting method on a simple demonstration problem, to illustrate some of the difficulties that can be expected for the problems studied in chapters 3 and 4. These difficulties lead us to use a continuation (or *homotopic*) approach, which is detailed later in chapter 2.

1.1 Principle and first applications

1.1.1 Single shooting method

We make here a brief presentation of the single shooting method, which is part of the *indirect* methods, and is based on Pontryagin's Maximum Principle (we refer readers interested in these methods to [32, 18, 4, 35]). We recall that *direct* methods, on the other hand, typically involve the partial or total discretization of the problem, and then use various approaches (SQP and interior point techniques for instance) to solve the resulting problem. Direct methods are thus supposed to be robust, but tend to have a relatively low precision, and a huge problem size depending on the discretization stepsize used. This makes these methods ill-suited to some particular cases, such as the problems studied in chapter 3, which present a bang-bang control structure with a huge number of commutations.

Single shooting, on the other hand, consists in finding a zero of the *shooting function* associated with the original problem. There is no discretization, even if the method still involves an integration of the system in some way. It is a fast and high precision method, that does not require any assumptions

on the control structure of the problem.

Yet a major drawback of this class of methods is that they require a good initial guess: as they typically consist in applying a Quasi-Newton solver to the shooting function, the convergence radius may be quite small, depending on the problem. This is especially true when the shooting function has is not smooth, which is the case for the bang-bang orbital transfers studied in chapter 3 or the singular arc problems studied in chapter 4. This is why we use a continuation approach to obtain a suitable initial point, which is described in chapter 2.

Necessary conditions, PMP and Boundary Value Problem

Notation/Remark: for clarity, the time t will often be omitted in the formulas, except in the ambiguous cases.

We consider a general optimal control problem in the Bolza form

$$(P)\begin{cases} Min\ g(t_0,x(t_0),t_f,x(t_f)) + \int_{t_0}^{t_f} l(t,x,u)\ dt & Objective \\ \dot{x} = f(t,x,u) & Dynamics \\ u \in U & Admissible\ Controls \\ \psi_0(t_0,x(t_0)) = 0 & Initial\ Conditions \\ \psi_1(t_f,x(t_f)) = 0 & Terminal\ Conditions \end{cases}$$

We use here and in all the following the notations: x for the state, u for the control, U the set of admissible controls, and f for the state dynamics. We introduce the costate p, of same dimension as the state x, and define the **Hamiltonian** by

$$\mathcal{H}(t,x,p,u) = l(t,x,u) + (p|f(t,x,u)).$$

Remark: we thus assume in all the following that we are in the ***normal*** *case, meaning that the costate p_0 associated to the integral objective l is non zero, and can be made equal to 1, by dividing the whole costate p by p_0...*

The Pontryagin's Maximum Principle (originally stated in [32]) then states that, under the assumptions:
• $\exists\ (x,u)$ feasible for (P), with x absolutely continuous and u measurable.
• f and l are continuous with respect to u and C^1 with respect to t and x.
• g, ψ_0, ψ_1 are C^1 with respect to x.

Let $(\overline{x},\overline{u})$ be an optimal pair for (P), then

(i) $\exists\, \overline{p} \neq 0$ absolutely continuous such that we have the Hamiltonian system

$$\begin{cases} \dot{\overline{x}} = \frac{\partial \mathcal{H}}{\partial p}(t, \overline{x}, \overline{p}, \overline{u}) \\ \\ \dot{\overline{p}} = -\frac{\partial \mathcal{H}}{\partial x}(t, \overline{x}, \overline{p}, \overline{u}) \end{cases}$$

(ii) $\overline{u}$ is solution of $Min_{w \in U} \mathcal{H}(t, x, p, w)$ ae in $[t_0, t_f]$.

(iii) "Transversality conditions": $\exists\, (\overline{\mu}_0, \overline{\mu}_1)$ such that

$$(TC) \begin{cases} \psi_0(t_0, x(t_0)) = 0 \\ \overline{p}(t_0) = -\frac{\partial \Phi}{\partial x_0}(t_0, \overline{x}(t_0), t_f, \overline{x}(t_f), \overline{\mu}_0, \overline{\mu}_1) \\ \\ \psi_1(t_f, x(t_f)) = 0 \\ \overline{p}(t_f) = \frac{\partial \Phi}{\partial x_f}(t_0, \overline{x}(t_0), t_f, \overline{x}(t_f), \overline{\mu}_0, \overline{\mu}_1) \end{cases}$$

with

$$\Phi : (t_0, x_0, t_f, x_f, \mu_0, \mu_1) \mapsto g(t_0, x_0, t_f, x_f) + (\psi_0(t_0, x_0)|\mu_0) + (\psi_1(t_f, x_f)|\mu_1)$$

Remark: this explains why indirect methods are sometimes referred to as necessary condition *methods.*

Now we denote $y = (x, p)$ and φ the state-costate dynamics derived from the Hamiltonian system. Then if Γ denotes the set valued map of optimal controls, then solving (P) is equivalent to solving the following Boundary Value Problem[1]

$$(BVP) \begin{cases} \dot{y} \in \Phi(y) = \varphi(y, \Gamma(y)) & ae\ in\ [t_0, t_f] \\ c_0(t_0, y(t_0)) = 0 & Boundary\ Conditions\ at\ t_0 \\ c_1(t_f, y(t_f)) = 0 & Boundary\ Conditions\ at\ t_f \end{cases}$$

Note: these Boundary Conditions c_0 and c_1 correspond to the Transversality Conditions mentioned above, that contain the Initial and Terminal conditions of (P) in addition to the constraints on the costate p.

Initial Value Problem and Shooting method

We assume here that the expression of the optimal control given by the necessary conditions Γ is actually a function, denoted γ. It is possible to integrate $y = (x, p)$ if we set the value of $y(t_0)$, and we then obtain the following Initial Value Problem

$$(IVP) \begin{cases} \dot{y} = \Phi(y) = \varphi(y, \gamma(y)) & ae\ in\ [t_0, t_f] \\ y(t_0) = \beta & Initial\ Value \end{cases}$$

[1] or more precisely, Two Point Boundary Value Problem $(TPBVP)$, as the boundary conditions apply only at 0 and t_f.

We introduce now an application called the **shooting function**, which basically maps the initial value β to the value of the Boundary conditions at t_f for the corresponding solution of (IVP). In practice, the initial conditions ψ_0 of the problem (P) already give a part of $y(t_0)$, so the unknown of the shooting function is reduced to the "missing" part, that we denote z. A frequent situation is when the initial conditions determine the initial state $x(t_0)$, therefore z is actually the initial costate $p(t_0)$. Then the value of the shooting function is given by the boundary conditions at t_f for the solution $\hat{y}$ of (IVP) corresponding to $y(t_0) = [x_0, z]$

$$(Shooting\ function) \qquad S : z \mapsto c_1(\hat{y}(t_f)).$$

Finding a zero of the shooting function S is then equivalent to the resolution of (BVP), and therefore also gives a solution of (P). The "shooting method" thus consists in solving the equation $S(z) = 0$.

Remark: we do not consider here the case of state constraints, which raise particular difficulties with indirect methods (see for instance [12], or [28], where several formulations of the Maximum Principle for the state constraints are described). However, this is an interesting class of problems, that we plan to study in the future.

A first simple example: bang-bang control

We present now a first simple example to illustrate the practical application of the shooting method. Moreover, this demonstration problem allows us to highlight some of the difficulties that can be expected when dealing with a bang-bang control structure, which is the case for the problems studied later in chapter 3.

1.1.2 Problem statement

Let us consider the following optimal control problem:

$$(P) \begin{cases} Min \ \int_0^2 |u| dt \\ \dot{x}_1 = x_2 \\ \dot{x}_2 = u \\ |u| \leq 1 \\ x(0) = (0, 0) \\ x(2) = (0.5, 0) \end{cases}$$

The Hamiltonian is defined by

$$\mathcal{H} : (t, x, p, u) \mapsto |u| + p_1 \ x_2 + p_2 \ u$$

and the optimal control by

$$\begin{cases} u = -sgn(p_2) & if \quad |p_2| > 1 \\ u = 0 & if \quad |p_2| < 1 \\ u = -\alpha\ p_2 & with \quad \alpha \in [0,1]\ otherwise. \end{cases}$$

So the control is discontinuous, with a bang-bang structure.
If we define the corresponding *switching function* ψ by

$$\psi : (t, x, p) \mapsto 1 - |p_2|,$$

then the commutations of the bang-bang structure are the zeros of ψ.

We have the Boundary Value Problem

$$(BVP) \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u \\ \dot{p}_1 = 0 \\ \dot{p}_2 = -p_1 \\ x(0) = (0,0) \\ x(2) = (0.5,0) \\ p(0)\ free \\ p(2)\ free \end{cases}$$

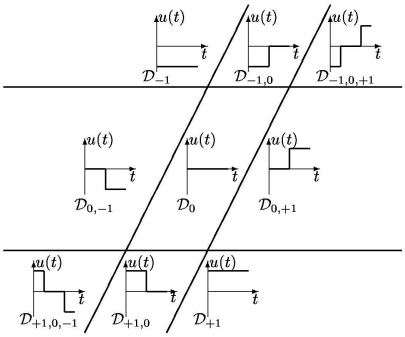
and define the shooting function

$$S(z) = \hat{x}(2) - (0.5, 0)$$

where $\hat{x}$ is the state corresponding to the integration of the Initial Value Problem

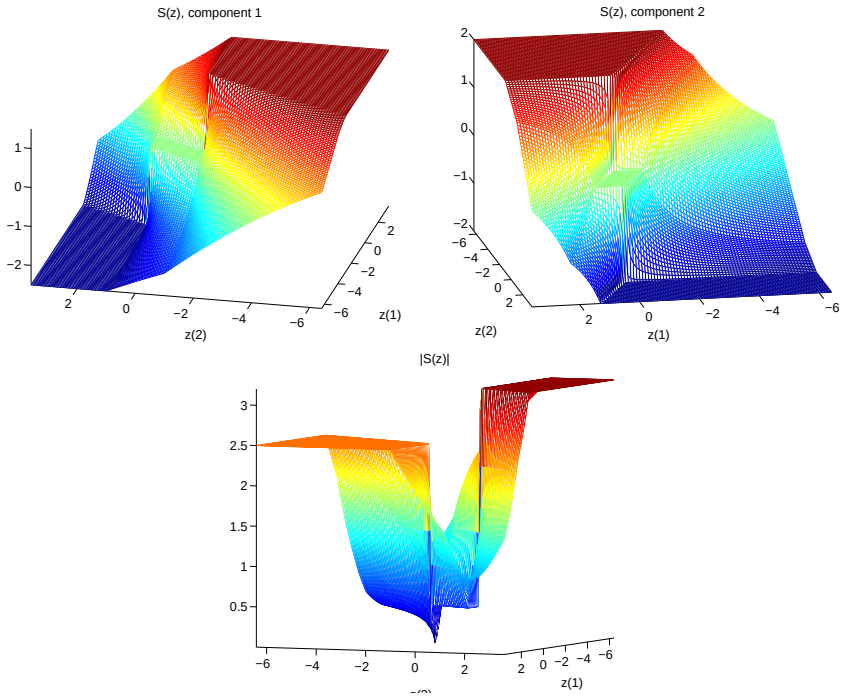
$$(IVP) \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u \\ \dot{p}_1 = 0 \\ \dot{p}_2 = -p_1 \\ x(0) = (0,0) \\ p(0) = z. \end{cases}$$

If we examine the possible values of the initial costate $p(0) = z \in \mathbf{R}^2$, we can see that the costate dynamics and optimal control expression lead to 9 different possible optimal control structures:



Control structures corresponding to the value of $p(0) \in \mathbf{R}^2$

If we represent the shooting function S, we can recognize these 9 domains



Shooting function for Problem (P)

It must be pointed out that S is not differentiable on the boundaries of these domains, and is not even defined in $(0, -1)$ and $(0, 1)$. Thus difficulties are to be expected when trying to solve $S(z) = 0$ with a Newton-like algorithm, especially if the initial guess does not lie in the correct domain. And actually, even in this very simple case, trying to solve directly $S(z) = 0$ from a random initial point is non trivial. We try below the single shooting method, with an initial point in each of the 9 domains. We indicate whether the convergence is attained ("CV" with the number of shooting function calls in parentheses) or not (-).

$p(0) = (0, 2)$ -	$p(0) = (1, 2)$ -	$p(0) = (2, 2)$ CV(61)
$p(0) = (-1, 0)$ CV(30)	$p(0) = (0, 0)$ -	$p(0) = (1, 0)$ CV(63)
$p(0) = (-2, -2)$ CV(39)	$p(0) = (-1, -2)$ CV(26)	$p(0) = (0, -2)$ -

Single shooting convergence results for various initializations

Of course, here there are only a few regions to explore, and convergence may even be attained from an initial point in the "wrong" region (at an increased iteration cost), but this is due to the simplicity of problem (P).

Nevertheless, we observe that the single shooting already fails for 4 attempts out of 9. So we understand that it is not realistic to try to solve difficult problems directly by single shooting. This is why we resort to a continuation approach to find a suitable initial point.

1.1.3 Continuation approach

As we have seen, the successful application of the shooting method can sometimes be quite tricky, as it tends to be extremely sensitive to the initial guess. This is due to the fact that they basically consist in applying a quasi-Newton method to find a zero of the shooting function. Depending on the problem, this shooting function can be highly nonlinear or non smooth, thus leading to a very small convergence radius for the shooting method in practice. This is in particular the case for problems with a bang-bang or singular control structure, the shooting function being a multi-valued map in the latter case.

A possible way to overcome these difficulties is to use a continuation approach to find a suitable initial point for the shooting method. Basically, the principle of a continuation (or *homotopic*) method is to solve a difficult problem by starting from the known solution of a somewhat related, but easier problem. By related, we mean that there must exist an application H, called a *homotopy*, with the right properties, connecting the two problems.

Notation: in all the following, we keep the notation H for the homotopy, while $\mathcal{H}$ denotes the Hamiltonian.

DEFINITION 1 *Let r and f be two applications from $\mathbf{R}^n$ into $\mathbf{R}^n$. We call an homotopy connecting r and f any application H:*

$$\begin{aligned} H : \overline{\Omega} \times [0,1] &\to \mathbf{R}^n \\ (z,\lambda) &\mapsto H(z,\lambda) \end{aligned}$$

with Ω a bounded open set in $\mathbf{R}^n$ and H continuous, so that

$$H(\cdot,0) = r \quad \textit{and} \quad H(\cdot,1) = f$$

The first task is thus to find such a proper application H for our optimal control problems. To do this, we choose to introduce the homotopic parameter λ in the criterion of the original problem. This parameterization is done such that the case $\lambda = 1$ corresponds to the unmodified problem, and that the problem for $\lambda = 0$ is much more regular and can be directly

solved by single shooting. We then obtain a family of boundary value problems $(BVP)_\lambda$ parametrized by $\lambda \in [0,1]$, and we can use the corresponding shooting functions (S_λ) for the continuation, by defining the homotopy

$$H : (z, \lambda) \mapsto S_\lambda(z).$$

Now, let us assume that we have found a solution z_0 of the starting problem $(BVP)_0$, ie a zero of $H(\cdot, 0)$. The principle of the continuation method is to start from this solution at $\lambda = 0$ to attain $\lambda = 1$, where we have a solution of the original problem.

Discrete continuation

The first idea is naturally to try to solve a sequence of problems $(BVP)_{\lambda_k}$, with a sequence (λ_k) ranging from 0 to 1. Each intermediate problem then uses the solution of the previous one as an initial guess. However, finding a suitable sequence (λ_k) is often problematic in practice, as the question of setting the step in λ is left to the experimenter, and often implies a lot of trial-and-error tests. This is why one has in many cases to resort to a full path following method.

Predictor Corrector and Piecewise Linear methods

Contrary to the discrete continuation approach described above, full path following methods try to follow the zero path entirely, and can be classified into two broad families, the Predictor Corrector (PC) methods and the Piecewise Linear (PL) continuation methods. Extensive documentation concerning both classes of methods can be found in the reference works by E.Allgower and K.Georg, see [1, 2] in particular.

Predictor Corrector (or "differential") continuation methods follow the zero path as a differentiable curve. They are generally fast, as the step-size along the homotopic parameter λ is computed automatically, and the path following can therefore take a very small number of steps to converge sometimes. In return, differential continuation requires some smoothness properties of the path (typically C^2-differentiability and a Jacobian of maximal rank). Also, the computation of the Jacobian of the homotopy can raise some practical difficulties, especially in our case where H is a shooting function that requires an IVP integration. An efficient PC method is implemented for instance in the package HOMPACK by Watson ([38]), that was used by T.Haberkorn to solve the family of orbital transfer problems studied in chapter 3, see [26].

Piecewise Linear (or "simplicial") methods, on the other hand, actually follow a piecewise linear approximation of the zero path, and are supposed

to be more robust, though slower, than Predictor Corrector methods. This work is based on the application of these methods to optimal control problems with an expected low regularity, namely with bang-bang and singular arc control structure. We recall in chapter 2 the general principle of these simplicial methods, and describe some of the improvements we have tried to bring in this particular context.

We illustrate now on the previous simple example the continuation method used to obtain a reliable initial point.

Back to the simple example: continuation on the objective

We parametrize here the objective by $\lambda \in [0,1]$:

$$J_\lambda = \int_0^2 \lambda|u| + (1-\lambda)|u|^2 dt$$

This gives a family of problems (P_λ) such that (P_1) is our original problem (P). Applying the PMP to (P_λ) gives the state and costate dynamics (unchanged)

$$\varphi \begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = u(t) \\ \dot{p}_1(t) = 0 \\ \dot{p}_2(t) = -p_1(t) \end{cases}$$

and the expression of the optimal control

$\lambda < 1$

$$\begin{array}{lll} if\ |p_2(t)| \leq \lambda & then & u^*(t) = 0 \\ if\ |p_2(t)| > 2-\lambda & then & u^*(t) = -sgn(p_2(t)) \\ & else & u^*(t) = -sgn(p_2(t))\frac{|p_2(t)|-\lambda}{2(1-\lambda)} \end{array}$$

$\lambda = 1$

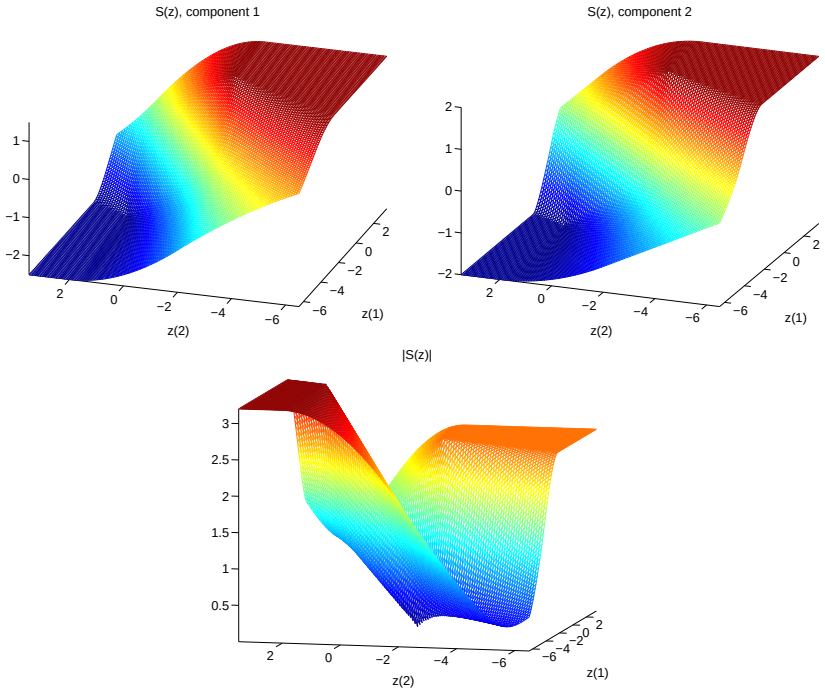
$$\begin{array}{lll} if\ |p_2(t)| < 1 & then & u^*(t) = 0 \\ & else & u^*(t) = -sgn(p_2(t)) \end{array}$$

with the boundary conditions on the costate p:
- $p(0)$ is free (thus $z = p(0)$).
- $p(2)$ is free, as $x(2)$ is fixed.

If we note S_λ the shooting function associated to (P_λ), our aim is now to follow a zero path of the homotopy

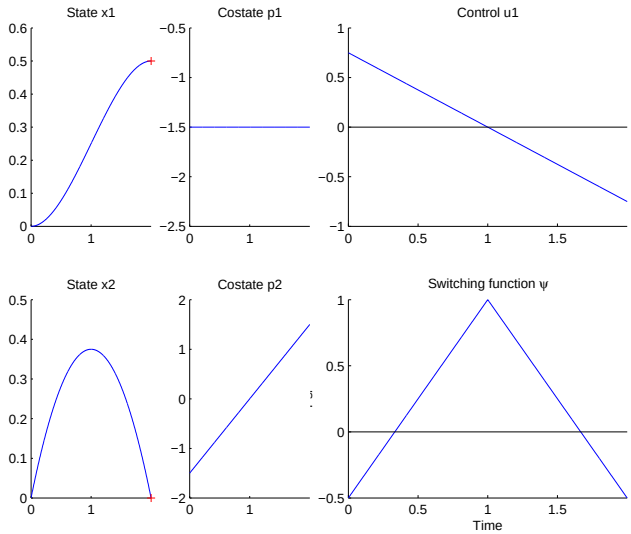
$$H : (z, \lambda) \mapsto S_\lambda(z)$$

from $\lambda = 0$ to $\lambda = 1$. For $\lambda = 0$ the problem (P_0) is much more regular, as we can see that the optimal control is continuous, instead of the discontinuous, bang-bang structure for $\lambda = 1$. Then the corresponding shooting function S_0 is also more regular, as shown on the graphs below.



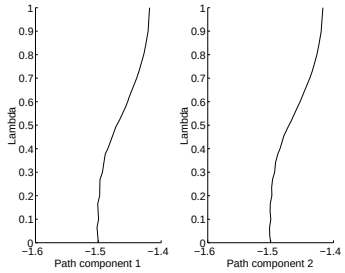
Shooting function for (P_0) *at* $\lambda = 0$

Indeed, for (P_0) the single shooting method converges for any of the 9 initializations tried previously, and gives the following solution:



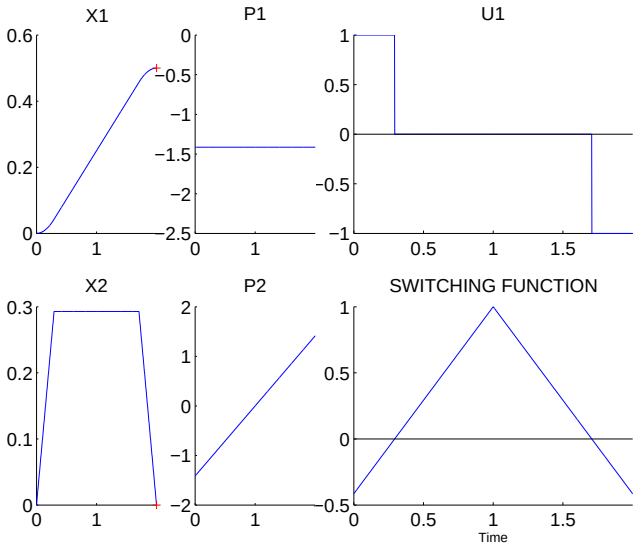
Solution for (P_0) - $z^* = (-1.5, -1.5)$

Now all we have to do is to perform the path following from $\lambda = 0$ to $\lambda = 1$, which is done by the simplicial algorithm detailed in chapter 2. We represent here the two components of the zero path (that happen in this simple example to be the same, which is of course not the case in general).



Zero path for problem (P)

Now, if we take the solution z_1 at the end of the path (for $\lambda = 1$) as initial point, the single shooting converges immediately. Here is the solution with the commutations when the switching function $\psi = 1 - |p_2|$ vanishes.



Solution for problem (P) - $z^* = (-\sqrt{2}, -\sqrt{2})$.

A second simple example: singular arcs

We consider here a second simple example, to illustrate the case of singular arcs, which is studied later in chapter 4. Let us define the problem family

$$(P_\lambda) \begin{cases} Min \int_0^2 \lambda|u(t)| + (1-\lambda)u^2(t)dt \\ \dot{x} = u \\ |u| \leq 1 \ on \ [0,2] \\ x(0) = 0 \\ x(2) = 0.5 \end{cases}$$

The application of the Maximum Principle gives the expression of the optimal control

$$
\begin{array}{lllll}
\lambda < 1 & & & & \\
 & if\ |p(t)| \leq \lambda & then & u^*(t) = 0 & \\
 & if\ |p(t)| > 2 - \lambda & then & u^*(t) = -sgn(p(t)) & \\
 & & else & u^*(t) = -sgn(p(t)) \frac{|p(t)| - \lambda}{2(1-\lambda)} & \\
\lambda = 1 & & & & \\
 & if\ |p(t)| < 1 & then & u^*(t) = 0 & \\
 & else\ if\ |p(t)| > 1 & then & u^*(t) = -sgn(p(t)) & \\
 & else & & u(t) = -\alpha(t)\ sgn(p(t)) & , \alpha(t) \in [0,1].
\end{array}
$$

We notice that when $|p(t)| = 1$, the optimal control is multi valued. If this occurs on a non-trivial interval, we have a *singular arc*.

We have the Boundary Value Problem

$$
(BVP_\lambda) \begin{cases} \dot{x}(t) = u(t) \\ \dot{p}(t) = 0 \\ x(0) = 0 \\ x(2) = 0.5 \\ p(0)\ free \\ p(2)\ free \end{cases}
$$

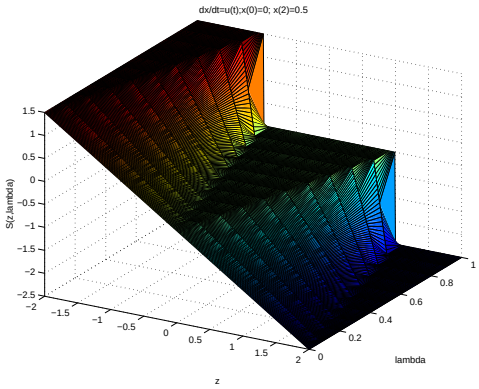
and define the shooting function S:

$$
S_\lambda(z) = \hat{x}(2) - 0.5
$$

where $\hat{x}$ is solution of the Initial Value Problem

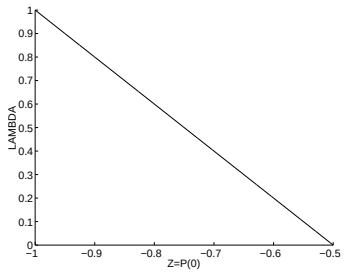
$$
(IVP_\lambda) \begin{cases} \dot{x}(t) = u(t) \\ \dot{p}(t) = 0 \\ x(0) = 0 \\ p(0) = z. \end{cases}
$$

We set the homotopy $H : (z, \lambda) \mapsto S_\lambda(z)$, as usual. For $\lambda = 0$, the solution corresponds to a constant control $u = 0.25$, with $z^* = p^*(0) = -0.5$. For $\lambda = 1$, any non negative control that satisfies $\int_0^2 u(t)dt = 0.5$ is optimal, which corresponds to a singular arc over $[0, 2]$, with $z^* = p^*(0) = -1$. We draw here the shooting function S.



Shooting function for a simple singular arc example

As we can see, S is not defined at the solution for $(z, \lambda) = (-1, 1)$. If we try a differential continuation to follow the path, the algorithm stops just before $\lambda = 1$, which is not surprising. A simplicial method, on the other hand, actually converges to the correct solution z^* at $\lambda = 1$:



Path following for a simple singular arc example

1.2 Some convergence results

1.2.1 Hamiltonian minimization properties

We begin with some results concerning the Hamiltonian minimization, that were presented by J.Gergaud in [24] (see also [29]).

Notations: for the sake of brevity, we often use in the following the notation $y = (x, p)$, with y of dimension n (typically twice the state dimension). We also denote by $\mathcal{H}$ the Hamiltonian, the standard capital H being reserved for the homotopy used in the continuation.

We first recall a standard result:

THEOREM 1 *Assume that $U \subset \mathbf{R}^m$ is a convex compact set whose interior is nonempty, and that the Hamiltonian function $\mathcal{H} : [a, b] \times \mathbf{R}^n \times U \to \mathbf{R}$ is continuous and convex with respect to the control u. We denote $\Gamma(t, x, p)$ the set of solutions of $\min_{u \in U} \mathcal{H}(t, x, p, u)$ Then Γ is a non empty compact convex valued map.*

And the following definitions:

DEFINITION 2 *A set valued map $F : X \to Y$ is upper semi continuous (usc) (in the sense of Berge, [8] page 114) at x_0 iff for all open subset O of Y containing $F(x_0)$, there exists a neighborhood U of x_0 such that $F(U) \subset O$. F is upper semi continuous iff F is usc at every point in X.*

DEFINITION 3 *A set valued map $F : X \to Y$ is lower semi continuous (lsc) at x_0 iff for all $y_0 \in F(x_0)$ and all sequence (x_k) that converges to x_0, there exists a sequence (y_k), $y_k \in F(x_k)$ that converges to y_0. F is lower semi continuous iff F is lsc at every point in X.*

DEFINITION 4 *We call proper function any function with real values that never takes the value $-\infty$ and is not identically equal to $+\infty$.*

DEFINITION 5 *A function $f : \mathbf{R}^m \to \overline{\mathbf{R}}$ is inf-compact (in the sense of [33]) iff*

$$\forall (y, a) \in \mathbf{R}^m \times \mathbf{R}, \quad \{u \in \mathbf{R}^m : f(u) - (u|y) \leq a\} \quad \textit{is compact.}$$

($\overline{\mathbf{R}}$ is the extended real line and $(\cdot|\cdot)$ stands for the usual inner product in $\mathbf{R}^m$.)

LEMMA 1 *A compact valued map G is usc if and only if for all sequence (x_k) that converges to x, $(G(x_k))$ converges to $G(x)$ according to $\forall \epsilon > 0,\ \exists k_0 > 0$ such that $\forall k > k_0,\ G(x_k) \subset G(x) + \epsilon\ B(0, 1)$.*[2]

[2] $B(0, 1)$ stands for the closed unit ball of center 0 and radius 1.

Proof. see [30] page 66.

LEMMA 2 *Let $(f_k)_{k\in\mathbf{N}}$ be a sequence of proper convex lower-semicontinuous functions defined over $\mathbf{R}^m$. Under the following assumptions*
(i) $(f_k)_{k\in\mathbf{N}}$ converges point-wise to f
(ii) $int(domf) \neq \emptyset$
(iii) f is inf-compact
Then,

$$\lim_{k\to\infty} \inf_{u\in\mathbf{R}^m} f_k(u) = \inf_{u\in\mathbf{R}^m} f(u)$$

and, $\forall \epsilon > 0$, there is $k_0 \in \mathbf{N}$ such that

$$argmin_{u\in\mathbf{R}^m}\ f_k(u) \subset argmin_{u\in\mathbf{R}^m}\ f(u) + \epsilon\ B(0,1) \quad \forall k \geq k_0.$$

Proof. see [33], page I.3.54.

THEOREM 2 *Consider the same hypotheses as in Theorem 1.*
Then Γ has the following convergence property:
If (t_k, x_k, p_k) is a sequence that converges to (t, x, p) then
(i) $inf_{u\in\mathbf{R}^m}\ \mathcal{H}(t_k, x_k, p_k, u) \to \inf_{u\in\mathbf{R}^m}\ \mathcal{H}(t, x, p, u)$ when $k \to +\infty$
(ii) $\forall \epsilon > 0,\ \exists k_0 > 0$ such that $\forall k > k_0,\ \Gamma(t_k, x_k, p_k) \subset \Gamma(t, x, p) + \epsilon\ B(0, 1)$.

Proof. Let (t_k, x_k, p_k) be a sequence that converges to (t, x, p).
We note $f_k(u) = \mathcal{H}(t_k, x_k, p_k, u) + \delta(u/U)$ and $f(u) = \mathcal{H}(t, x, p, u) + \delta(u/U)$ (where $\delta(u/U) = 0\ if\ u \in U,\ and + \infty\ if\ u \notin U$).
For the problems studied in the following, it is clear from the expression of the Hamiltonian (see below) that the (f_k) are convex and lower-semicontinuous.
Let us verify the assumptions of Lemma 2:
(i) if $u \notin U$ then $f_k(u) = f(u) = +\infty\ \forall k$
if $u \in U$ then $f_k(u) = \mathcal{H}(t_k, x_k, p_k, u)$, and as $\mathcal{H}$ is continuous we have $\mathcal{H}(t_k, x_k, p_k, u) \to \mathcal{H}(t, x, p, u)$, so $f_k(u) \to f(u)$.
(ii) $int(domf) = int(U) \neq \emptyset$.
(iii) Let be $v \in \mathbf{R}^m, a \in \mathbf{R}$, then $\{u\ |\mathcal{H}(t, x, p, u) + \delta(u/U) - (u|v) \leq a\} = U \cap \{u\ |\mathcal{H}(t, x, p, u) - (u|v) \leq a\}$. This is a closed subset of $\mathbf{R}^m$ included in U compact, which is therefore compact. This gives the inf-compactness of f.
Now Lemma 2 proves the theorem.

COROLLARY 1 *Consider the same hypotheses as in Theorem 1.*
Then Γ is upper-semicontinuous.

Proof. Theorem 1, Lemma 1 and Theorem 2 give this result.

Remark: In the case where $\mathcal{H}$ is strictly convex, we then have the well-known property (see for instance [23](Theorem 6.1 p.75) and [10]) that u^* is a continuous function (as Γ is then a continuous function).

1.2.2 Convergence properties

The following results were presented by J.Gergaud in [15], and are primarily derived from the books [5] by J.P. Aubin and A. Cellina, and [22] by A.F. Filippov, whose notations we will keep.

Notations:
- *let* $K \subset \mathbf{R}^n$, $\overline{co}\ K$ *denotes the closed convex hull of* K
- *let* $M \subset \mathbf{R}^n$, $M^\delta = \{m \text{ such that } d(m, M) \leq \delta\}$

We recall here the definition of a δ-solution as in [22] (page 76).

DEFINITION 6 *A function* y *is called a* δ*-solution of* $\dot{y}(t) \in F(t, y(t))$*, with* $F : [a, b] \times \mathbf{R}^n \to \mathbf{R}^n$ *an upper-semicontinuous set valued map, if over an interval* $[a, b]$ y *is absolutely continuous and*

$$\dot{y}(t) \in F_\delta(t, y(t)) = [\overline{co}\ F(t^\delta, y^\delta)]^\delta$$

where

$$F(t^\delta, y^\delta) = \cup_{s \in t^\delta, z \in y^\delta}\ F(s, z).$$

LEMMA 3 *Let* (y_k) *be a sequence of* $AC_n([a, b])$ *(AC stands for absolutely continuous) such that:*
(i) $\forall t \in [a, b]$, $\{y_k(t)\}_k$ *is relatively compact.*
(ii) $\exists l$ *such that* $|\dot{y}_k(t)| \leq l$ *almost everywhere in* $[a, b]$.
Then there exists a subsequence still noted (y_k) *that converges uniformly to an absolutely continuous function* $y : [a, b] \to \mathbf{R}^n$*, and for which the sequence* $(\dot{y}_k)$ *converges weakly-* to* $\dot{y}$ *in* $L^\infty_n([a, b])$.

Proof. The proof follows the principle of the demonstration of Theorem 4, page 14-15 in [5]. The sequence (y_k) is equicontinuous as

$$|y_k(t') - y_k(t'')| = \int_{t'}^{t''} \dot{y}_k(t)\ dt\ \leq l|t' - t''|.$$

The Arzelá-Ascoli theorem implies the existence of a subsequence, still noted (y_k), that converges uniformly to y in $C_n([a, b])$. Moreover, $\dot{y}_k \in \overline{B}(0, l) \subset L^\infty_n([a, b])$, and $L^\infty_n([a, b])$ is the dual of $L^1_n([a, b])$. Thus the Alaoglu theorem implies that this closed ball is weak-* compact. As $L^1_n([a, b])$ is separable this closed ball is metrizable for the weak-* topology (cf [11]). Then there exists a subsequence, still noted $(\dot{y}_k)$, that converges weakly-* to z in $L^\infty_n([a, b])$.

We now have to prove that y is absolutely continuous and that $\dot{y} = z$. First, (y_k) is absolutely continuous, thus

$$y_k(t') - y_k(t'') = \int_{t'}^{t''} \dot{y}_k(s)\ ds \qquad (*)$$

(y_k) converges (uniformly) to y, so the left hand side converges to $y(t') - y(t'')$. As $(\dot{y}_k)$ converges weakly-* to z, for all components i we have

$$< \mathbf{1}, \dot{y}_{k,i} >_{L^1,L^\infty} = \int_a^b \dot{y}_{k,i}(s)ds \rightarrow < \mathbf{1}, z_i >_{L^1,L^\infty} = \int_a^b z_i(s)ds$$

(where $\mathbf{1}$ is the constant map equal to 1). So the right hand side of (*) converges to $\int_{t'}^{t''} z(s)ds$. We have then

$$y(t') - y(t'') = \int_{t'}^{t''} z(s)ds$$

with z in $L^\infty_n([a,b])$, thus in $L^1_n([a,b])$. This means that y is absolutely continuous and that $\dot{y}(t) = z(t)$ almost everywhere.

THEOREM 3 *Let (y_k) be a sequence of $AC_n([a,b])$ that converges to y and satisfies $\dot{y}_k(t) \in K$ for all k and t, with K compact. Then y is absolutely continuous and $\dot{y}_k(t) \in \overline{co}\ K$ for all t.*

Proof. The proof is based on Filippov's Lemma 13, page 64 of [22].

THEOREM 4 *Let F be a non empty compact convex valued map, defined on an open set $\Omega \subset \mathbf{R}^{n+1}$. Let (y_k) be a sequence of δ_k-solutions defined on $[a,b]$ that converges uniformly to $y : [a,b] \rightarrow \mathbf{R}^n$ when $\delta_k \rightarrow 0$, and such that the graph of y is in Ω. Then y is a solution of the differential inclusion $\dot{y}(t) \in F(t, y(t))$.*

Proof. see Filippov's Lemma 1, page 76 of [22]

LEMMA 4 *Let $\varphi : \Omega \times [0,1] \rightarrow \mathbf{R}^n$, with Ω an open subset of $\mathbf{R}^n$, be a set valued map verifying*
(i) φ is upper-semicontinuous and non empty compact convex valued.
(ii) $\varphi_\lambda = \varphi(\cdot, \lambda)$ is a usual function and is piecewise $- C^1$ for $0 \leq \lambda < 1$.
Let us assume that the solutions of $\dot{y}_\lambda(t) = \varphi_\lambda(y(t))$ remain in a fixed compact K and are defined on an interval $[0, t_f]$. Then y_λ is a δ-solution of the differential inclusion $\dot{y}(t) \in \varphi(y(t), 1)$, and δ tends to 0 when $\lambda \rightarrow 1$.

Proof. φ is upper-semicontinuous at $(y^*, 1)$ for all $(y^*, 1) \in K$, thus for all $\epsilon = \delta$, there exists η such that for all $|y - y^*| < \eta_{y^*} < \delta$, $|\lambda - 1| < \eta_{y^*} < \delta$, we have $\varphi_\lambda(y, \lambda) \in \varphi(y^*, 1)^\epsilon$.

Thus $K \subset \cup_{y^* \in K} B(y^*, \eta_{y^*})$ and as K is compact, we have $K \subset \cup_{i=1}^q B(y_i, \eta_i)$.

$\forall \epsilon = \delta, \forall y_i, \exists \eta_i\ |y - y_i| < \eta_i < \delta$ and $|\lambda - 1| < \eta_i < \delta$, $\varphi_\lambda(y) \in \varphi(y_i, 1)^\epsilon$.
For all $y \in K$, there exists y_i such that $y \in B(y_i, \eta_i)$ and thus

$$\varphi_\lambda(y) \ \in \ (\varphi(y_i, 1))^\epsilon \ \subset \ (\varphi(y_i^\eta, 1))^\epsilon \ \subset \ (\varphi(y^\delta, 1))^\epsilon.$$

Then for all λ such that $|\lambda - 1| < \eta$ and for all $y \in K$,
we have $\varphi_\lambda(y) \in (\varphi(y^\delta, 1))^\delta$.

THEOREM 5 *Let us assume that the solutions of $(BVP)_\lambda$ remain in a fixed compact subset of $[0, t_f] \times K$, $K \subset \Omega$, with Ω an open subset of $\mathbf{R}^n$. Then from any sequence (y_{λ_k}) of solutions of $(BVP)_{\lambda_k}$, such that $\lambda_k \to 1$ when $k \to +\infty$, we can extract a subsequence (y_k) verifying:*
(i) (y_k) converges uniformly to y solution of $(BVP)_1$.
(ii) $(\dot{y}_k)$ converges weakly- to $\dot{y}$ in $L^\infty_n([0, t_f])$.*

Proof. $\varphi(y, \lambda)$ is usc[3] thus $\varphi(K, [1 - \epsilon, 1])$ is compact. There exists l such that $|\dot{y}_\lambda(t)| < l$ for $\lambda \in [1 - \epsilon, 1]$. The y_λ are absolutely continuous, and Lemma 3 says that we can extract a subsequence (y_k) that converges uniformly to y, and such that $(\dot{y}_k)$ converges weakly-* to $\dot{y}$ in $L^\infty_n([0, t_f])$. As per Lemma 4, (y_k) is a δ_k-solution. $\varphi(y, 1)$ is non empty compact convex valued, so Theorem 3 says that y is a solution of the differential inclusion $\dot{y}(t) \in \varphi(y(t), 1)$. Initial and terminal conditions can be written as $h_0(y(0)) = 0$ and $h_f(y(t_f)) = 0$ with h_0 and h_f continuous. The uniform convergence of (y_k) implies that y verifies the boundary conditions, thus y is a solution of $(BVP)_1$.

COROLLARY 2 *Under the hypotheses of Theorem 5, assume that $\dot{x} = f(t, x, u)$ provides an expression of the control of the form $u = S(t, x) + R(t, x)\dot{x}$, with R and S continuous and R linear with respect to x. Consider the subsequence $(y_k) = (x_k, p_k)$ from Theorem 5, and denote $(u_k) = S(t, x_k) + R(t, x_k)\dot{x}_k$. Then (u_k) converges weakly-* in $L^\infty_n([0, tf])$.*

Proof. see [17], proof of Proposition 3.2, page 551-552.

It should be noted that Theorem 5 and Corollary 2 only give the convergence of a subsequence from the sequence of solutions (y_{λ_k}), which comes from the Arzelá-Ascoli theorem in Lemma 3. As far as we know, under the chosen assumptions we cannot state the convergence of the sequence of solutions itself.

[3] usc stands for upper-semicontinuous

Chapter 2

PL continuation - Simplicial method

We recall here the general principle of simplicial algorithms, and then detail some algorithmic points we have introduced as part of this work. We would like to emphasize that we have tried to stay coherent with the problems studied, characterized by a low regularity. We want in particular to keep the robustness of the simplicial methods, therefore the modifications introduced should not require additional regularity assumptions on the path (which probably would not be satisfied in practice anyway). This is also why we use at the core a basic version of the simplicial algorithm, and have not considered some existing improvements (such as the mixing of Newton steps for instance).

2.1 Simplicial methods principle

As mentioned earlier, readers interested in Predictor Corrector and Piecewise Linear continuation methods should refer in particular to E.Allgower and K.Georg [1, 2, 3], as well as M.Todd [36, 37] for simplicial methods more specifically, to mention only a few.

Piecewise Linear continuation methods actually follow the zero path of the homotopy H by building a piecewise linear approximation of H, hence their name. This approximation is built over a certain subdivision of the space, most often in a particular way called a *triangulation* in *simplices*. The popular use of this kind of subdivision has led PL continuation methods to be often referred to as "simplicial methods".

The main advantage of these methods is that they put extremely low requirements on the homotopy H. As no derivatives are used, the homotopy is not supposed to be differentiable, unlike with PC methods. Continuity is

in fact sufficient, and should not even be necessary in all cases. Also, this class of methods can be more easily adapted to the case of a multi-valued homotopy, which is interesting as we plan to study singular arcs problems that fall into this category. The main disadvantage of these methods is that they are quite slower than Predictor-Corrector methods, when the latter converge of course.

Simplices and triangulations

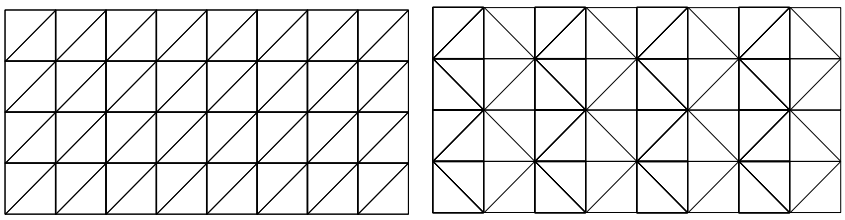
In the following we consider an homotopy $H : \mathbf{R}^{n+1} \to \mathbf{R}^n$. Let us begin with some useful definitions.

DEFINITION 7 (SIMPLICES AND FACES) *A simplex is the convex hull of $n+2$ affinely independent points (called the vertices) in $\mathbf{R}^{n+1}$, while a k-face of a simplex is the convex hull of k vertices of the simplex (note: k is typically omitted for $(n+1)$-faces, which are just called faces).*

DEFINITION 8 (TRIANGULATION) *A triangulation is a countable family T of simplices of $\mathbf{R}^{n+1}$ verifying:*
- *The intersection of two simplices of T is either a face or empty*
- *T is locally finite (a compact subset of $\mathbf{R}^{n+1}$ meets finitely many simplices).*

In practice, a triangulation is described by the form of its simplices (meaning the relations between the vertices), and also by its "pivoting rules", which tell how to build a simplex adjacent to a given one.

The following graphs show two famous triangulations, the K_1 from Freundenthal, and the "Union Jack" J_1 from Todd. These are "uniform" triangulations, meaning that the simplices keep the same size all along the space, unlike some "refining" triangulations shown below.

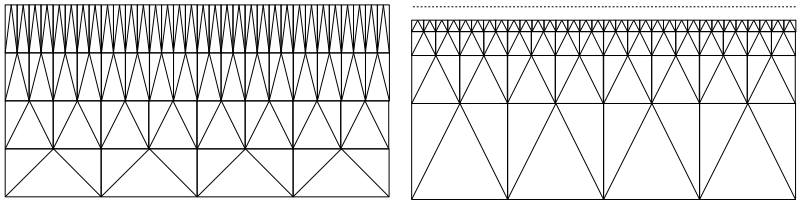


Uniform triangulations K_1 and J_1 of $\mathbf{R} \times [0,1]$

These two triangulations are very easy to manipulate, as the pivoting rules can be expressed as a mere geometric reflexion (see [2] 13.3 for instance). This simplicity is the reason why we have naturally began the numerical experiments with these two triangulations, and we have found their overall behaviour quite robust. Of course, several other, more sophisticated triangulations have been developed, however their use is a bit more complicated.

For instance, we have made some tests the D_1 described by Dang in [21] (which looks same as the J_1 in dimension 2, but you can see the differences on the graphs page 29), that have not been quite convincing.

Then there are also some "refining" triangulations, in which the size of the simplices decreases when getting nearer to the convergence. The idea behind these is to combine the advantages of speed at the beginning of the zero path, where things are supposed to go smoothly, and precision at the convergence, to obtain an accurate solution.



Refining triangulations: J_4 and J_3 (of $\mathbf{R} \times [0, 1[$ for the latter)

Remark: The graph above shows that for the J_3 triangulation the size of the simplices tends to zero as the homotopic parameter tends to the convergence value (for instance 1) but never actually reaches it. A particular stopping criterion is used in this case, namely when the homotopic parameter crosses a certain threshold $1 - \epsilon$, with $\epsilon = 10^{-6}$ or 10^{-9} for instance.

However, we have observed that using these more evolved triangulations can be problematic in practice. We have noticed that they often encounter difficulties either at the beginning (especially for J_3 whose starting simplices are too large to initiate the path following properly), or near the convergence, where the simplices actually become to small. This is probably due to the particular nature of the problems we study: our homotopies are shooting functions, which can be highly nonlinear even at the beginning of the continuation, therefore the path is not that easy to follow. Moreover, as the situation is often problematic at $\lambda = 1$, the use of small simplices tends to make the algorithm to stagger on difficulties when coming near the convergence. Besides, the primary objective of the continuation approach is for us not to solve the problem precisely, but just to provide a good initial point for the shooting method. So the whole point of using small simplices to have a more precise solution is not always relevant in our case anyway.

Labeling and zero path following

In addition to the choice of the triangulation, we need a relation between the homotopy and the vertices of the simplices. This is done by "labeling" the vertices with respect to the homotopy value at these points. We use here the

standard vector labeling described below, but there are other possibilities, like for instance the integer labeling described in Appendix B.2, page 145.

DEFINITION 9 (LABELING) *We call labeling a map l that associates a value to the vertices v_i of a simplex. We label here the simplices by the homotopy H: $l(v^i) = H(z^i, \lambda^i)$, where $v^i = (z^i, \lambda^i)$.*

An affine interpolation on the vertices thus gives a PL approximation of the homotopy H over the triangulation T. We denote H_T this approximation, whose zero path we want to follow. We recall for this the notion of "completely labeled face".

DEFINITION 10 (COMPLETELY LABELED FACE) *A face $[v_1, .., v_{n+1}]$ of a simplex is completely labeled if and only if it contains a solution v_ϵ of the equation $H_T(v) = \vec{\epsilon}$ for all sufficiently small $\epsilon > 0$ (where $\vec{\epsilon} = (\epsilon, .., \epsilon^n)$).*

There exists a more implementable way of characterizing a completely labeled face, given for instance in [2], chapter 12.3, page 159.

DEFINITION 11 (LABELING MATRIX) *The labeling matrix L of a face $[v_1, .., v_{n+1}]$ is defined by*

$$L = \begin{pmatrix} 1 & \dots & 1 \\ H(v_1) & \dots & H(v_{n+1}) \end{pmatrix}.$$

Then we have the following property.

LEMMA 5 *A face $[v_1, .., v_{n+1}]$ is completely labeled if and only if its labeling matrix L is nonsingular and L^{-1} is lexicographically positive, ie the first non-vanishing entry in any row of L^{-1} is positive.*

Proof. see [2], proposition 12.3.3, page 159-160

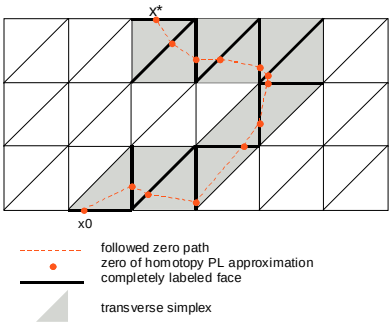
Now we recall the fundamental property on which is based the core of a simplicial algorithm.

LEMMA 6 *Each simplex possesses either zero or exactly two completely labeled faces (being called a* transverse *simplex in the latter case).*

Proof. see [2], chapter 12.4, page 163-165.

The constructive proof of this property, which gives the other completely labeled face of a simplex that already has a known one, is often referred to as *PL step, linear programming step*, or *lexicographic test*. Once again, we refer the reader to [2] (pp 163-165), or [24] (pp 69-70), for a detailed description of these mechanisms. Then per the triangulation properties, there exists a unique transverse simplex that shares this second completely labeled face, that can be determined via the pivoting rules specific to the triangulation.

A simplicial algorithm thus basically follows a sequence of transverse simplices, from a given first transverse simplex with a completely labeled face at $\lambda = 0$, to a final simplex with a completely labeled face at $\lambda = 1$ (or $1 - \epsilon$ for some refining triangulations that never reach 1, such as J_3), which contains an approximate solution of $H(z, 1) = 0$.

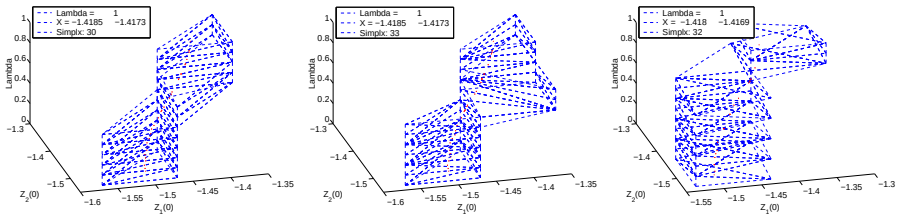


Schematic Zero Path following for K_1 triangulation in dimension 2

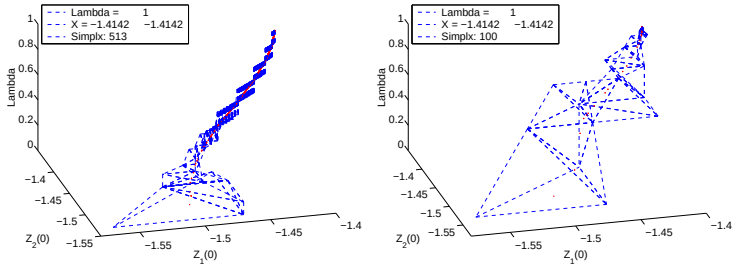
Sample path followings

We illustrate here the path following for the simple problem (P) described in chapter 1, for various triangulations (whose complete description can be found for instance in [2, 37, 21]).

Triangulation	Simplices	Solution	Final norm
$K_1(10^{-1})$	30	(-1.418,-1.417)	2.13 10^{-3}
$J_1(10^{-1})$	33	(-1.418,-1.417)	2.13 10^{-3}
$D_1(10^{-1})$	32	(-1.418,-1.417)	1.94 10^{-3}
$J_3(10^{-1})$	68	(-1.414,-1.414)	7.17 10^{-12}
$J_4(10^{-1})$	513	(-1.414,-1.414)	1.31 10^{-9}



Path following form $\lambda = 0$ to $\lambda = 1$ - K_1, J_1 and D_1 triangulations



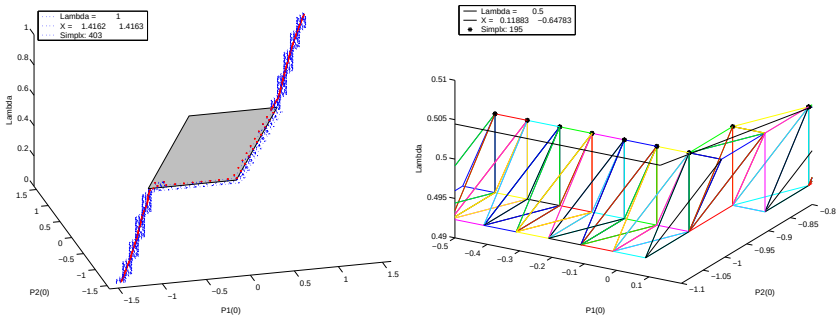
Path following form $\lambda = 0$ to $\lambda = 1$ - J_4 and J_3 triangulations

Following a path of non maximal rank

On a side note, we can illustrate on this simple example the robustness of the simplicial methods, and more specifically the ability to follow a path that is not of maximal rank (contrary to the differential continuation for instance). We consider here another continuation for the problem (P), in which we modify the terminal conditions according to λ, leading to the family of problems

$$(P_\lambda)\begin{cases} Min \ \int_0^2 |u|dt \\ \dot{x}_1 = x_2 \\ \dot{x}_2 = u \\ |u| \leq 1 \\ x(0) = (0,0) \\ x(2) = (0.5-\lambda, 0) \end{cases}$$

Here we have the original problem (P) for $\lambda = 0$, while $\lambda = 1$ corresponds to the "mirror" problem, with the opposite optimal control. We notice that for $\lambda = 0.5$, the terminal and initial conditions coincide, with $x(0) = x(2) = 0$. So in this case, the null control is the optimal solution, which corresponds to any value of $p(0)$ within the central area on the graph on page 11. Therefore the zero path is not of maximal rank at $\lambda = 0.5$. Nevertheless, the simplicial algorithm still converges to the right solution, as illustrated below.



Central area crossing at $\lambda = 0.5$

We observe that the simplicial algorithm manages to cross the central area at $\lambda = 0.5$ by actually following its frontier, with simplices having vertices on both sides. Incidentally, depending on the triangulation and/or meshsize used, the algorithm can follow the other frontier instead of the "lower-right" one on the graph above.

Simplices and labeling matrix stability

As mentioned before, at each step of a simplicial algorithm the current transverse simplex is updated. More precisely, it is built from the previous one with which it shares a completely labeled face, which means that there is

only one vertex to modify. This is done by the pivot rules after the lexicographic test has determined which vertex of the previous simplex is to be replaced (ie is not part of the exit completely labeled face). At the same time, the inverse of the labeling matrix is updated directly, instead of re-inverting the updated labeling matrix.

Then one may be worried about the accumulation of numerical errors when the number of followed simplices grows large. To check this, we periodically recompute all the vertices from the formal expression of the simplex, and the corresponding labeling matrix inverse.

The following tables show the evolution of the cumulative error on the simplex vertices and labeling matrix inverse. The tests are made with the simple example problem from chapter 1 and the two singular arcs problems studied in chapter 4 (discretized BVP formulation). In each case we recall the triangulation used, the dimension of z, and the number of simplices followed.

Demo Problem: $K_1(10^{-6})$*, dimension 2,* $> 3\ 000\ 000$ *simplices*

Check frequency	Simplex error (abs/rel)	Labeling error (abs/rel)
10 000	0 / 0	$10^{-8} - 10^{-12}$ / $10^{-15} - 10^{-17}$
100 000	0 / 0	$10^{-9} - 10^{-11}$ / $10^{-16} - 10^{-18}$
1 000 000	0 / 0	10^{-10} / 10^{-16}

Problem 1: $K_1(10^{-2})$*, dimension 101,* $> 200\ 000$ *simplices*

Check frequency	Simplex error (abs/rel)	Labeling error (abs/rel)
1 000	$10^{-15} - 0/10^{-17} - 0$	$10^{-9} - 10^{-11}/10^{-13} - 10^{-15}$
10 000	$10^{-15} - 0/10^{-17} - 0$	$10^{-9} - 10^{-11}/10^{-13} - 10^{-14}$
100 000	0 / 0	$10^{-10} - 10^{-11}$ / 10^{-14}

Problem 2: $K_1(10^{-2})$*, dimension 82,* $> 100\ 000$ *simplices*

Check frequency	Simplex error (abs/rel)	Labeling error (abs/rel)
1 000	$10^{-15} - 0$ / $10^{-17} - 0$	$10^{-9} - 10^{-11}$ / $10^{-13} - 10^{-15}$
10 000	$10^{-17} - 0$ / $10^{-19} - 0$	$10^{-9} - 10^{-11}$ / $10^{-13} - 10^{-15}$
100 000	0 / 0	10^{-10} / 10^{-14}

We observe that the simplex error is numerically negligible, sometimes even undetectable in double precision. As for the labeling error, it can be considered quite reasonable, especially for the two problems with a high dimension (we recall that the computation of this matrix normally involves one homotopy call per vertex, in addition to the matrix inversion). So it seems that we do not have to worry about the accumulation of these updating errors.

Convergence property

For a multi-valued homotopy H, we have the following convergence property.

THEOREM 6 *We consider a PL continuation algorithm using a selection of H for labeling and a refining triangulation of $\mathbf{R}^n \times [0,1[$ (such as J_3 for instance). We make the two assumptions regarding the path following:*
(i) all the faces generated by the algorithm remain in $K \times [0,1]$, with K compact.
(ii) the algorithm does not go back to $\lambda = 0$.
Then if H is usc and compact convex valued, the algorithm generates a sequence (z_i, λ_i) such that $\lambda_i \to 1$, and there exists a subsequence still noted (z_i, λ_i) that converges to $(z, 1)$ such that $0 \in H(z, 1)$.

Proof. the proof comes from [1], chapter 4, page 56.

Specific algorithmic points

We describe in the following some improvements we have tried to come up with during our experiments with the simplicial method (more details about the Simplicial package can be found in appendix A). In accordance with the context of this work, we tried to keep the two following points in mind. First, we assume that the essential part of the overall computational cost comes from the homotopy evaluations, while the purely "simplicial" operations are negligible. This assumption is well verified in practice due to the particular nature of our homotopies (shooting functions). Besides, as we are supposed to study problems with a low regularity in order to benefit from the simplicial method's robustness, we want these modifications to preserve these low requirements as much as possible.

2.2 Junction homotopy

In a simplicial algorithm, the initialization of the path following consists in finding a first completely labeled face on the boundary (typically at $\lambda = 0$). With this face we can build the first transverse simplex of the path, and then perform a sequence of lexicographic tests and pivoting steps, as described before. The point is that obtaining such a face is not always a trivial matter.

As this face is supposed to contain a zero of the PL approximation of the homotopy, a natural idea is to find a zero of $H(\cdot, 0)$, and then try to build a face centered on this point. The first step, finding a zero of the homotopy at the beginning of the continuation, is supposed to be easy: it is part of the principle of the continuation approach.

However, the second step, building a face centered on this solution, does not guarantee that this face is completely labeled. Indeed, the face must contain a zero of the *PL approximation of H over the face*, which is not the

same as a zero of H itself. In practice, depending on the regularity of the homotopy around the starting zero, a sufficiently small face can turn out to be completely labeled (larger faces usually mean a less accurate approximation of H, and are therefore less likely to be completely labeled). An annoying point, however, is that using a small triangulation leads to a huge number of simplices followed, hence an unreasonable overall computational cost. More importantly, this does not always work, sometimes even ridiculously small meshsizes will not give a completely labeled face. So what we would like is a method to obtain a completely labeled face for a given triangulation size. For the path initialization we seek a face at $\lambda = 0$, but we will see that more generally, finding a completely labeled face at a certain level λ_j can be useful too.

2.2.1 Principle

The principle of junction homotopies follows the idea (see for instance [2], Lemma 13.2.6 page 183) that if the labeling is affine, then it is much easier to find a corresponding completely labeled face. More precisely, we consider an intermediate homotopy H_j that takes place at a fixed λ_j with respect to the main path following, and uses its own parameter $\lambda' \in [0, 1]$. We try to connect an application $z \mapsto A\ (z - z_j)$ to $H(\cdot, \lambda_j)$, with the matrix A such as to provide a reasonable affine approximation of $H(\cdot, \lambda_j)$ near the junction starting point z_j. In the initialization case, for $\lambda_j = 0$, this starting point is typically the solution z_0 obtained by a nonlinear solver applied to $H(\cdot, 0) = 0$. The difference with the main homotopy is that building a first completely labeled face for the affine labeling is much easier, as this labeling is identical to its PL approximation on the face. Then if A is nonsingular, the face centered on z_j is completely labeled. The triangulation size for this junction homotopy is the meshsize δ for the wanted completely labeled face, except for the stepsize with respect to the homotopic parameter λ', which we will discuss below.

This intermediate homotopy is supposed to be quick, requiring only a small number of simplices to complete, provided we have an acceptable matrix A. In this regard, we try to further minimize the number of evaluations of H and use a "steep" meshsize, with a stepsize of 1 with respect to λ'. Therefore, junction vertices either verify $\lambda' = 0$ or $\lambda' = 1$ (which always correspond to λ_j on the main path), and we set the labeling for H_j:

$$\begin{cases} H_j(z, \lambda') = A\ (z - z_j) & if\ \lambda' = 0 \\ H_j(z, \lambda') = H(z, \lambda_j) & if\ \lambda' = 1. \end{cases}$$

Now only the vertices on the $\lambda' = 1$ level require an evaluation of H, while the others at $\lambda' = 0$ only cost some matrix/vectors operations.

It is of course possible to set a smaller stepsize along λ', and use a labeling of the form:

$$H_j(z, \lambda') = (1 - \lambda')\ A\ (z - z_j)\ +\ \lambda' H(z, \lambda_j)$$

However, this requires more evaluations of H, which we want to avoid since in our problems, most of the computational cost comes from these calls (H is typically a shooting function and involves the integration of an IVP). Similarly, keeping the first labeling with a stepsize lower than 1 for λ' has been abandoned for the same reasons. In fact, since they actually lead to longer junctions, the only justification behind these less steep labelings is to allow more difficult junctions that would fail otherwise. This difficulty has been solved the other way around, by putting some restrictions on the use of junctions in the case of meshsize changes, see below.

2.2.2 Applications

As said before, junction homotopies have been first introduced for the path following initialization, namely to find the first completely labeled face.

Then another possibility offered by junctions is to change the meshsize of the triangulation during the path following. The idea of changing the size of the simplices along the zero path in order to improve the path following is not new. It is indeed illustrated by the family of refining triangulations, which typically use large simplices at the beginning of the path for a fast progression, then increasingly smaller simplices near the convergence for a good accuracy at the solution. In this particular case, the variable size of the simplices is a structural property of the triangulation, which often leads to more complex pivoting rules, as the "levels" shifts must be taken into account. Under favorable circumstances these refining triangulations can give impressive results. However, their use for the optimal control problem we consider has been rather disappointing, which is why we try another way. Basically, what we choose to do is to interrupt the path following at some points, and try to compute a "better" meshsize (this step is detailed below in 2.3, page 37). Then we perform a junction homotopy to obtain a first completely labeled face for this new triangulation, and resume the path following from there.

To finish with, a third application of junctions can be the solution refining, where we try to find a better solution than the one the path following has converged to. One way to do this is to try a sequence of junctions at the end of the path, with decreasing triangulation sizes (see 2.4.2 page 53 for more details).

Remark: it is worth noting that junction homotopies have been implemented in such a way that the triangulation used for the junction may be of a different kind than the triangulation for the main path following. The only restriction is that a junction must use a uniform triangulation, as its objective is precisely to build a completely labeled face for a given meshsize. For instance, if a uniform triangulation (ie K_1, J_1 or D_1) is selected for the main path, then junctions will use the same triangulation, to facilitate the transitions between main and junction homotopies. Conversely, the first initialization junction for a path with a refining triangulation (J_3 or J_4) has to use a uniform triangulation to find the starting face. This difference then implies some conversion of the obtained face from the junction triangulation format to the main one, which fortunately can be rather easily done.

2.2.3 Practical formulations

We recall that we seek for the junction a practical affine approximation $z \mapsto A(z - z_j)$ of $H(\cdot, \lambda_j)$. The first idea that we have implemented is merely to take for A an approximation of the Jacobian of $H(\cdot, \lambda_j)$ by centered finite differences.

Remarks: Another possibility to obtain directly the Jacobian of H would be to use the variational equation of the shooting problem, that we recall briefly. We consider the initial value problem

$$(IVP) \begin{cases} \dot{y} = \phi(y) \\ y(0) = \beta \end{cases}$$

We assume that ϕ is continuous, and continuously differentiable with respect to y. If $\hat{y}(\cdot, \beta)$ is a solution of (IVP), then $\frac{d\hat{y}}{d\beta}$ is a solution of

$$(Var.\ eq.) \begin{cases} \dot{Y} = \frac{\partial \phi}{\partial y}(y)\ Y \\ Y(0) = Id \end{cases}$$

This property can be used to obtain the derivatives of the shooting function, but the regularity requirements are too high in our context. Then, there are also more sophisticated ways of computing these derivatives, in particular the internal numerical differentiation (IND) presented by Bock in [9]. However, due to the expected low regularity of the problems studied, our goal is actually to find a formulation that does not require derivatives at all. This is why we have not really focused our efforts in a clever computation of these derivatives, and just used basic finite differences until we found a better way.

The main disadvantage of this first formulation is that the step for the finite differences has to be chosen carefully. In our case, the homotopy is a shooting function, whose evaluation requires the integration of an Initial

Value Problem. This integration is done with a certain error, depending on the integrator used, and on the number of steps (for fixed step integrators) or the tolerances (for variable step integrators). If the step is set too large, the approximation of the Jacobian can be so poor that it is unusable. On the other hand, a step too small compared to the precision of the IVP integration can produce some "numerical noise", by differentiating the error.

Moreover, there is the question of the sensitiveness of the homotopy with respect to some components of the unknown. In relation with the scaling of the unknown z, this can require the use of different stepsizes $h_i, i = 1, \ldots, n$ rather than an uniform step h for all dimensions. Similarly, the question of the current meshsize is also of interest, as we would like the approximation to be valid in the neighbourhood of the starting face of the junction (junctions are supposed to just find a completely labeled face near their starting point, and therefore to cover only a short path). We have then modified the previous idea to reflect this link between the finite differences stepsize and the triangulation meshsize. We set now the stepsize by dividing the meshsize by a certain coefficient k, namely

$$h_i = \frac{\delta_i}{k} \quad , \forall i \in [1..n].$$

Numerical experiments suggest that values between 10 and 1000 for k usually give the better results. This variant is a step ahead the first one, but still suffers from the sensitiveness to the stepsize somehow (that is, to the choice of k), which seems to be an unavoidable consequence of using finite differences. Anyway, as said before we expect to encounter shooting functions that may not be differentiable at the solution.

So we would like to find another way, that would not rely on derivatives at all. A possible idea is to use directly the piecewise linear approximation of $H(\cdot, \lambda_j)$ by interpolation on the vertices of the junction starting face. First, there is a slight modification from the previous formulations of the form $z \mapsto A(z - z_j)$, as we prefer here to work with barycentric coordinates of z and z_j with respect to the starting face $[v_1, \ldots, v_{n+1}]$. So we set

$$F = \begin{pmatrix} 1 & \ldots & 1 \\ & & \\ v_1 & \ldots & v_{n+1} \end{pmatrix},$$

and the barycentric coordinates of z and z_j (noted $\hat{z}$ and $\hat{z_j}$) are given by

$$\hat{z} = F^{-1} \begin{pmatrix} 1 \\ z \end{pmatrix} \quad and \quad \hat{z}_j = F^{-1} \begin{pmatrix} 1 \\ z_j \end{pmatrix}.$$

Now we denote the labeling matrix of the face $[v_1, .., v_{n+1}]$ by

$$L = \begin{pmatrix} H(v_1, \lambda_j) & \dots & H(v_{n+1}, \lambda_j) \end{pmatrix}.$$

and set the affine labeling for the junction as

$$z \mapsto L\ (\hat{z} - \hat{z}_j).$$

This expression vanishes for $z = z_j$, and coincides with $H(\cdot, \lambda_j)$ on the vertices v_i, $\forall i \in [1..n+1]$, and we can use it for the junction. The main improvement of this third formulation is to get rid of the derivatives, as we expect a low regularity of the shooting functions. An additional (however minor) benefit is that the junction initialization then only requires n homotopy calls, instead of $2n$ for the centered finite differences.

Summary:
• *First formulation: $A \approx Jac(H(\cdot, \lambda_j))$ via finite differences, with a stepsize h either uniform or proportional to the meshsize δ.*
• *Second formulation: uses the PL approximation of $H(\cdot, \lambda_j)$ on the junction starting face, requires no derivatives.*

A numerical comparison of these formulations on the orbital transfer family of problems can be found in 3.4.2, page 80. Overall, the experiments indicate that the second formulation is indeed more stable than the first one. It is therefore the default mode in the Simplicial code for handling junction homotopies.

2.3 Adaptive triangulation

2.3.1 Principle

One of the generally admitted reasons that makes simplicial algorithm slower than differential continuation methods is the difficulty of adapting the triangulation meshsize to the followed path. Unlike PC methods, where the search direction and steplength are computed automatically to follow the path at best, PL methods intrinsically lack such mechanism, due to the fixed size of the simplices.

Indeed, refining triangulations are a nice way to introduce a variable simplex size, yet this size is still not related to the path actually followed, and is therefore not necessarily well adapted. Moreover, we have found that our problems are not a favorable case for this kind of triangulations. Briefly, the homotopy sensitiveness forbids the use of large simplices at the beginning

of the path, while small simplices can actually make the algorithm more vulnerable to numerical difficulties when the path regularity is low near the end. Except from the case of very simple demonstration problems, most of the time we have not been able to use the refining triangulations (J_3, J_4) in a satisfying way.

Moreover, even with the basic K_1, which has eventually been chosen for most experimentations, as it has shown the best reliability, the choice of the meshsize has turned out to be a non trivial matter. Too large, and the approximation is too far from the zero path to complete the path following. Too small, and the path following takes ages. And of course, it is not quite efficient to try several choices of meshsizes and perform several path following just to find an appropriate setting for a particular problem.

So we would like to try to adapt the triangulation to the followed path dynamically, in order to improve the precision and/or speed of the continuation. For practical reasons, we choose to perform this adaptation only when the simplicial algorithm reaches a certain "level" $\lambda = \lambda_i$, meaning that we have a completely labeled face whose vertices all belong to $\mathbf{R}^n \times \lambda_i$. At this point we compute the new meshsize (see 2.3.2), based on the current triangulation size and the path followed since the previous level $\lambda = \lambda_{i-1}$, as detailed below. Then we proceed to a junction homotopy (see 2.2) to obtain a completely labeled face with the new meshsize. Now the path following can continue from this face, with the new triangulation, until we reach the next level λ_{i+1}. We used to perform these adaptive junctions at new levels in both directions, meaning when we go "back" to the previous level $\lambda = \lambda_{i-1}$ as well. However, in all the problems studied it happens that the paths are monotonous, with an increasing λ. In this case, going down one level merely indicates some localized straying from the path, and should be simply ignored, in order not to further disturb the following. In most cases, this "turning back" lasts only for a few simplices anyway, after which the path following goes on along an increasing λ.

Remarks: this procedure, although rather simple in its principle, is in practice a bit tricky and requires a careful implementation to handle some particular cases correctly. This is actually the most complicated part of the Simplicial code, and a complete line-by-line description would be quite tedious for the reader (and probably not that interesting anyway, since most of it is just plain coding...). However, we would like to briefly mention here some points of interest.

• First, it is sometimes possible to avoid the complete junction homotopy, and just start over the path following with the new meshsize. This is the case when the starting face for the junction happens to be completely

labeled already for the main homotopy $H(\cdot, \lambda_i)$ (we recall that the sole purpose of the junction is precisely to provide such a face). So we test for this possibility just before the beginning of the junction path following, so we can skip this step if the test is positive. This situation does occur in practice, even if it does not seem to be very frequent. As the detection is a mere lexicographically positive check, whose cost is the same as a junction initialization, and much lower than a full junction, I think it is better to test for it.

Another case is when the computation of the new meshsize actually leads to keeping the same triangulation until the next level. Obviously there is no need for a junction here, as the meshsize does not change and the path following just continues normally. However, this has to be detected separately, as it does not fall into the previous case: of course we already know a completely labeled face, but the starting face for the junction is built with (z_i^*, λ_i) (the current zero of the PL approximation H_T of H) as its isobarycenter. So the vertices of this face do not coincide with the previous ones, even if the meshsize is the same (well, theoretically they *could*, if z_i^* were *exactly* the isobarycenter of the previous face...). And with a bit of bad luck, it is quite possible that the starting face be not completely labeled with respect to $H(\cdot, \lambda_i)$, as this property depends on the PL approximation of H over the face. So this case has to be treated separately, in order to avoid the risk of a complete (and technically useless) junction and meshsize "change". This saving is not negligible, as this case happens regularly enough in practice, due to the kind of inertia incorporated in the new meshsize computation (in order to limit inopportune mesh changes, see below).

Note: we just recall that in case of a meshsize increase, the new face has no particular reasons to be completely labeled, even if it contains a completely labeled face. So unlike previously, we cannot skip the junction directly (it is still possible to fall in the first case pointed before, though).

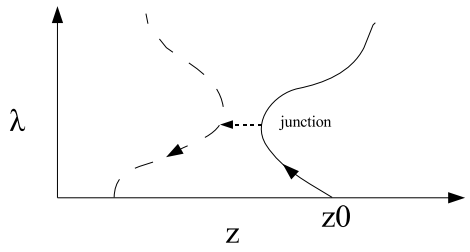
• Then, another point is that unfortunately, a junction is not guaranteed to converge, and more importantly to converge in a reasonable time. Keeping in mind that one of the objectives of the adaptive triangulation is to improve the overall path following performance, it seems obvious that the cost of the junctions has to be controlled in some way. Otherwise, we run the risk of spending more time completing the junctions than the main path gain provided by the meshsize adaptation. This concern is actually quite legitimate, as shown for instance in the experiments with the orbital transfer problems in 3.4 on page 72).

I did not find any miracle solution for this, the best I have come up with is to cap the number of simplices allowed for junctions at a relatively low value, granted that junction homotopies are supposed to be fast anyway. If

the junction does not complete within this limit (or if it fails for any other reason, by the way), we just dismiss the new meshsize and keep the current one until the next level. This limitation should be set such as not to impair the junctions completion in most cases, as it is just supposed to take care of the isolated problematic junctions at a reasonable cost. Numerical tests suggest that this approach gives satisfying results with a limit of 100 simplices for instance. Besides, setting a very low simplices limit for the junctions (50 or below) actually does more harm than good, as it blocks many junctions and severely degrades the whole adaptive mechanism. On the other hand, setting the constraint too loose (more than 500 for instance) means little improvement, as faulty junctions are then allowed to "waste" more simplices.

A little more sophisticated idea is to cancel the junction before the limit if it does not make sufficient progress (for instance if the parameter λ' has not reached a certain value after a certain number of simplices). However, the choice of these "certain" values is quite hazardous in practice, and the tests made with this idea have not been very convincing so far.

• An interesting, but fortunately rare phenomenon we can observe is when the path following seems to "turn back" to $\lambda = 0$ after a junction. The probable explanation for this is the existence of two distinct zero paths close to each other at a certain level λ_i. A junction that happens here can go wrong and end in a completely labeled face corresponding to the other path. Then the algorithm can follow this second path down to $\lambda = 0$, which by the way gives another zero of $H(\cdot, 0)$. If we draw the path projections (with λ on the vertical axis), this is indicated by a kind of horizontal "jump" at the junction level.



A simple and effective way to prevent this particular behaviour is just to limit the number of simplices allowed for junctions, which is actually already done for performance reasons, as said in the previous point. However, we thus give up the adaptive process at this level, and this approach reaches its limits anyway if the branches are very close (maybe if the junction takes place near a bifurcation for instance). In this case, trying to prevent the "jumps" may require such a low limit that it actually blocks most junction

attempts along the path...

Incidentally, such jumps do not necessarily lead back to $\lambda = 0$, as the path following may well go ahead onto the other branch and eventually converge. An example of such a behaviour can be seen with the orbital transfer problem family studied in chapter 3, more precisely in 3.4 (p. 72).

• A last remark concerning the change of triangulation during the following is that we theoretically loose the property according to which a simplicial algorithm does not cycle. Fortunately this risk seems to be rather hypothetical, as we did not encounter this case in the numerical experiments.

2.3.2 New meshsize computation

So let us assume that we have reached a certain level λ_j, how should we adapt the current meshsize δ in order to improve the path following ? We describe here the two distinct mechanisms of meshsize adaptation we use in practice.

The first one, called "deviation from zero path control", uses the norm of the homotopy H at the current solution z_j as a measure of the path following accuracy. By comparing this norm to a certain tolerance value, it decides whether it is appropriate to uniformly shrink or expand the meshsize, or not.

The second one, referred to as "anisotropic deformation", is a more related to the global direction of the path. It looks at the relative weight of each dimension in the path progression since the previous level, and can then increase or decrease the meshsize along each dimension separately, hence its name. Practical tests show that the first approach is better suited to performance gain, whereas the second is more oriented towards precision improvement. We naturally try then to combine both aspects as efficiently as possible, which we call the "full adaptive mode".

Remark: in both the following mechanisms, the last component of the meshsize, δ_{n+1}, which corresponds to the homotopic parameter λ, is treated separately from the others. This is because this size has a strong influence over the whole path following, as it gives the "height" of the next level, at which the next triangulation change can occur. Therefore, we are more restrictive for changing this size, to keep a better control over the adaptive process. Further details about this particular handling are given below, in particular in the full adaptive mode description.

Deviation from zero path control

Basically, the first method we have used is to adapt the meshsize for the next level with respect to the quality of the following at the current level λ_i. When we think the path following goes well, it should be safe to increase the meshsize in order to improve performance. On the contrary, when we seem to stray away from the path, reducing the meshsize might help to get back on track. The matter is, of course, how we estimate the accuracy of the path following at the current state.

A simple way to obtain a hint about it is to test the validity of the PL approximation of the homotopy over the current face. More precisely, we take the current solution (z^i, λ_i) that is supposed to be a zero of H_T, we just evaluate the real homotopy H at this point and compare its norm to a certain tolerance. Then if the norm of $H(z^i, \lambda_i)$ is low enough, the following can be considered as rather accurate, and we can increase the meshsize. Conversely, if the norm is too high, the following is probably too rough, and the meshsize should be reduced to get closer to the path. In between, we do nothing, as the situation is not bad enough to require a slowdown, but not good enough to allow a safe speedup. This last point is important, as the process must have a kind of inertia to prevent for instance an annoying sequence of expansion/reduction if the norms checked stay in the neighbourhood of the tolerance.

A preemptive remark should be made concerning the "reducing the meshsize to improve the accuracy" part. This is reasonably true in general, from a qualitative point of view. However, there is no guarantee in practice that we can keep the norm of $H(z^i, \lambda_i)$ at arbitrarily low values, even with extremely small meshsizes. Indeed, the term of "tolerance" can be a bit misleading here, as it suggests a strict control of the homotopy norm. Actually, it is more like an indicative "acceptable" value for the norm, which we try (rather loosely) to respect. Now let us have a closer look at this deviation thing.

We use here the following notations:

- z^i denotes the nodes of the zero path on the current level. Thus z^i is the zero of $H_T(\cdot, \lambda_i)$ contained in the completely labeled face on the current level λ_i.
- h_{tol} is the (indicative) tolerance with respect to whom we determine the possible reduction or expansion of the meshsize.

Algorithm:

If $|H(z^i, \lambda_i)| < h_{tol} \times 10^{-1}$

Then $\delta_j \leftarrow \delta_j \times 2 \quad , j \in [1..n+1]$

Else if $|H(z^i, \lambda_i)| > h_{tol} \times 10$

Then $\delta_j \leftarrow \delta_j / 2 \quad , j \in [1..n+1]$

If $|H(z^i, \lambda_i)| > h_{tol}$

Then $\delta_j \leftarrow \delta_j / 2 \quad , j \in [1..n]$

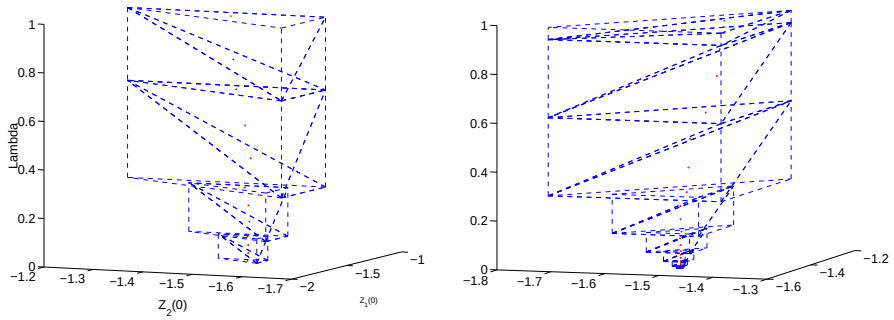
Remarks:

• The size δ_{n+1} is treated slightly differently, as the stepsize for λ has an impact on the whole adaptive process. Indeed, it determines the "height" of the levels $[\lambda_i, \lambda_{i+1}]$, at which the adaptive steps take place, so we are a bit more restrictive about modifying δ_{n+1} than the other δ_j. Apart from this particular case, all other dimensions are treated evenly here, contrary to the anisotropic deformation process described below.

• Numerical experiments tend to indicate that this mechanism is better suited to the performance improvement, via a controlled expansion of the meshsize. On the other hand, trying to satisfy a too low tolerance soon leads to a very small meshsize, with an unreasonable cost in terms of simplices.

• Due to the scaling applied to z and $y = (x, p)$, the order of magnitude of the values of H along the path is generally not too widespread in practice. Therefore, the scope of relevant values for h_{tol} is also limited, fortunately. Typically, the default setting of 10^{-1} has been chosen to obtain a satisfying balance between the performance gain and the precision loss. A value of 1 corresponds to a more aggressive path following, and can give a faster following at the expense of stability. On the other hand, a tolerance of 10^{-2} enforces a more cautious behaviour, which can have a heavy cost in terms of simplices. In our experiments, setting a tolerance above 1 or below 10^{-2} is either overly permissive or restrictive, and both give poor results, with a very unstable or extremely long path following respectively.

Here is an example of this deviation mechanism, applied to the demonstration problem. As this problem is quite easy, the deviation leads to a steady increase of the triangulation size over the path, until the convergence is reached at $\lambda = 1$ (note that the last level is resized with respect to λ so as to reach exactly $\lambda = 1$).



Deviation effect on demo problem - starting meshsize of 10^{-1} *and* 10^{-2}

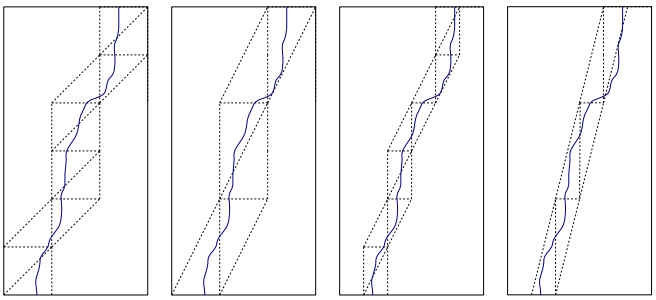
Anisotropic deformation

The second adaptive mechanism tries to take into account the fact that the path is not the same along all components of the unknown z. For the $j-th$ dimension, if the corresponding meshsize value δ_j is too large, the vertices may stray far from the zero path. This is not necessarily harmful, for instance if we loose only a little precision and still manage to have a fast following. However, if the homotopy is sensitive with respect to this particular component of z, these far vertices can get extremely hazardous labelings, or even cause numerical errors during the homotopy evaluation. This difficulty is quite real in our case, where the homotopy is a shooting function and z is an IVP unknown. Due to the numerical integration performed during the homotopy evaluation, relatively small variations of the initial costate can lead to completely different trajectories, or worse even leave the domain of definition of the shooting function. Incidentally, this phenomenon is further amplified by the scaling applied to the unknown z, which sometimes actually needs to be loosed a bit to reduce the shooting function sensitiveness (this is in particular the case for the orbital transfer problems studied in chapter 3, where the integration can be very long).

Obviously, shrinking the meshsize mechanically brings the vertices closer to the zero path, yet a smaller meshsize means more simplices, and a longer path following. Same problem as if the original meshsize is chosen too small at the beginning, where we have no idea of what the path is like. So conversely, if we find that the size δ_j is too small compared to the path progression along the $j-th$ dimension, it means that we are unnecessarily close to the path along this dimension. Therefore, if we increase δ_j a little, we can probably go faster without straying too far away from the path.

All in all, this is a rather qualitative approach: we try to shrink and/or expand the meshsize in order to fit the zero path better, and so hope both to improve the precision and performance of the following. Here is a schematic illustration of this anisotropic deformation mechanism in dimension 2. We

consider an "unbalanced" path, where there is much more progress along the second dimension than the first, so the general idea would be to increase the size δ_2 and decrease δ_1. We represent the possible transverse simplices crossed by the path following with a uniform meshsize ($\delta_1 = \delta_2$). The two following graphs show the expected simplices for a doubled δ_2 and a halved δ_i respectively, while the last graph shows both modifications.



Schematic illustration of the anisotropic deformation mechanism
$\delta_1 = \delta_2 \quad , \quad \delta_2 \leftarrow \delta_2 * 2 \quad , \quad \delta_1 \leftarrow \delta_1/2 \quad , \quad (\delta_1, \delta_2) \leftarrow (\delta_1/2, \delta_2 * 2)$

So what we expect is that the meshsize reduction part would help keeping the vertices not too far from the path, while the meshsize augmentation part would reduce the overall number of simplices needed. These two aspects are naturally antagonists, and we hope that their combination can combine both benefits.

In practice, we use the information on the progress made by the path following over the previous level (from λ_{i-1} to λ_i) to see if some components of the meshsize should be modified for the next level. Of course, we implicitly expect the path to behave more on less the same way over $[\lambda_i, \lambda_{i+1}]$, which is why the anisotropic deviation generally does better with relatively small levels.

Moreover, one has to be careful with these meshsize modifications: brutal changes can have a disastrous effect, leading from a too large meshsize to a too small one and vice-versa. This is why we have eventually limited the changes to a factor between 0.5 and 2, and even drop intermediate values: we just either double or halve the size δ_j if we deem it appropriate, and leave it untouched otherwise. The last point is important, as it introduces a kind of inertia in the process, as with the deviation described above. This aspect is here necessary for the same reasons, namely to avoid a phenomenon of yo-yo, where a size δ_j could undergo an unwanted sequence of increase/decrease once an "appropriate" value has been found with respect to the path.

We use here the following notations:

- δ is the triangulation meshsize, a vector of dimension $n+1$; δ_{n+1} is thus the stepsize with respect to the homotopic parameter λ
- z^{i-1} and z^i denote the nodes of the zero path on the previous and current level. This means that z^{i-1} is a zero of $H_T(\cdot, \lambda_{i-1})$ and z^i is a zero of $H_T(\cdot, \lambda_i)$
- Δ is the progression since the previous level, rescaled with respect to the meshsize, ie
$$\Delta_j = \frac{|z_j^i - z_j^{i-1}|}{\delta_j}, \quad j \in [1..n]$$
- $progress_{up} > 1 > progress_{low} > progress_{toolow}$ are three thresholds whose default values are 2, 0.2 and 0.1. To determine whether the size δ_i is to be increased, decreased or left untouched, we compare the ratio "progress along i-th dimension divided by average progress" to these thresholds, as written below.

Algorithm: for each dimension j, $j \in [1..n]$

If $\Delta_j \geq \frac{\sum_k \Delta_k}{n} \times progress_{up}$

Then $\delta_j \leftarrow \delta_j \times 2$

Else if $\Delta_j \leq \frac{\sum_k \Delta_k}{n} \times progress_{low}$ and $\Delta_j \geq \frac{\sum_k \Delta_k}{n} \times progress_{toolow}$

Then $\delta_j \leftarrow \delta_j / 2$

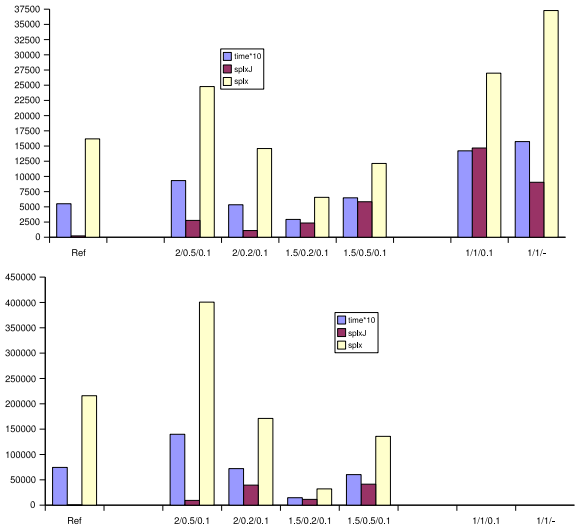
According to the principle explained above, $progress_{up}$ should indicate a progression clearly above the average, and $progress_{low}$ a progression clearly below the average. Additionally, the range between the values $progress_{up}$ and $progress_{low}$ delimits a region of inertia, where the size is not modified. This property is necessary to ensure a better overall stability of the meshsize deformation, same as for the deviation described previously.

The third threshold $progress_{toolow}$ has been introduced specifically to prevent the unwanted behaviour of a size δ_j tending to 0 when there is almost no progression along a certain dimension j. On the path projection along the $j-th$ dimension, this visually corresponds to a very steep or nearly "vertical" path. This situation does actually happen in practice sometimes, so we have to take care of it.

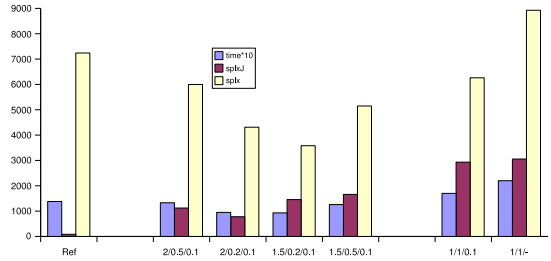
Remarks:
• *The component δ_{n+1} is not modified in this process, as we try not to meddle too much with the size corresponding to λ, which can already be modified*

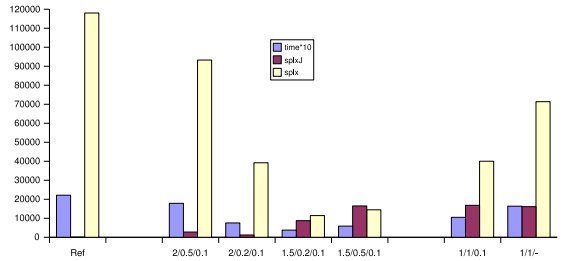
in the deviation control process.
• *Concerning the double objective of the anisotropic deformation, the graphs above show of course an idealistic situation. It appears in practice that this mechanism is better suited to the precision improvement rather than the performance gain. This is not really an inconvenience, as it nicely complements the effects of the deviation described above.*

Here is a comparison of different settings for the three anisotropic thresholds $progress_{up,low,toolow}$, on the discretized singular arc problems studied in chapter 4. For each setting the columns indicate the total time in seconds (rescaled by a factor 10 for clarity), the number of junction simplices, and the number of simplices for the main path. The first group of columns on the left corresponds to the reference uniform triangulation. The four groups in the middle correspond to the $(2, 0.5, 0.1), (2, 0.2, 0.1), (1.5, 0.2, 0.1), (1.5, 0.5, 0.1)$ settings. The two last groups on the right correspond to the $(1, 1, 0.1), (1, 1, -)$ settings (which do not converge for the $K_1(0.01)$ on Problem 1).



Anisotropic settings comparison - Problem 1 (dim 101)
Triangulations $K_1(0.1)$ *and* $K_1(0.01)$.





Anisotropic settings comparison - Problem 2 (dim 82)
Triangulations $K_1(0.1)$ *and* $K_1(0.01)$.

First, the poor performance (and even failure for the $K_1(0.01)$ on Problem 1) of the two groups on the right illustrates the necessity of some inertia zone. The $(1, 1, -)$ setting, which *always* modifies *all* components of the meshsize, leads to an insane amount of junction simplices and main simplices (sometimes even more than the reference !), with of course a huge execution time. The $(1, 1, 0.1)$ case is slightly faster, as it avoids the abusive reduction of meshsize components with a low progression. However, the identity of $progress_{up}$ and $progress_{low}$ still impairs this configuration greatly.

Then if we look at the four configurations in the middle, we observe the same general allure on the four test suites. The execution time is better for the three on the right, and especially for the $(1.5, 0.2, 0.1)$, with a particularly low number of main simplices. Of course, this is not that surprising as it corresponds to the most aggressive setting: a meshsize expansion for all components with a progression only 50% above the average, and a reduction only for components below one fifth of the average progression. On the other hand, the most conservative (in terms of meshsize) setting $(2, 0.5, 0.1)$ is logically slower, with a rather high number of main simplices.

The second and fourth anisotropic configurations, with the $(2, 0.2, 0.1)$ and $(1.5, 0.5, 0.1)$ settings, correspond to the largest and shortest inertia zones. This is coherent with the number of junctions simplices used, which tends to be minimal (respectively maximal) for these settings. Both give moderate performances in terms of execution time, between the first and third (fastest) configurations.

Here are the numerical results, with for each setting the number of simplices for the main path, the junction simplices, the execution time (in seconds), and the euclidian norm of the homotopy at the end of the path $|H(z_1, 1)|$.

Problem 1, $K_1(10^{-1})$ and $K_1(10^{-2})$

Settings	Main	Junc.	Time	$\lvert H(z_1,1)\rvert$
2/0.5/0.1	24782	2774	932	0.122
2/0.2/0.1	14589	1101	534	0.078
1.5/0.2/0.1	6566	2345	293	0.077
1.5/0.5/0.1	12137	5840	648	0.100
1/1/0.1	26986	14673	1421	0.074
1/1/−	37277	9041	1573	0.158
Main	Junc.	Time	$\lvert H(z_1,1)\rvert$
400673	9453	13991	0.133
171224	39554	7214	0.098
32156	11441	1473	0.211
135807	41415	6031	0.083
xxx	xxx	xxx	*xxx*
xxx	xxx	xxx	*xxx*

Problem 2, $K_1(10^{-1})$ and $K_1(10^{-2})$

Settings	Main	Junc.	Time	$\lvert H(z_1,1)\rvert$
2/0.5/0.1	5998	1122	133	0.965
2/0.2/0.1	4311	778	95	0.894
1.5/0.2/0.1	3582	1454	93	1.040
1.5/0.5/0.1	5149	1660	126	0.890
1/1/0.1	6261	2932	170	0.829
1/1/−	8926	3056	220	0.812
Main	Junc.	Time	$\lvert H(z_1,1)\rvert$
93281	2778	1789	0.288
39208	1221	755	0.763
11424	8743	379	1.125
14462	16503	589	1.016
40031	16833	1052	0.432
71370	16122	1637	0.317

Despite the aggressive setting $(1.5, 0.2, 0.1)$ being faster, we see that it also degrades the final norm quite a bit. Conversely, the inertial configuration $(2, 0.2, 0.1)$ seems to do well with the final norm, and seems a good candidate for the combination with the deviation control mechanism. On a side note, the two non-inertial configurations have a good final norm (when they converge...), which is merely a consequence of the small meshsize they produce (besides the quite long execution time).

Full adaptive mode

Now we would like to combine these two mechanisms in the most constructive possible way. As mentioned before, the main benefit we can expect from the deviation control is a performance gain, so we stick with it. Therefore we choose to configure the anisotropic refinement with rather conservative settings, so that it can primarily alleviate the resulting precision loss. The resulting formulation generally leads to an interesting gain in the number of simplices, without degrading the final precision. Actually, we often obtain a better accuracy at the solution than with a basic uniform triangulation, in

addition to the path speedup.

Summary:
In practice, we typically use the following settings for the full adaptive mode:
- $h_{tol} = 0.1$ *for the deviation control*
- $progress_{up,low,toolow} = (2, 0.2, 0.1)$ *for the anisotropic deformation*

These values tend to give the overall best results on the whole set of problems, even if better custom settings can be found for each problem individually with a bit of tuning.

To finish with, here are several remarks about the whole adaptive triangulation process, and especially the stepsize with respect to λ.

The meshsize adaptation is done for the next level, based on the path following over the previous level. We recall that the "height" of these levels is fixed by the last component of the meshsize, δ_{n+1}. If this size is large, there are only few levels over the path, and the meshsize adaptation is too limited to be effective. Conversely, if this size is too small, then the risk is to spend too much time in changing the meshsize at very close intervals. So a middle ground has to be found, with two practical consequences.

First, we have to be cautious with the modifications of δ_{n+1} in the adaptive process. This is why we have eventually removed it from the anisotropic deformation, to avoid potential interferences with the modifications made by the deviation control part. It is then easier to keep control of its evolution along the path, and we have furthermore enforced a lower bound δ_{min} on δ_{n+1}, to prevent it from becoming overly small, for instance if the deviation tolerance is set too low. Moreover, we also sometimes have to "crop" the value δ_{n+1} to prevent the apparition of vertices above $\lambda = 1$ or below $\lambda = 0$. All in all, at the end of the new meshsize computation we enforce the following constraint

$$\delta_{min} \leq \delta_{n+1} \leq min(\lambda, 1 - \lambda),$$

with for instance

$$\delta_{min} = 10^{-6}.$$

A visual illustration of this particular point can be found on the graphs on page 43, where we can see that the last level is resized to match the $\lambda = 1$ limit. Actually, the first values of δ_{n+1} on these examples are also resized to respect the $\lambda = 0$ bound, but this does not appear on the graph due to the small starting meshsizes.

Then, it explains why in practical tests the adaptive mode does better (compared to the fixed triangulation mode) with an initial δ_{n+1} of 10^{-2} rather than 10^{-1} (see in particular 3.4). For our problems, setting initial values of δ_{n+1} outside of $[10^{-2}, 10^{-1}]$ usually gives poor results anyway, with either an awfully long or unstable path following. So in the end we have come to use almost only these two settings, namely 10^{-1} for ordinary cases, and 10^{-2} for fast problems (the singular arcs problems for instance, see chapter 4).

Last but not least, it should be noted that the difficulties with the junctions have actually plagued the whole adaptive meshsize system for a long time. At first, the early formulations for the junctions still used derivatives, and the new meshsize computation rules were much less restricted than in the current version. Without even mentioning the important loss of stability, the desperately high rate of junction failures, and the heavy cost of the lucky successful ones, were such that the adaptive mode was utterly unusable most of the time. A first noticeable improvement was to put some strict limitations on the possible meshsize changes, with a factor of 2 or 0.5 only at a time. This restored the stability back to normal, and made the junctions easier, in addition to the introduction of the third formulation. Then the adaptive mode became practically interesting at last, even if the experiments with the orbital transfer problems in chapter 3 show that the numerical cost of the junctions can still be sometimes quite significant.

2.4 Solution refining

We try in this part to address the question of the final precision of the simplicial algorithm. In our case, the primary objective of the continuation is to provide a suitable initial point for a shooting method. Therefore the accuracy of the solution at the end of the path can at first be considered of secondary interest. However, in some cases we observe that this final shoot takes indeed a long time, nearly as much as the whole path following ! So we wonder if a better initial point could make the shoot converge faster, as it basically uses a quasi Newton method after all. Another situation is when the final shoot does not converge in practice, due to intrinsic difficulties such as singular arcs. In this case, we would still like to obtain a more accurate solution than the one we would normally have.

So our objective here is to improve the precision of the solution at the end of the path following, for which we have thought of two methods. The first one aims at a better solution by the use of a refining triangulation before the convergence at $\lambda = 1$. The second one is based on the principle

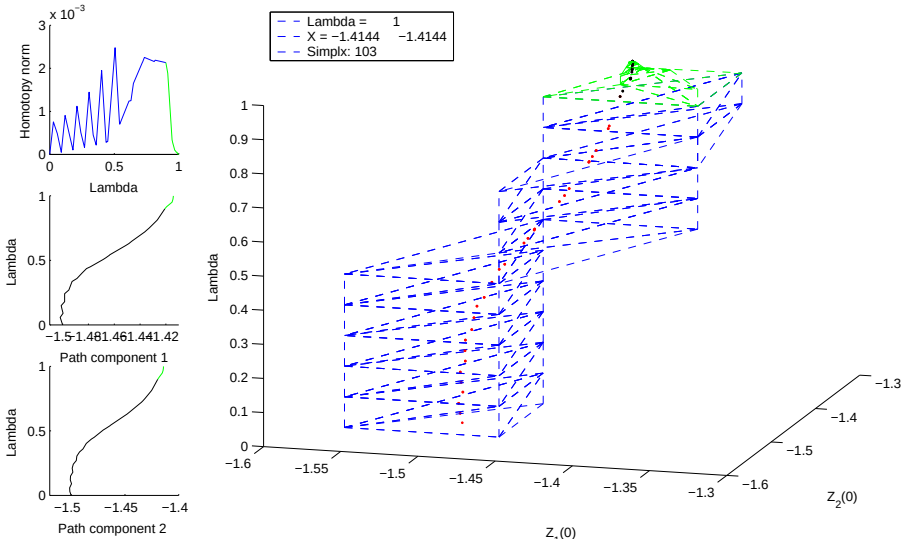
of Merril's restart algorithm, and rather tries to refine the solution after the convergence.

2.4.1 Refining before convergence

The idea here is to try to reuse the principle of refining triangulations, but only near the convergence. We thus hope to avoid the difficulties met by these triangulations (especially J_3) at the beginning of the path. Basically, we begin the path following with an uniform meshsize, a K_1 for instance, from $\lambda = 0$ until a certain value λ_{refine}. At this point we proceed to a triangulation change, and continue the following with a J_3 (or J_4), until normal convergence.

Remark: the triangulation change at $\lambda = \lambda_{refine}$ does not require a junction homotopy in practice. The first completely labeled face for the refining triangulation is rather built directly from the last completely labeled face of the uniform triangulation, which is possible as the size does not change.

The following graph illustrates this approach on the demonstration problem.



Solution refining: K_1 until $\lambda = 0.9$, followed by J_3 refinement

Solution	Triangulation	Simplices	Norm
$\lambda = 0.9$	$K_1(0.1)$	30	$2.13\ 10^{-3}$
$\lambda = 1$	$J_3, (\epsilon = 10^{-9})$	90	$4.47\ 10^{-13}$

The effect of the triangulation change at $\lambda = 0.9$ is quite obvious on the first graph on the right, which shows the homotopy norm along the pathh. We immediately observe a quick decrease of the homotopy norm as the J_3

refinement begins, with a very good final norm of $1.47\ 10^{-13}$. We notice that the refinement already takes thrice the number of simplices of the main path until $\lambda = 0.9$, which is one of the flaws of this idea (even if the global execution time is here below 2 seconds).

Although things go well on this simple example, this formulation actually still suffers from some flaws of the refining triangulations, namely some convergence difficulties and a high cost in terms of simplices. This relative lack of success is another hint that refining triangulations may not be adapted in our case after all.

2.4.2 Refining after convergence

The second idea to obtain a more accurate solution is based on the principle of Merril's restart algorithm (see [2], p181 or [24], p74 for instance). More precisely, starting from the solution z_1^* at the end of the path, we try to perform a sequence of junctions homotopies at the $\lambda = 1$ level, with decreasing meshsizes. In practice, we divide the meshsize by 2 at each attempt, so that the junctions should not be too difficult to complete. This approach does not suffer from the convergence problems mentioned before: if one of the junctions fails, we simply ignore it and move on to the next meshsize value. We choose to stop the process after a fixed number of refine attempts, as setting a stopping criterion over the norm of $H(\cdot, 1)$ at the solution would not guarantee the termination. Therefore, we can roughly predict the final meshsize used, depending on the initial meshsize and the maximal number of refine attempts. For instance, starting from an initial size $\delta = 10^{-1}$, 10 refine attempts lead to $\delta \approx 10^{-4}$, and 20 to $\delta \approx 10^{-7}$. If the adaptive mode is enabled, there is some degree of variation, as the meshsize at the end of the path has probably evolved from its initial value.

Here are the results of this method applied to the demo problem: we perform the path following with an ordinary $K_1(0.1)$, then try 20 successive refinements.

Solution	Meshsize	Norm
Arrival at $\lambda = 1$	0.1	$2.13\ 10^{-3}$
Refinement 10	$1.95\ 10^{-4}$	$1.92\ 10^{-5}$
Refinement 20	$1.91\ 10^{-7}$	$4.29\ 10^{-10}$

The norm decrease is quite regular on this simple problem, which is not the case in general. Also, the refinements are here almost instantaneous (with a total of 1 second for the initial path following plus the 20 junctions at $\lambda = 1$), which once again is not a general behaviour.

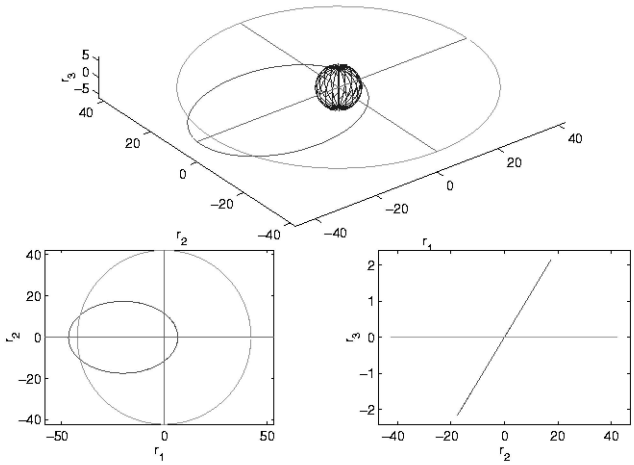
In fact, contrary to the adaptive meshsize junctions that occur during the path following, these junctions taking place after the convergence often require more simplices, as the homotopy is less regular. Therefore, the maximum number of simplices allowed for junctions should be raised accordingly, up to 10000 for instance, depending on the problems.

Furthermore, it should be noted that the numerical cost of these refinement junctions is usually quite significant, which is not to be overlooked. Therefore this method seems the more appropriate as a replacement for the final shooting, when the latter does not converge. In this case we have few other options to obtain a better solution, apart from retrying the path following with a smaller meshsize (or using more cautious adaptive settings). This solution refinement method can then prove interesting in terms of final precision / overall cost ratio. An example of practical application, in addition to the adaptive mode, is described later in the singular arcs part, more precisely in 4.3.3, page 106.

Chapter 3

Orbital Transfer problems

The problem we address here is a case of Earth orbital transfer, studied in the context of a collaboration with the CNES (contract R&T 02/CNES/0257/00-DPI 500, see [16]). We want to move a satellite from a low, elliptic, and inclined initial orbit to an geosynchronous equatorial orbit, while maximizing the final mass (payload).



Orbital transfer (the inclination on the third graph is rescaled)

The chosen objective, the maximization of the final mass, leads to a bang-bang structure for the optimal control, with a huge number of commutations for low thrusts. As a consequence, it is not possible in practice to solve this problem directly by single shooting. The extensive study (with a differential continuation method) and physical interpretations of this problem can be found in Thomas Haberkorn's PhD thesis ([26]), some of these results being also presented in [25].

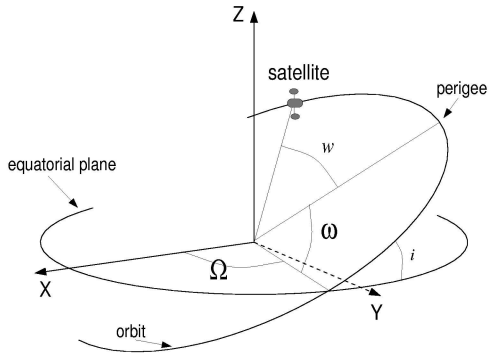
3.1 Problem statement

We consider that the forces applied to the satellite are the Earth attraction (central force) and the propulsor's thrust:

$$\ddot{r} = -\mu_0 \frac{r}{|r|^3} + \frac{u}{m}$$

with r the position vector, u the thrust (ie the control), m the mass of the satellite and μ_0 the gravitational constant of the Earth ($\mu_0 = \mathcal{G}m_T$, with $\mathcal{G}$ the universal gravititational constant and m_T the masse of the Earth), and $|.|$ the euclidian norm.

The state vector consists of the position, speed and mass of the satellite at a given time t. It is of course possible to express the position and speed in a geocentric Cartesian system, yet according to the expected high number of revolutions, this would lead to strong oscillations, which is detrimental to numerical stability. This is why we prefer to use a modified set of classical orbital elements, which describes the movement of the satellite in a more orbit-related point of view. There we use the first five components of the state vector to characterize the osculating orbit (the orbit the satellite would follow if no thrust was applied), while the sixth component indicates the current position of the satellite on this orbit. As the orbit deformation is quite smooth, especially for low thrust transfers, this guarantees a very good numerical stability for our state vector, which would not be the case with the cartesian expression. This particular choice of coordinates is illustrated below.



Orbital parameters

with

- P and e: ellipse parameter and eccentricity
- θ: true anomaly
- Ω: ascending node longitude
- ω: argument of perigee

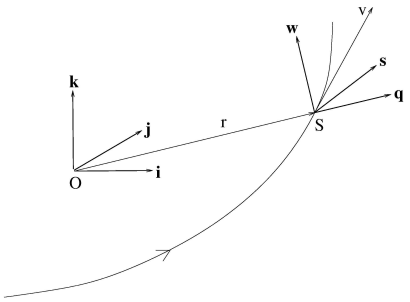
- i: inclination

Let us now define the state variables in $\mathbf{R^7}$:
• Orbit parameter P
• Eccentricity vector (e_x, e_y), in the orbit plane, oriented towards perigee
• Rotation vector (h_x, h_y), in the equatorial plane, colinear to the intersection of orbit and equatorial planes
• True longitude L
• Mass m

According to the previous notations, these parameters are defined as:

$$\begin{array}{lll} e_x = e\ cos\ (\Omega + \omega) & , & e_y = e\ sin\ (\Omega + \omega) \\ h_x = tan(i/2)cos\Omega & , & h_y = tan(i/2)sin\Omega \\ L = \Omega + \omega + \theta & & \end{array}$$

As for the three-dimensional control, we choose to express it in the moving reference frame attached to the satellite, as shown below.



Control expression in the moving reference frame (q, s, w)

The normalized control u (such as the thrust $\overrightarrow{T(t)} = u(t)\ T_{max}$) is thus expressed in $\mathbf{R^3}$ as: radial thrust q, transverse thrust s and normal thrust w.

If we denote T_{Max} the maximal thrust and I_{sp} the specific impulse of the propeller, the chosen approximation of gravitational forces leads to the

following dynamics of the problem (see for instance [16]):

$$\left\{\begin{array}{ll}
\dot{P}(t) & = \frac{2T_{Max}}{m(t)}\sqrt{\frac{P^3(t)}{\mu_0}}\frac{s(t)}{Z(t)} \\
\dot{e}_x(t) & = \frac{T_{Max}}{m(t)}\sqrt{\frac{P(t)}{\mu_0}}\frac{1}{Z(t)}[Z(t)sin(L(t))q(t) + A_1(t)s(t) \\
 & \quad -e_y(t)(h_x(t)sin(L(t)) - h_y(t)cos(L(t)))w(t)] \\
\dot{e}_y(t) & = \frac{T_{Max}}{m(t)}\sqrt{\frac{P(t)}{\mu_0}}\frac{1}{Z(t)}[-Z(t)cos(L(t))q(t) + A_2(t)s(t) \\
 & \quad +e_x(t)(h_x(t)sin(L(t)) - h)y(t)cos(L(t)))w(t)] \\
\dot{h}_x(t) & = \frac{T_{Max}}{2m(t)}\sqrt{\frac{P(t)}{\mu_0(t)}}\frac{X(t)}{Z(t)}cos(L(t)).w(t) \\
\dot{h}_y(t) & = \frac{T_{Max}}{2m(t)}\sqrt{\frac{P(t)}{\mu_0(t)}}\frac{X(t)}{Z(t)}sin(L(t)).w(t) \\
\dot{L}(t) & = \sqrt{\frac{\mu_0}{P^3(t)}}Z^2(t) \ + \ \frac{1}{m(t)}\sqrt{\frac{P(t)}{\mu_0}} \\
 & \quad \times \frac{1}{Z(t)}(h_x(t)sin(L(t)) - h_y(t)cos(L(t)))w(t) \\
\dot{m}(t) & = -\frac{T_{Max}}{I_{sp}g_0}\ |(q(t), s(t)), w(t)|
\end{array}\right.$$

With

$$\left\{\begin{array}{l}
Z(t) = 1 + e_x(t)\cos(L(t)) + e_y(t)\sin(L(t)) \\
A_1(t) = e_x(t) + (1 + Z(t))\cos(L(t)) \\
A_2(t) = e_y(t) + (1 + Z(t))\sin(L(t)) \\
X(t) = 1 + h_x^2(t) + h_y^2(t)
\end{array}\right.$$

If we note $x = (P, e_x, e_y, h_x, h_y, L)$ and $u = (q, s, w)$ we obtain the following formulation of the maximal mass orbital transfer problem, with the initial and terminal conditions corresponding to the two orbits:

$$(P_{m_f})\left\{\begin{array}{l}
Max\ m(t_f) \\
\dot{x} = a(x) + \frac{T_{Max}}{m}B(x)u \\
\dot{m} = -\frac{T_{Max}}{I_{sp}g_0}\ |u| \\
|u| \leq 1 \\
IC : x(t_0) = (11625; 0.75; 0; 0.0612; 0; \pi; 1500) \\
TC : x(t_f) = (42165; 0; 0; 0; 0; free; free) \\
t_0 = 0 \\
t_f\ fixed
\end{array}\right.$$

Contrary to minimum-time problems, in which the transfer time is the objective to be minimized, and thus the final time t_f is free, we consider here a fixed time transfer. The reason for this choice is that it is not obvious that the minimum fuel problem with free final time has a solution.

Besides, the actual criterion used for the maximization of the payload is not $Max\ m(t_f)$ but (which is equivalent per the mass dynamic)

$$Min \int_{t_0}^{t_f} |u(t)|dt.$$

The Hamiltonian is thus defined by (t is here omitted for clarity)

$$\mathcal{H}(x, m, p, p_m, u) = (1 - \frac{T_{Max}}{I_{sp}g_0} p_m)\ |u| + \frac{T_{Max}}{m}(B(x)u|p)\ +\ (a(x)|p).$$

If $B^t(x)p \neq 0$ then let us define the **switching function** ψ:

$$\psi(x, m, p, p_m) = 1 - \frac{T_{Max}}{I_{sp}g_0}\ p_m\ -\ \frac{T_{Max}}{m}|B^t(x)p|.$$

The application of Pontryagin's Maximum Principle then leads to the following expression of the optimal control

$$\begin{cases} u = -\frac{B^t(x)p}{|B^t(x)p|} & if\ \psi(x, m, p, p_m) < 0 \\ u = -\alpha\frac{B^t(x)p}{|B^t(x)p|} \quad \alpha \in [0, 1] & if\ \psi(x, m, p, p_m) = 0 \\ u = 0 & if\ \psi(x, m, p, p_m) > 0 \end{cases}$$

Else if $B^t(x)p = 0$ we have

$$\begin{cases} u \in S(0, 1) & if \quad 1 - \frac{T_{Max}}{I_{sp}g_0}\ p_m < 0 \\ u \in B(0, 1) & if \quad 1 - \frac{T_{Max}}{I_{sp}g_0}\ p_m = 0 \\ u = 0 & if \quad 1 - \frac{T_{Max}}{I_{sp}g_0}\ p_m > 0 \end{cases}$$

We can see that this control has a bang-bang structure, as its norm switches between 0 and 1 at zeros of the switching function ψ. We now make two assumptions, which are numerically verified:
• We assume that $B^t(x)p$ is non-zero on $[t_0, t_f]$
• There is no **singular arc**, that is to say that we do not have $\psi(x, m, p, p_m) = 0$ on any finite interval.

3.2 Resolution approach

As mentioned before, the resolution approach for this family of problems is described in detail in [26], and we just recall here the general idea. While maximizing the final mass leads to a difficult problem, due to the discontinuous nature of the optimal control, changing the objective into the quadratic

one $Min \int_{t_0}^{t_f} |u(t)|^2 \, dt$ gives a much more regular problem, with a continuous control. So we perform a kind of "energy to mass" continuation, by considering the family of problems (BVP_λ) with the following objective

$$J_\lambda = \int_{t_0}^{t_f} \lambda|u(t)| + (1-\lambda)|u(t)|^2 \, dt.$$

and defining the homotopy as the corresponding shooting function

$$H : (z, \lambda) \mapsto S_\lambda(z).$$

However, the first numerical experiments on this problem have shown the existence of annoying local solutions. These were dealt with by using the longitude instead of time as the integration variable, which eventually leads to a formulation with both fixed final time and longitude.

Note: a consequence of this longitude formulation is that the resulting shooting function is overly sensitive with respect to the costate p_L associated to the longitude. Therefore, using the usual scaling on this component of the (IVP) unknown leads to numerical difficulties. As we would like to keep the method as little problem dependant as possible, we have introduced a less strict scaling mode instead of merely adjusting this component manually. This "soft scaling" only rescales values to the interval $[10^{-2}, 10]$ instead of $[10^{-1}, 1]$.

For the integration of the (IVP) required in the evaluation of the homotopy, as we expect a bang-bang control with a high number of commutations, we primarily use a variable step integrator (RKF45), which performs quite well in practice. We have also tried several fixed step Runge Kutta formulas, which have all failed to complete the path following. The cause may be the progressive appearance of the bang-bang control structure, which is difficult to handle correctly with a fixed step integration (the transfers at low thrusts are quite long, with several hundreds of commutations).

3.2.1 Initialization

For this problem family, the initialization of the continuation (ie solving the problem for $\lambda = 0$) actually becomes non-trivial when the maximal thrust T_{max} (in Newtons) reaches low values. This difficulty is solved in [26] by using a preliminary continuation on the initial conditions, but we have found here another possibility.

More precisely, we try to solve the problem at $\lambda = 0$ by single shooting, but with a fixed step integration instead of RKF45. The initialization for this shooting attempt is very crude, as we set $p(0) = 0$. However, we set

the more sensitive final longitude and time according to the empiric laws presented in [15], which state that both the final time and longitude are actually inversely proportional to the maximal thrust. Then, the (fixed) number of integration steps is also set inversely proportional to T_{max}, as summarized below.

T_{max}	L_f	t_f (h)	RK4 steps
10N	29.89539	150	300
5N	56.64919	300	600
1N	270.67959	1 500	3 000
0.5N	538.21759	3 000	6 000
0.2N	1340.83159	7 500	30 000
0.1N	2678.52159	15 000	50 000

Initial shooting at $\lambda = 0$ initialization - Fixed step RK4 integration

Here are the results of this fixed step initial shooting, which are quite good.

T_{max}	t_f (h)	Solution norm	H calls	Time(s)
10N	153.87	$6.75\ 10^{-15}$	113	1
5N	304.76	$2.31\ 10^{-15}$	169	2
1N	1 529.65	$3.74\ 10^{-15}$	147	7
0.5N	3 065.80	$6.05\ 10^{-15}$	145	13
0.2N	7 659.83	$2.20\ 10^{-14}$	183	86
0.1N	15 315.79	$2.14\ 10^{-14}$	123	96

Initial shooting at $\lambda = 0$ results - Fixed step RK4 integration

Now the interesting point: these solutions at $\lambda = 0$, which correspond to a RK4 fixed step integration, actually allow us to initialize properly the simplicial algorithm, even though the (IVP) integration along the path is performed with RKF45. Thus there is no need for the preliminary continuation on the initial conditions anymore, which is a significant gain of time at low thrusts.

Remark: it would be interesting to check whether this fixed step solution would also be a sufficient initialization for the differential continuation.

3.2.2 Path following and solution

As mentioned before, the path following uses RKF45 for the integration, with a moderate precision (we set 10^{-8} and 10^{-6} for the absolute and relative error tolerances). After the convergence at $\lambda = 1$, we then proceed with a final shoot with a higher precision, namely $(10^{-12}, 10^{-10})$ for the tolerances. Here are summarized the results of the path followings and final shoots for thrusts ranging from 10 to 0.1 Newton. The columns correspond to

- **Initial norm** indicates the norm of H at the fixed step solution (which is there re-integrated with RKF45).
- **Simplices** indicates the number of simplices followed for the path

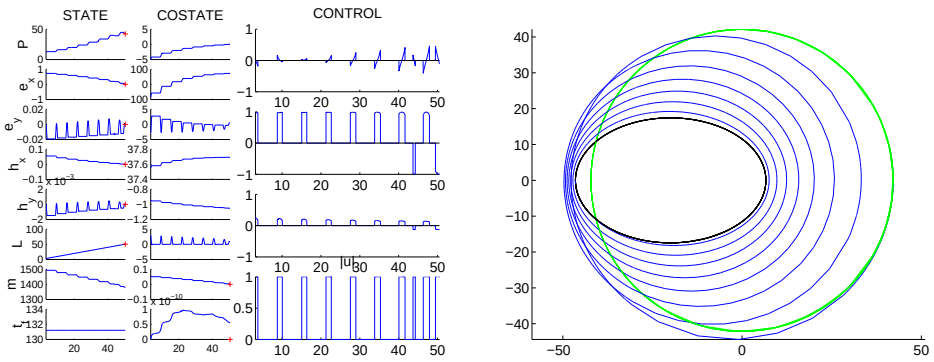
- **Time (s)** indicates the execution time in seconds.
- **Objective** indicates the value of the objective (fuel consumption in kgs).
- **Final norm** indicates the norm of H at the solution after the final shoot.

T_{max}	Initial norm	Simplices	Time (s)	Objective	Final norm
10N	$3.29\ 10^{-4}$	1336	20	121.21	$3.43\ 10^{-10}$
5N	$1.55\ 10^{-4}$	1909	51	121.58	$4.63\ 10^{-10}$
1N	$3.09\ 10^{-3}$	840	132	121.78	$8.02\ 10^{-11}$
0.5N	$3.58\ 10^{-3}$	2028	548	121.69	$1.12\ 10^{-9}$
0.2N	$6.34\ 10^{-3}$	912	661	121.71	$3.21\ 10^{-9}$
0.1N	$8.33\ 10^{-3}$	775	1252	121.70	$1.75\ 10^{-6}$

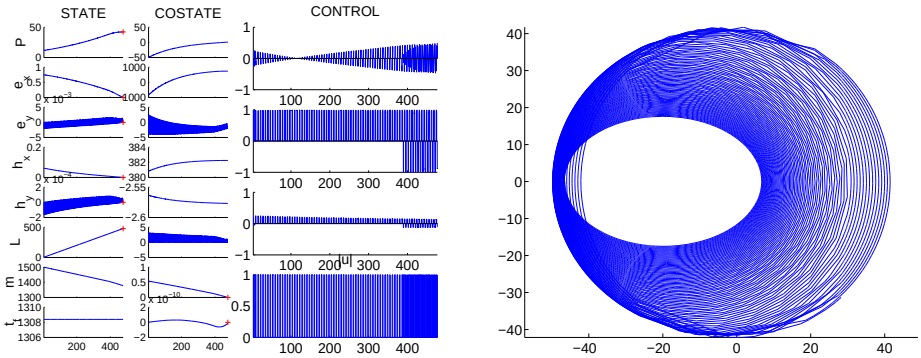
Concerning the path following, we observe that the initialization works fine, with a first junction generally requiring less than 100 simplices despite the integrator change. This is coherent with the good norm (around 10^{-3}) for the re-integrated fixed step solution. The following itself converges normally with a uniform $K_1(10^{-1})$ triangulation, with 1000 to 2000 simplices. As for the final shoot, we observe that for lower thrusts T_{max} the precision decreases, while the time taken by the shoot increases.

Compared to the differential continuation used in [26], the final norms are similar, and the objective values coincide as well. Actually, the solutions found are the same, which is of course quite reassuring. Concerning the performance, the execution time of the simplicial method is roughly up to 3 times longer than the differential continuation. This is not surprising, as PC methods are known to be faster than PL methods, when they converge.

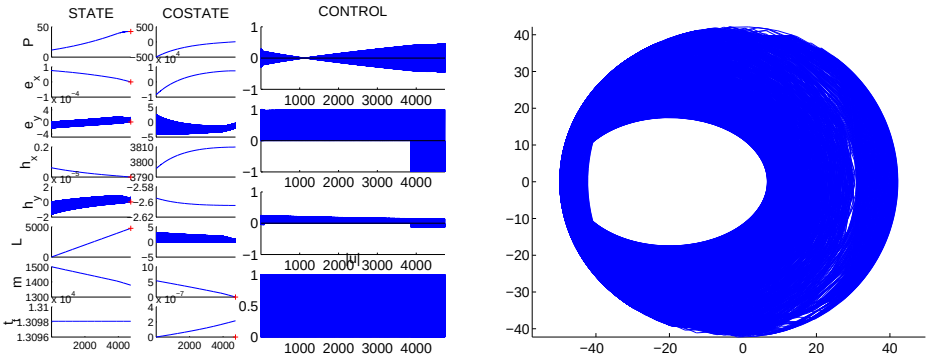
Here are the solutions corresponding to $T_{max} = 10, 1$ and $0.1N$. The two columns on the left represent the state and costate components, the column in the middle the three components and norm of the control, and finally the trajectory (in 2D) from the initial elliptic orbit to the geostationary. The switchings of the bang-bang control are clearly visible (for the $10N$ graph anyway...) on the control norm, ie on the bottom graph in the middle.



Solution and trajectory for $T_{max} = 10N$



Solution and trajectory for $T_{max} = 1N$



Solution and trajectory for $T_{max} = 0.1N$

Remark: if we try to re-integrate these solutions with a fixed step formula, with a second shoot using RK4 for instance, it seems difficult to obtain the same precision, even with a very large number of steps. This might be another hint of the inadequacy of (uniform at least) fixed step integrators for bang-bang control with many commutations.

Indeed, the correct handling of this bang-bang control (with several hundreds of commutations for low thrusts), with no a priori knowledge about the structure, is an impressive achievement of the combined continuation/single shooting approach. On a side note, we would like to mention that the resolution of these problems highlighted some interesting physical properties and interpretations, and refer once again the readers interested in these matters to Thomas Haberkorn's thesis ([26]).

3.3 Comparison of different integrators

We compare in this part two other variable step integrators for the final shoot, namely the DOP853 and ODEX codes by E.Hairer and G.Wanner, described in details in [27]. We use the same absolute and relative tolerances (ie 10^{-12} and 10^{-10}) for all the shoots, while keeping the RKF45 with

$(10^{-8}, 10^{-6})$ for the path. We first examine the shoot results, then have a closer look at the corresponding trajectories.

3.3.1 Final shoot results

We indicate here the shoot results for the three integrators (of course, the RKF45 results are the same as mentioned before).
- **Steps** indicates the number of integration steps at the solution
- **Final norm** is the norm of H after the shoot
- **Objective** is the objective value at the solution
- **Shoot (s)** indicates the time taken by the shoot (in seconds)
- **Shoot (H)** indicates the number of homotopy evaluations for the shoot

T_{max}	Steps	Final norm	Objective	Shoot (s)	Shoot (H)
10N	1 552	$3.43\ 10^{-10}$	121.21	4	55
5N	2 932	$4.63\ 10^{-10}$	121.58	11	87
1N	15 170	$8.02\ 10^{-11}$	121.78	38	51
0.5N	27 768	$1.12\ 10^{-9}$	121.69	105	74
0.2N	67 769	$3.21\ 10^{-9}$	121.71	161	49
0.1N	134 528	$1.75\ 10^{-6}$	121.70	350	49

Solutions for 10,5,1,0.5,0.2 and 0.1N - RKF45 integrator

T_{max}	Steps	Final norm	Objective	Shoot (s)	Shoot (H)
10N	670	$2.86\ 10^{-10}$	121.21	5	57
5N	1 328	$1.52\ 10^{-11}$	121.58	15	88
1N	7 095	$5.82\ 10^{-10}$	121.78	55	54
0.5N	12 805	$3.51\ 10^{-9}$	121.69	164	96
0.2N	31 894	$1.13\ 10^{-8}$	121.71	194	45
0.1N	63 810	$1.92\ 10^{-5}$	121.70	570	65

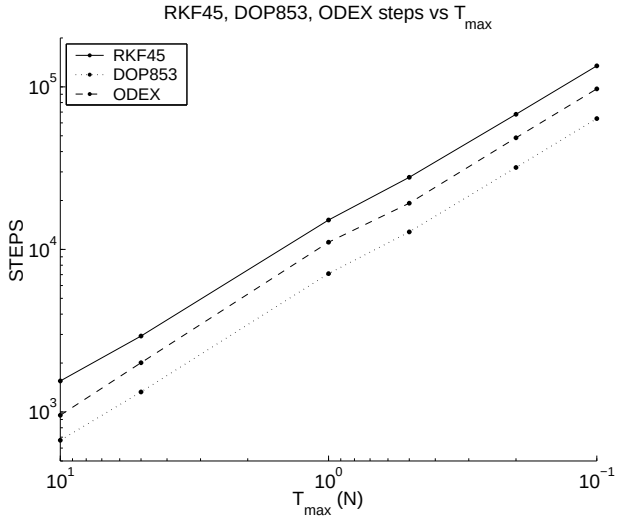
Solutions for 10,5,1,0.5,0.2 and 0.1N - DOP853 integrator

T_{max}	Steps	Final norm	Objective	Shoot (s)	Shoot (H)
10N	954	$3.96\ 10^{-10}$	121.21	6	58
5N	2 008	$1.07\ 10^{-11}$	121.58	13	69
1N	11 072	$9.61\ 10^{-10}$	121.78	58	51
0.5N	19 242	$7.77\ 10^{-10}$	121.69	216	109
0.2N	48 591	$1.49\ 10^{-8}$	121.71	365	76
0.1N	97 104	$1.13\ 10^{-2}$	121.70	381	37

Solutions for 10,5,1,0.5,0.2 and 0.1N - ODEX integrator

The solutions are similar (identical up to 3 to 5 digits depending on the components of z for $0.1N$ for instance), with the same objective values. The final norms are also quite close, with the exception of the poor result of ODEX in the $0.1N$ case (the solver stops here prematurely, with only 37 homotopy calls). Concerning the shoot time the order is less regular, but overall RKF45 appears to be the fastest, followed by DOP853, then ODEX.

Now if we also look at the integration steps with respect to the thrust T_{max}, we see some interesting things, as shown on the graph below.



Integration steps with respect to T_{max} - RKF45, DOP853 and ODEX (logarithmic scale on both axes)

For all three integrators, the number of integration steps taken is numerically inversely proportional to the maximal thrust (as are also the transfer time and total longitude), which illustrates well the extreme regularity of the orbital transfer problem family with respect to T_{max}.

$$Integration\ steps \times T_{max} \approx C^t.$$

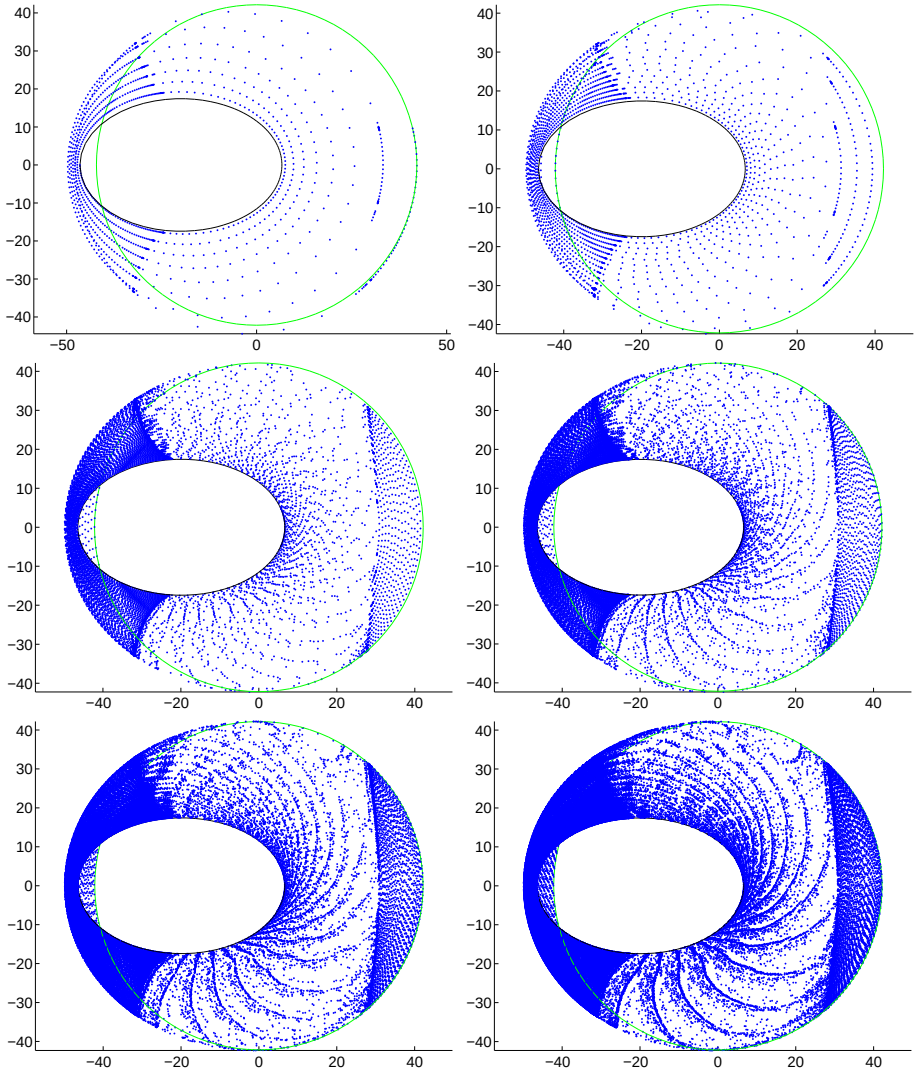
Moreover, we observe a steady relation in the number of integration steps, with DOP853 taking about half the steps needed by RKF45, while ODEX stands roughly halfway between the two. These relations seem to hold extremely well, regardless of the maximal thrust.

3.3.2 Solution trajectories

Let us have now a closer look at the trajectories obtained by the three variable step integrators. The trajectories shown below correspond to the solution given by the final shoot at the end of the path following, for the thrusts $T_{max} = 10, 5, 1, 0.5, 0.2$ and $0.1N$. On each graph the initial and final orbits are drawn, in addition to the dots corresponding to each integration step. The spacing between the dots thus indicates the integration stepsize (we recall that the integration variable is here always the longitude L, even if it is sometimes abusively labeled as "time" on some graphs).

RKF45 integrator

We begin with the results of the RKF45 integrator.



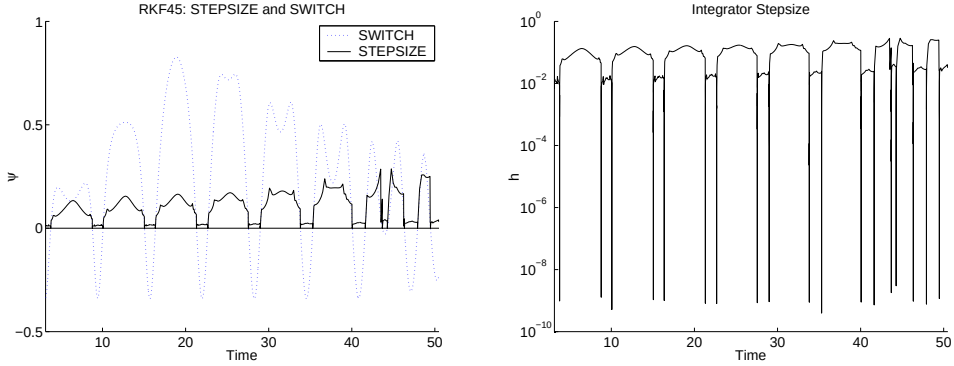
Trajectories for 10,5,1,0.5,0.2 and 0.1N - RKF45 integrator

The first thing we can note about these trajectories is another indication of the extreme regularity of the problem with respect to the maximal thrust T_{max}. Trajectories for all thrusts share the same pattern for the integration nodes, only with more revolutions for lower thrusts. Concerning the spreading of the nodes, they seem here quite regularly spaced along the orbit, but clearly with a smaller step over thrust arcs (centered on apogees and later perigees) than non-thrust arcs.

Now let us have a closer look at the evolution of the stepsize during the integration, and more precisely along with the switching function ψ. We recall that the switching function ψ and the optimal control u^* are closely related, namely:

- $|u^*(t)| = 1$ when $\psi(t) < 0$, t is then within a thrust arc.
- $u^*(t) = 0$ when $\psi(t) > 0$, t is then within a non-thrust arc.
- u^* is discontinuous at t when $\psi(t) = 0$, t is then a commutation, or switching time.

Note: the following graphs showing the stepsize correspond to the 10N transfer. The qualitative behaviour for lower thrusts is similar, but the graphs provide no visual interest due to the huge number of commutations. The first graph shows the switching function ψ and the integration stepsize h, the second graph shows only the stepsize, with a logarithmic scale for a better vision of the three ranges of values taken (corresponding to the thrust, non-thrust and commutation cases).



Integration stepsize and switching function (10N) - RKF45 integrator

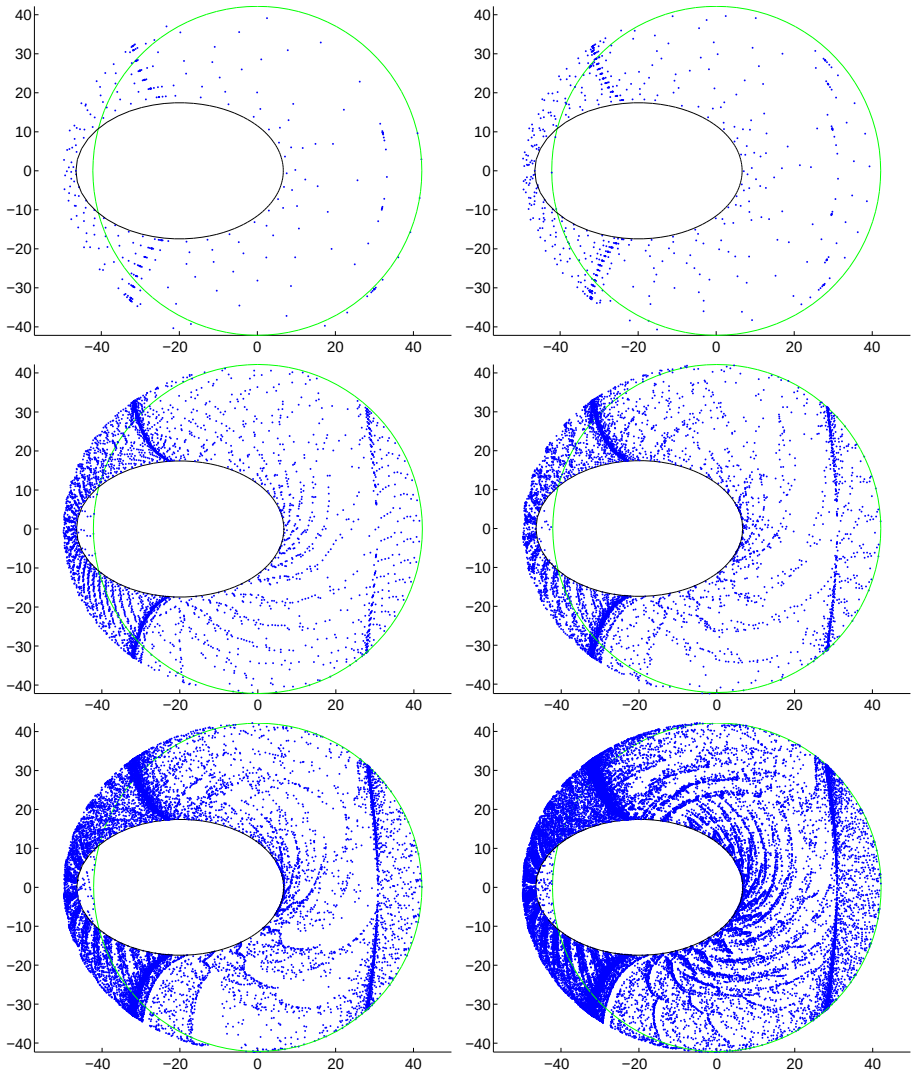
The logarithmic scale on the right graph gives a better idea of the order of magnitude of the stepsize, and we can distinguish three ranges, that interestingly coincide very well with the sign of the switching value ψ. First, we can see that the integration stepsize becomes very small (as low as 10^{-9}) at the commutations, when the switching function ψ vanishes. This indicates that the integrator correctly detects the right hand side discontinuities induced by the control switching at the commutations.

Over the thrust arcs, the stepsize is stable and relatively low, around 10^{-2}, and slightly increasing from one revolution to the next one. The difference in stepsizes between the first and last thrust arcs of the transfer is quite clear on the left part of the graphs for T_{max} below $1N$: the final thrust arcs overlap the initial ones and show more spaced dots, like for the thrust arcs on the right part of the graphs. Over the non-thrust arcs, the step is nearly one order of magnitude larger, around 10^{-1}, and is rather regular too. This difference of range probably finds its explanation in the dynamics of the system. With the state expressed in terms of orbital parameters, a null control just makes the satellite move along on its orbit, the orbit itself staying the same. Then in the state $[P, e_x, e_y, h_x, h_y, L, m]$ only the longitude

varies, whereas all other orbital parameters and the mass remain constant. Therefore it seems logical that the integration can be performed with a larger step over the non-thrust arcs. Concerning the moderate increase of the step over the successive thrusts arcs, it may be related to the change of form of the orbit, that tends to a circular one, with a lower angular speed of the satellite than for the early apogees, which would allow a faster integration along the longitude.

DOP853 integrator

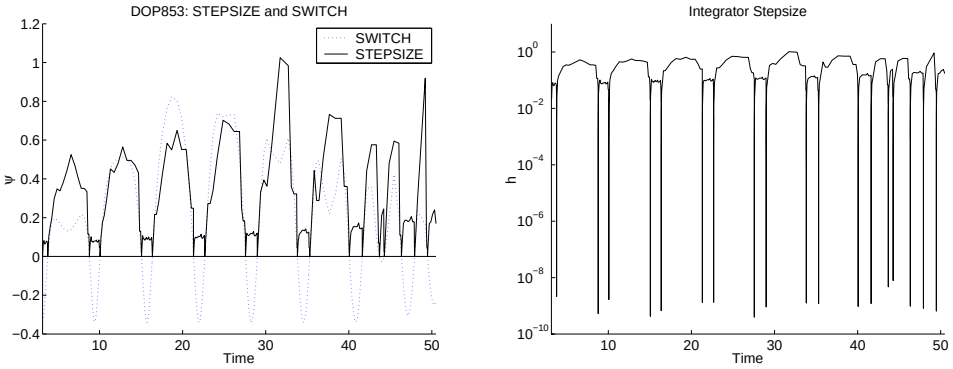
We show now the trajectories for the DOP853 integrator, which belongs to the same class of methods (embedded Runge Kutta) as RKF45, but is of higher order, 8 instead of 5.



Trajectories for 10,5,1,0.5,0.2 and 0.1N - DOP853 integrator

As before, we can see that these trajectories still look similar for all thrusts. The similarities with RKF45 are the regular spreading of the integration nodes over both thrust and non-thrust arcs, still with smaller steps over the former. We also notice the same difference of stepsize between the early apogees thrust arcs and the later apogees or perigees thrust arcs. Overall, the fact that DOP853 takes only about half as many nodes as RKF45 seems to correspond to a globally larger stepsize, over both kinds of arcs. Another first glance difference is the presence of a huge concentration of nodes around the commutations between the thrust and non-thrust arcs, especially visible on the $1N$ and $0.5N$ graphs for instance. There were of course also very small steps around the commutations for RKF45, but there was not such a contrast with the thrust arcs.

Now let us move on to the detailed view of the stepsize.



Integration stepsize and switching function (10N) - DOP853 integrator

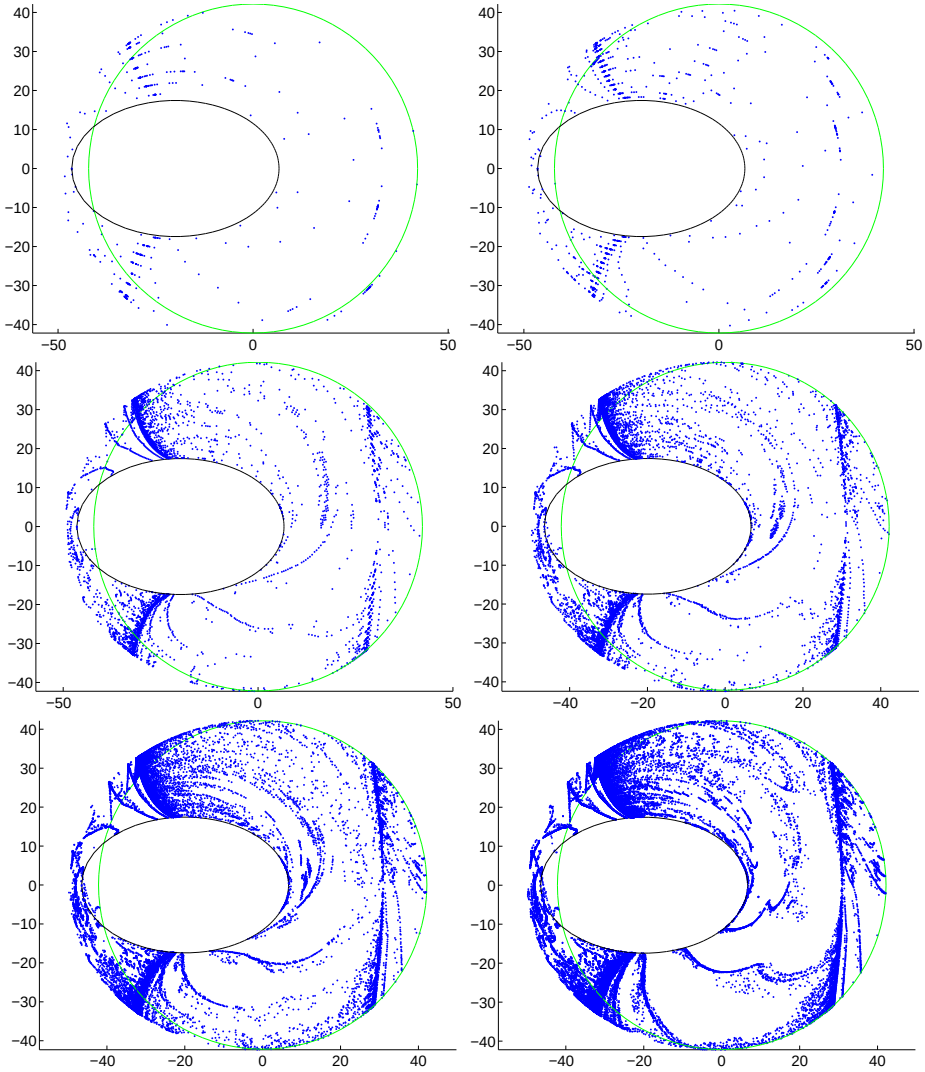
This confirms the important similarities of behaviour between DOP853 and RKF45, as we find again three distinct ranges of values for the stepsize, that perfectly match the sign of the switching function. First, same as before, the stepsize becomes extremely small (around 10^{-9} again) around zeros of the switching function, as the integrator has to deal with the right hand side discontinuities due to the commutations. Then concerning the thrust arcs, the stepsize is still regular and rather small, with values roughly around 10^{-1}. Outside of the thrust arcs, we still have larger stepsizes, between 0.1 and 1 roughly.

The only noticeable differences with RKF45 seem to be the overall larger stepsize (except at the commutations, which may explain the visual impression of dots accumulation), and the slightly closer stepsizes over the thrust / non-thrust arcs at the end of the transfer. The larger steps are most probably a consequence of the higher order of DOP853, while the last point could indicate a slightly better performance of the integrator when the orbit becomes more circular. All in all, we have the same qualitative behaviour

between the to integrators, with a quantitative advantage to DOP853 in terms of overall stepsize. This is quite coherent with them both implementing embedded Runge-Kutta methods, but of higher order for DOP853.

ODEX integrator

We finish with the ODEX integrator, which unlike the two others uses an extrapolation method, with both variable order and stepsize.

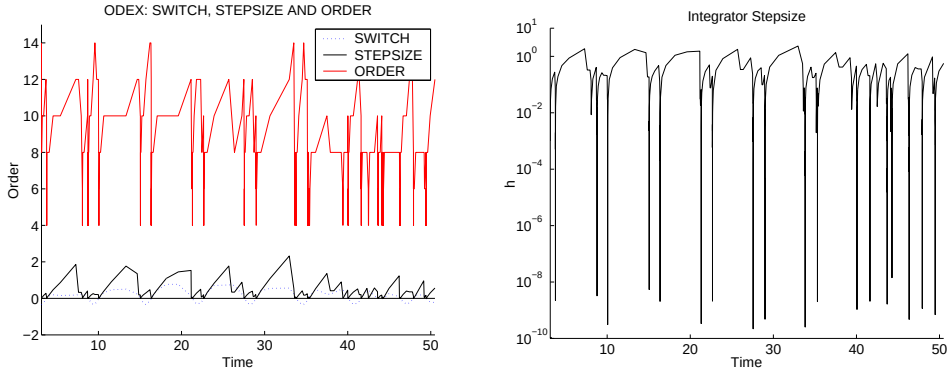


Trajectories for 10,5,1,0.5,0.2 and 0.1N - ODEX integrator

Compared to the two previous integrators, there is an obvious difference in the pattern of the integration nodes for ODEX. Contrary to what it was for RKF45 and DOP853, their spreading appears quite irregular, with some noticeably large "blanks" occurring not only on non-thrusts arcs, but

also on thrusts arcs. These indicate very large integration steps, which are indeed a trait of extrapolation methods. What is a bit surprising is the seemingly lesser distinction between thrust and non-thrust arcs in terms of stepsize, especially visible on the early apogees (on the left of the graphs). Still, a common point with the previous integrators is the concentration of nodes around the commutations, and more precisely *before* the commutations. This is slightly different from what we have seen before with DOP853 for instance, where the accumulation of nodes seems rather centered on the commutations. Here we rather have increasingly small steps before the commutations, and relatively large steps just after the commutations.

As with ODEX the order can also change at each step, we have represented it along with the stepsize and the switching function.



Integration order, stepsize and switching function (10N) - ODEX integrator

Once again, we can see that the stepsize drops at the commutations to very small values (10^{-9} and below). An interesting point is that the integration order similarly drops there to its minimal value (4), which is supposed to allow an efficient handling of the discontinuities, as pointed out in [27], II.9 page 237. Otherwise, the steps are still globally larger over the non-thrusts arcs than thrust arcs, but are in both cases less regular than before. In particular we can see that the stepsize increase after a commutation is less steep than with RKF45 and DOP853, for which we have an almost symmetric evolution of the stepsize on each arc. This may be related to the fact that the step control mechanisms are different for ODEX, which could also explain the fading of the stepsize gap between the thrust and non-thrust arcs near the end of the transfer.

As expected from the trajectories, the overall stepsize values are higher than with DOP853, between 10^{-1} and 1 over thrust arcs and above 1 over non-thrust arcs. Despite this ODEX curiously takes more steps than DOP853, halfway between RKF45 and DOP853 actually. Maybe this is due

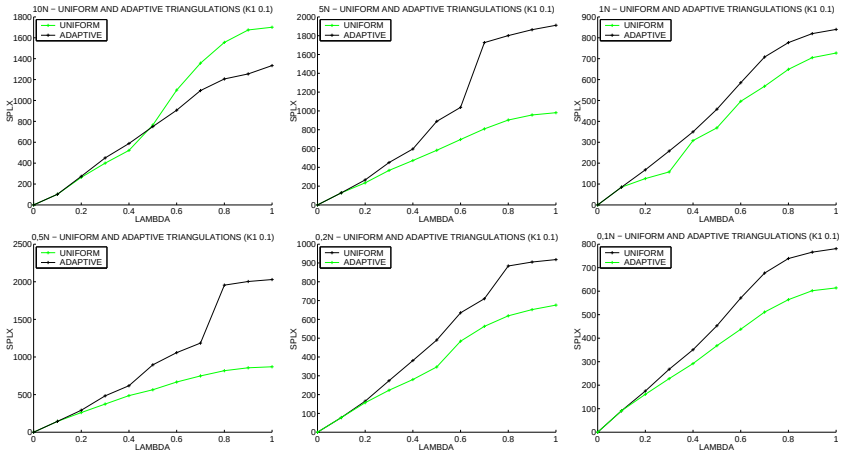
to some difficulties to cross the commutations, as we notice an accumulation of integration nodes before the switching times, which is not the case for the two other integrators. The order is comprised between 4 (minimal value, attained at the switchings) and 14, which is higher than for the two previous integrators. At this point, it seems that the extrapolation method might not be a good choice for this problem, as its high order results in longer integration times, with the number of steps still larger than for DOP853.

3.4 Adaptive triangulation study

We wish here to experiment on these problems with the adaptive meshsize described in 2.3. We test here the default configuration for the full adaptive mode, namely a deviation tolerance set to 10^{-1} and $(2, 0.2, 0.1)$ for the anisotropic deformation thresholds. First we compare the path followings with and without the adaptive mode, in order to study the impact on the number of simplices followed. Then we look at the final shoots at the end of the path, to see how they are affected by the precision of the solution at $\lambda = 1$.

Path following

We represent here the number of simplices followed along the path. It is obviously always increasing, starting from 0 to the total number of simplices at $\lambda = 1$. We plot together the simplices for the uniform (reference) and adaptive meshsize, for $T_{max} = 10, 5, 1, 0.5, 0.2$ and $0.1N$.
Note: the maximum number of simplices allowed for the junctions is 100, except 500 for the $0.2N$ *case that presents junctions difficulties (see later)*



Simplices followed for Uniform and Adaptive meshsize 10,5,1,0.5,0.2 and 0.1N - $K_1(10^{-1})$ triangulation

First we observe a curious "gap" of simplices between the levels $\lambda = 0.6, 0.7$ or $\lambda = 0.7, 0.8$ on some paths, namely the $5, 0.5$ and $0.2N$ (we study this phenomenon a bit later). Concerning the gain of the adaptive mode in terms of simplices, it does not seem that important, except in the two cases where these gaps are avoided (5 and $0.5N$). In other cases, the gain exists but is moderate (with the exception of the $10N$ case, where the adaptive actually takes more simplices!). Concerning this last remark, more generally, we have less to expect from the adaptive meshsize when a large uniform meshsize such as $K_1(10^{-1})$ already converges well, as there is not much left to improve. Now that we have an idea of the simplices gain, we examine the precision improvement, and its consequences on the final shoot.

Final Shoot

We look here more closely at the shoot results after the uniform and adaptive path followings. All shoots use the standard RKF45 with the tolerances $(10^{-12}, 10^{-10})$ (the results for the uniform $K_1(10^{-1})$ thus coincide with the ones presented at the beginning of this chapter). We indicate in the following tables
- $|H_1|$: the norm of the homotopy at the end of the path, **before** the shoot (this roughly measures the quality of the initial point for the shoot).
- **Shoot (H)** and **Final norm**: same as before, the number of homotopy calls and final norm for the shoot.
- **Total (H)** and **Total (s)**: the total number of homotopy calls, and execution time (in seconds).
Note: it should be noted that the homotopy calls made during the shooting attempts are most costly than the calls during the path following, as we require a more precise integration. Therefore, the total execution time is not exactly proportional to the total number of homotopy evaluations (for a given T_{max}), even though the computational cost essentially comes from these calls. The difference is not that important, however, due to the rather low number of "shoot" calls compared to the "path" calls...

T_{max}	$\|H_1\|$	Shoot (H)	Final norm	Total (H)	Total (s)
10N	0.24	55	$3.43\ 10^{-10}$	1553	20
5N	0.76	87	$4.63\ 10^{-10}$	2117	51
1N	0.05	51	$8.02\ 10^{-11}$	943	132
0.5N	0.74	74	$1.12\ 10^{-9}$	2226	548
0.2N	0.24	49	$3.21\ 10^{-9}$	1050	661
0.1N	0.04	49	$1.75\ 10^{-6}$	920	1252

$K_1(10^{-1})$ *triangulation - Uniform (reference) mode*

We first notice that the shoot seems to require more homotopy evaluations when the norm of the solution at the end of the path is higher (5 and 0.5N). Now are the results corresponding to the adaptive meshsize.

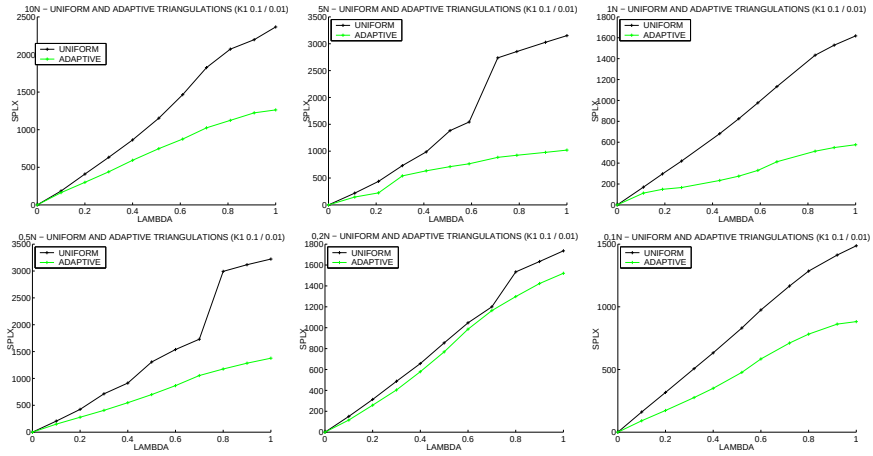
T_{max}	$\|h_1\|$	Shoot (H)	Final norm	Total (H)	Total (s)
10N	0.01	43	$1.93\ 10^{-10}$	2252	27
5N	0.002	66	$2.07\ 10^{-10}$	1425	35
1N	0.02	64	$4.11\ 10^{-11}$	1226	166
0.5N	0.01	44	$4.89\ 10^{-10}$	1277	305
0.2N	0.02	44	$2.16\ 10^{-7}$	1116	696
0.1N	0.02	70	$9.85\ 10^{-5}$	1018	1486

$K_1(10^{-1})$ *triangulation - Adaptive mode*

We observe that the norms at $\lambda = 1$ before the shoot are better than with the uniform meshsize. However, the improvement on the shoot is not very clear: it is better for the two previous worst cases again (5 and $0.5N$), but remains the same or even gets worse (1 and $0.1N$) otherwise. Overall, the poor results in terms of performance (global homotopy calls / execution time) indicate that the adaptive meshsize mechanism does not perform well with the $K_1(10^{-1})$ triangulation.

3.4.1 A better suited meshsize

The first experiments, made with the $K_1(10^{-1})$ used previously, confirm that the adaptive meshsize actually requires a smaller meshsize with respect to λ. Indeed, a value of 0.1 means about 10 adaptive junctions only along the path, which is too rough to benefit fully from the adaptive mode. This is why we also try a slightly modified meshsize, namely $\delta_i = 0.1, \forall i \in [1..n]; \delta_{n+1} = 0.01$, which we denote by $K_1(10^{-1}/10^{-2})$ in the following. This second triangulation, although leading to a slower path following than the basic $K_1(10^{-1})$, allows us to better study the effects of the adaptive meshsize. So we repeat now the same tests with a stepsize of 10^{-2} for λ.



Simplices followed for Uniform and Adaptive meshsize 10,5,1,0.5,0.2 and 0.1N - $K_1(10^{-1}/10^{-2})$ triangulation

An interesting thing is that we notice the same gaps as before for the $5, 0.5$ and $0.2N$ paths. Moreover, the adaptive meshsize still performs very well

for $5N$ and $0.5N$, while it has very little effect for $0.2N$ (we study these two cases more in details in the following). For other paths the gain is also more important than before, which confirms that the stepsize of 0.1 for λ is limiting the adaptive process.

Now let us have a look at the shoot results with the $K_1(10^{-1}/10^{-2})$.

T_{max}	$\|h_1\|$	Shoot (H)	Final norm	Total (H)	Total (s)
10N	0.24	80	$1.30\ 10^{-10}$	2685	34
5N	0.76	71	$4.11\ 10^{-10}$	3425	77
1N	0.05	36	$1.53\ 10^{-10}$	1793	223
0.5N	0.74	95	$1.55\ 10^{-9}$	3546	855
0.2N	0.24	83	$6.33\ 10^{-9}$	2030	1293
0.1N	0.04	58	$3.49\ 10^{-5}$	1726	2170

$K_1(10^{-1}/10^{-2})$ *triangulation - Uniform (reference) mode*

Same as before, we observe a rough link between the norm of the solution at the end of the path and the number of homotopy calls made during the shoot, with faster shoots for 1 and 0.1N. Here are the corresponding adaptive meshsize results.

T_{max}	$\|h_1\|$	Shoot (H)	Final norm	Total (H)	Total (s)
10N	0.01	48	$2.28\ 10^{-10}$	3736	49
5N	0.01	43	$1.78\ 10^{-10}$	2677	72
1N	0.02	33	$4.70\ 10^{-9}$	1742	209
0.5N	0.03	34	$9.78\ 10^{-10}$	2090	489
0.2N	0.03	49	$2.10\ 10^{-9}$	2552	1510
0.1N	0.02	46	$6.77\ 10^{-5}$	1772	2051

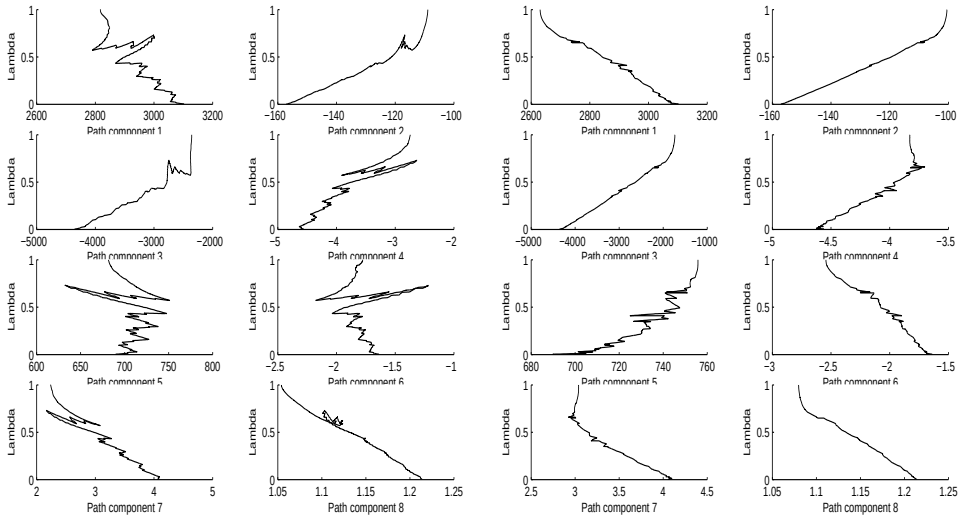
$K_1(10^{-1}/10^{-2})$ *triangulation - Adaptive mode*

This time things are more satisfying, with a clear improvement on the norms and a matching improvement on the shooting attempts (except for the already fast 1 and 0.1N cases). Now all shooting attempts take between 30 and 50 homotopy evaluations. In addition to the better gain in terms of simplices, as seen before, this confirms the fact that the adaptive meshsize requires a not too high stepsize with respect to λ in order to be effective. However, despite the gain in simplices for the main path and the faster shoot, the overall performance is a bit disappointing, as the cost of the junctions often offsets these improvements.

We focus now a little bit more on the qualitative aspect of the following, and have a closer look at two of these path followings, namely the 0.5 and $0.2N$ cases for the $K_1(10^{-1}/10^{-2})$ triangulation. The $0.5N$ path following shows an example of the gaps mentioned before, and we observe an important gain with the adaptive meshsize. Conversely, for $0.2N$ we also observe a (smaller) gap between $\lambda = 0.7$ and $\lambda = 0.8$, but the adaptive gain is here insignificant, and the junctions in this case are difficult (we have increased the maximum number of junctions simplices to 500).

A favorable case

Let us begin with the favorable case ($0.5N$).



Path for 0.5N - Uniform and Adaptive meshsize

We see that the path on the left, corresponding to the uniform meshsize, is quite irregular, with strong oscillations on components 1, 5 and 6 in particular. By looking more closely at what happens around $\lambda = 0.7$, we have the explanation of the simplices gap. We observe that the following clearly turns back after reaching $\lambda \approx 0.7$, going down to $\lambda \approx 0.6$ again, before resuming the progression until $\lambda = 1$. This of course explains why it seems to take so many simplices between the levels $\lambda = 0.7$ and $\lambda = 0.8$. Considering the general allure of the path before and after this event, this behaviour may indicate the existence of two distinct paths (or maybe a bifurcation).

On the other hand, the path on the right, corresponding to the adaptive meshsize, looks much smoother. The oscillations on components 1,5,6 are strongly reduced, and more importantly, there is no sign of turning back around $\lambda = 0.7$ (still, an inflexion is clearly visible on components 4,7 for instance). This better handling shows in the significant reduction in main simplices (1377 versus 3222). The adaptive junctions require 1286 simplices, which is nearly as much as the main path (we recall, however, that all junction simplices do not requires the evaluation of the homotopy, see later).

We have now a closer look at the evolution of the adaptive meshsize along the path. We represent only δ for λ multiple of 0.1 for clarity, even if the actual level height is $\delta_{n+1} = 0.01$ (and remains unchanged during the following).

λ	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8
0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.4	0.1	0.4	0.1	0.2	0.05
0.2-0.3	0.1	0.1	0.4	0.1	0.4	0.1	0.2	0.025
0.4	0.1	0.1	0.4	0.2	0.4	0.2	0.1	0.025
0.5-0.6	0.1	0.1	0.4	0.1	0.4	0.2	0.1	0.025
0.7-1	0.05	0.05	0.2	0.1	0.2	0.1	0.05	0.0125

Meshsize δ evolution along the path - 0.5N ($\delta_{n+1} = 0.01$)

We immediately notice the meshsize global reduction occurring between $\lambda = 0.6$ and $\lambda = 0.7$ (at $\lambda = 0.65$ more precisely), that would coincide with the deviation and turning back observed in the uniform case. The algorithm seems to correctly detect some difficulties there (in practice the homotopy norm crosses the threshold value of 0.1) and switches to a smaller meshsize for a more cautious following. This turns out to be a correct move, as we avoid the bad behaviour of the uniform meshsize. Besides, this small meshsize until the end is coherent with the nearly "vertical" aspect of the path for $\lambda \in [0.7, 1]$.

Concerning the different sizes for each dimension in general, we observe that the third and fifth components tend to have a larger meshsize, whereas the eighth component has a smaller one. This is coherent with the fact that component 3 is the one having the largest relative variation (more than a factor 3 between the initial and final value), while component 8 has a small variation (from 1.21 to 1.08 roughly). The case of component 5 is less clear, as even with the scaling, the variation does not seem that important. However, we also see on the graphs that this component is one of the most irregular on the uniform path, and the less smooth on the adaptive path. Moreover, it happens that the difficulties of the $0.2N$ case also seem related to this particular component (see just below).

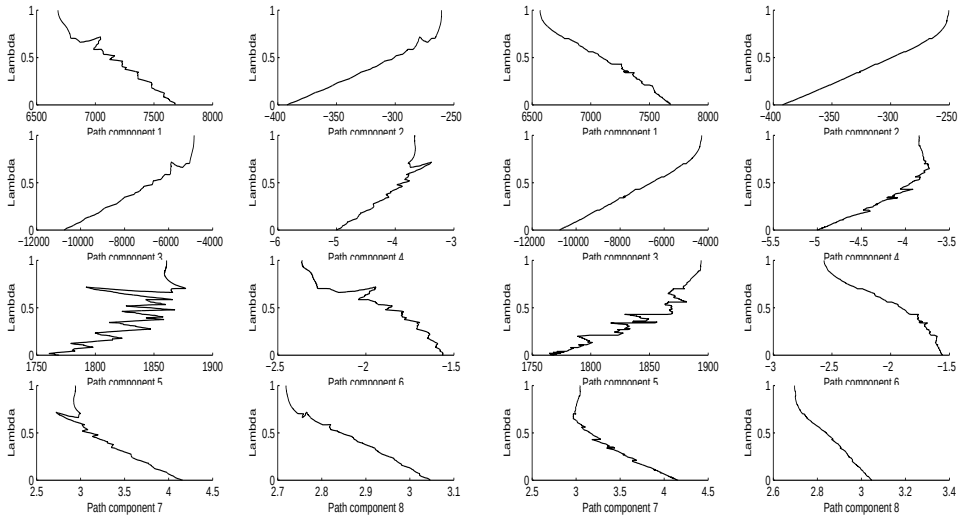
To finish with, the additional benefit of the adaptive meshsize, namely a faster final shoot, is also significant here. The norm at the end of the path (before the shoot) is of 0.03 instead of 0.74, and the shoot takes only 34 homotopy evaluations (versus 95 for the uniform meshsize). This is not negligible, all the more so as the homotopy evaluations for the shoot are most costly than for the path (the integration is done with lower tolerances).

These two factors lead in this case to an overall important gain, with a total execution time of 489s versus 855s (2090 and 3546 homotopy calls respectively).

An unfavorable case

Now we would like to understand what happens in the $0.2N$ case, for which the adaptive mode leads to a longer execution time than the reference. Con-

cerning the final shoot, there is indeed an improvement, with 49 homotopy calls versus 83 (the norm of H at the end of the path being of 0.03 instead of 0.24). The problem lies in the number of simplices followed on the main path, which is almost the same than with the uniform meshsize. This is probably related to the difficulties observed with the adaptive junctions, for which we have to increase the maximal number of simplices from 100 to 500. Otherwise junctions keep failing, and we basically keep the same meshsize all along. Yet even with these settings, the simplices gain is very limited compared to the cost of the junctions. We look at both paths with uniform and adaptive meshsize, in order to try to figure out these difficulties.



Path for 0.2N - Uniform and Adaptive meshsize

First we observe that in the adaptive case, the path looks smoother, like in the $0.5N$ case. However, if we look more closely at the component 5 of z, we notice some horizontal "jumps" corresponding to the junctions that take a lot of simplices to complete. This illustrates the phenomenon described in 2.3.1, when a junction causes the path following to jump from a zero path branch to another. It can sometimes then turn back to $\lambda = 0$, or like here just continue towards $\lambda = 1$. The problem is that this is not at all the purpose of an adaptive junction !

Indeed, we are supposed to have computed a new meshsize better suited to the path we are following. If we jump to another branch instead of continuing on this path, nothing guarantees that the new meshsize is suited to the new branch, which completely ruins the adaptive mechanism. Moreover, we run the risk of a precision loss of the following at these straying jumps, when the junction manages to catch up with the next branch. Of course, the consequences of this phenomenon are aggravated if this happens repeatedly. In this case, in addition to the possibility of repeated precision losses

at the jumps, we end up most of the time with an unadapted meshsize, as it was designed according to the progression *before* the jump. In such circumstances, maybe a lesser evil is indeed just to give up with the adaptive mode and keep the uniform meshsize.

We seem here to be in this situation, which would explain the difficulties of the adaptive mode. On this particular example, we observe that strictly limiting the junction simplices (to 50) does not give good results: we have a lot of junction failures after 50 simplices, with an overall much longer execution time than the reference. Trying a more aggressive adaptive setting, with a deviation tolerance of 1 for instance, turns out to be too unstable, while a more cautious setting (10^{-2}) does not give better results either.

Remark: incidentally, if we look at the evolution of the meshsize along the path, like before, we notice that the size for the component 5 becomes very small, below 10^{-2}*. This is probably one of the practical reasons of the poor adaptive performance in this case, and confirms that the handling of this unknown (namely* $p_{h_y}(0)$*) seems to be quite sensitive.*

First conclusions on the adaptive meshsize

Overall, these tests show a satisfying qualitative behaviour of the adaptive meshsize, in two distinct ways. First, we notice in most cases a significant reduction in the number of simplices followed for the main path, with a generally smoother following. The gain is quite impressive in some favorable cases, when we can avoid some irregularities in the path following. Besides, we also observe a general improvement of the precision of the following, which leads to faster final shooting attempts, with less homotopy evaluations. However, if the junctions are generally fast (less than 50 simplices), there are also some problematic situations, maybe related to the existence of several close branches of the path. In this case the junctions actually cause more harm than good, and it seems that the whole adaptive mode should be disabled.

Nevertheless, the comparison of the global execution times highlights the fact that the numerical cost of the adaptive junctions is here not negligible at all. Indeed, this cost often offsets the two gains mentioned above, leading to a limited overall performance improvement. Generally speaking, there is less to gain to expect for the path followings that already go smoothly with a large uniform meshsize, whereas we can expect a good improvement for the more irregular ones (except in the problematic case mentioned above).

Remark: these experiments confirm that a stepsize of 10^{-2} *for* λ *is better suited to the adaptive mode than* 10^{-1}*, which is too rough.*

On a side note, using the solution refining process described in 2.4 does work on these problems, but suffers from the same flaw. The speed gain on the final shoot (thanks to a better starting solution at $\lambda = 1$ after the refining) is lower than the cost of the refinement junctions, and we actually observe a longer global execution time.

3.4.2 Junction homotopy formulation

To finish with, we try to compare on this family of orbital transfer problems the different formulations for junctions homotopies described in 2.2. In particular, we would like to emphasize that both the "first" and "second" formulations (that use an approximation of the Jacobian of H by finite differences) cannot perform well without some numerical tuning, which is a strong limitation compared to the "third" (derivatives-free) formulation. The test suite consists in the raw path following (no final shoot), with a full adaptive mode. We keep the default values for the deviation tolerance and anisotropic thresholds, ie 0.1 and $(2, 0.2, 0.1)$. As we expect more difficult junctions with the first and second formulations, and to keep a coherent test environment, the maximum number of simplices allowed for junctions has been increased to 500 for all tests. Also, we choose the $K_1(10^{-1}/10{-}2)$ instead of the $K_1(10^{-1})$ because it implies more junctions and is better suited to the adaptive mode, as said before.

It should be noted that if the number of junction simplices naturally differs for each formulation, the number of simplices for the main homotopy path is also often different. This is due to the fact that the junctions may come up with different completely labeled faces, or can sometimes fail for a particular formulation (contrary to a first junction failure, an adaptive junction failure does not stop the path following in any case: we just keep the previous meshsize and go on).

First formulation

We being with the first formulation, in which we use centered finite differences, with an uniform stepsize h. We recall that the numerical integration along the path is performed by RKF45, with absolute and relative tolerances of 10^{-8} and 10^{-6}.

A uniform stepsize of 10^{-6} gives awful results, attaining convergence only in 2 cases out of 6, with thrice the usual homotopy calls and time, while the first junction fails for other values of T_{max}. Obviously, we must chose a larger step for the finite differences. Indeed, a stepsize of 10^{-3} or 10^{-1} manages much better, as indicated below.

The following tables indicate for each path following:
(i) **Path** the number of simplices for the main homotopy
(ii) **Junc.** the total number of simplices used for junction homotopies
(iii) **H calls** the total number of homotopy evaluations
(iv) **Time** the time (in seconds) taken by the path following

T_{max}	Path	Junc.	H calls	Time
10N	1218	2148	4233	44
5N	913	1940	2901	63
1N	605	1546	1738	216
0.5N	1079	7971	4524	887
0.2N	1140	3027	3253	1745
0.1N	-	-	-	-

T_{max}	Path	Junc.	H calls	Time
10N	1218	2398	4451	48
5N	1143	7185	4201	92
1N	773	662	1818	190
0.5N	1316	3725	3908	830
0.2N	1397	871	2793	1411
0.1N	904	425	1800	1794

Junctions for the first formulation, with $h = 10^{-3}$ *and* $h = 10^{-1}$

Then we move on to the modified formulation, in which the stepsize is proportional to the triangulation meshsize ($h_i = \frac{\delta_i}{k}, \quad i = 1, \ldots, n$).

With $k = 1000$ or $k = 100$ we obtain poor results, failing to complete the first junction for half the tests, and up to twice the usual homotopy calls. Then with k set to 10 or 1, it is much better, with results comparable to the reference second formulation.

T_{max}	Path	Junc.	H calls	Time
10N	1218	2134	4236	47
5N	913	1928	2904	63
1N	547	466	1418	171
0.5N	1019	1252	2591	578
0.2N	1395	707	2776	1594
0.1N	1450	736	2810	3198

T_{max}	Path	Junc.	H calls	Time
10N	1218	2490	4464	48
5N	761	2316	2711	56
1N	559	567	1476	154
0.5N	1260	2243	3210	629
0.2N	1404	670	2696	1314
0.1N	846	586	1906	1892

Junctions for the first formulation, with $k = 10$ *and* $k = 1$

Second (reference) formulation

Here are now the reference results for the second formulation, which does not require the derivatives of the homotopy. This point was important for the problems we study, as we want to be able to handle homotopies with a low regularity. An additional benefit is that there is no numerical parameter to set, and this formulation is naturally the default mode for the Simplicial code.

T_{max}	Path	Junc.	H calls	Time
10N	1383	3785	4188	45
5N	792	4129	2614	60
1N	586	447	1339	161
0.5N	1044	1069	2386	493
0.2N	1500	371	2421	1422
0.1N	1527	505	2545	2812

Junctions for the second formulation

We compare now the results of th first ($h = 10^{-3}, 10^{-1}$ and $k = 10, 1$) formulations with the second one, in terms of homotopy calls and total path time:

- **ratio (H)** stands for $\frac{\text{H calls for first formulation}}{\text{H calls for second formulation}}$

- **ratio (s)** stands for $\frac{\text{Time for first formulation}}{\text{Time for second formulation}}$

T_{max}	(H)	(s)	(H)	(s)
10N	1.01	0.98	1.06	1.07
5N	1.11	1.05	1.61	1.53
1N	1.30	1.34	1.36	1.18
0.5N	1.90	1.80	1.64	1.68
0.2N	1.34	1.23	1.15	0.99
0.1N	-	-	0.71	0.64

T_{max}	(H)	(s)	(H)	(s)
10N	1.01	1.04	1.06	1.07
5N	1.11	1.05	1.04	0.93
1N	1.06	1.06	1.10	0.96
0.5N	1.09	1.17	1.35	1.28
0.2N	1.15	1.12	1.11	0.92
0.1N	1.10	1.14	0.75	0.67

Comparison: first ($h = 10^{-3}, 10^{-1}$ and $k = 10, 1$) / second formulation

We observe that for the formulations ($h = 10^{-3}, k = 10$), for each T_{max} value, both ratios are quite close, which is coherent with the fact that the computational cost is mostly due to the homotopy calls. The measure of the execution time, however, is less reliable (for material reasons, and it depends on the harware as well), so the number of homotopy calls is in practice a handier performance index.

On the contrary, comparing the path simplices is not sufficient to judge the relative merits of the formulations, as can be seen for $T_{max} = 0.2N$, where the first tests count less simplices, despite a larger number of homotopy calls and longer execution time. This is due to the fact that junction homotopies also use homotopy calls, so a path following that spends lots of efforts in performing the junctions can be inferior to another one with more simplices but cheaper junctions.

However, taking also into account the simplices used for the junctions is not the answer, as the results for $10N$ and $5N$ clearly show: the third formulation counts nearly twice as much junction simplices, and about as much path simplices, for a lower total number of homotopy calls ! This actually comes from the nature of homotopy junctions that use a meshsize of 1 with respect to their homotopic parameter λ' (see 2.2). The vertices with $\lambda' = 0$ do not require an evaluation of H, only some matrix-vector operations, which in our case has a much lower cost. This is why just counting the total number - path and junctions - of simplices gives a flawed comparison of the different formulations.

So, odd as it may seem, the number of simplices taken by the junctions does not actually provide a good measure of their efficiency, only the global

homotopy count does. These tests based on the orbital transfer problems show that it is possible to find settings that yield acceptable performance for the first and (especially) the second formulation. However, they also reveal that poor values can lead to catastrophic results, first junction failures in particular. This is problematic as the correct settings are probably problem-dependant, or even vary with the integration method and precision chosen for a specific problem. Thus finding the appropriate values would likely require a sequence of trial and error tests on each particular case, which is of course quite a burden. Moreover, it is not even granted that such settings exist if the shooting function is not differentiable. In that aspect the third formulation is clearly superior to the two previous ones, as no derivatives are used. These tests also show that it generally gives better performance than the two others, even if there exist some lucky parameters values that could give better results for the latter.

Conclusion

With the continuation on the objective and the initialization provided by a fixed step shoot, the simplicial method allows us to solve the problem with a maximal thrust as low as $0.1N$. The solutions obtained coincide with the ones found by the differential continuation approach used in [26], but the execution times are up to three times longer, which was predictable.

Then, we make some comparisons between three variable steps integrators (RKF45, DOP853 and ODEX) that confirm the extreme regularity of this family of problems with respect to the maximal thrust. Concerning the handling of the bang-bang structure and huge number of commutations, it would seem that RKF45 has the best speed/precision combination. Of course, this is true for the particular settings we used, and there may be specific configurations more suited to the two other integrators (of higher order).

The experiments on the adaptive meshsize show a satisfying qualitative behaviour, with a smoother path following. In favorable cases this leads to a dual gain, with both a reduction of the number of simplices followed and a faster final shoot. However, we also observe that the numerical cost of the adaptive junctions is far from being negligible. Indeed, if the uniform meshsize already allows a clean path following, then we may end up with an insignificant overall performance gain. Another problematic case seems to be the presence of close but distinct branches of the zero path, which can perturb the adaptive junctions to the point of making the adaptive mode useless.

This suggests that the best way to further improve the adaptive mesh-

size is probably to find a cheaper formulation for the adaptive junctions, or possibly to find another way of performing the meshsize changes. To finish with, these experiments also indicate the superiority of the derivatives-free formulation for the junctions. Indeed, the others formulations (that approximate the Jacobian of the homotopy by finite differences) appear extremely sensitive to the numerical setting of the stepsize, and are probably less suited to problems with a low regularity anyway.

Chapter 4

Singular Arcs problems

Among optimal control problems, singular arcs problems are interesting and difficult to solve with indirect methods, as they involve a multi-valued control and differential inclusions. In short, singular arcs occur in indirect methods when the necessary conditions from the Hamiltonian minimization fails to determine uniquely the optimal control over a non-trivial interval. In this case, applying the Pontryagin's Maximum Principle leads to a Boundary Value Problem with a differential inclusion. We keep the usual notations: x the state, p the costate, u the control, $y = (x, p)$ and φ the state-costate dynamics. Then if Γ denotes the set valued map of optimal controls, then one has

$$(BVP) \begin{cases} \dot{y} \in \Phi(y) = \varphi(y, \Gamma(y)) \quad ae\ in\ [t_0, t_f] \\ Boundary\ Conditions \end{cases}$$

Multiple shooting is an efficient way to solve this kind of problems, but typically requires some a priori assumptions on the control structure (number and approximate location of singular arcs in particular). Another possibility, described for instance in [6], is to use a discretization of the (BVP) with some interior penalty techniques (which can be seen as a kind of discrete continuation).

So the primary objective of the continuation is here to provide a reliable singular structure detection and initialization, so that we can then use a method derived from multiple shooting to solve the problems precisely. The continuation is based on regularizing the problems by adding a quadratic (u^2) perturbation to the criterion (we limit here ourselves to the case where the Hamiltonian is linear with respect to the control).

Remark: problems with singular arcs sometimes present a particular difficulty called "chattering", ie an infinite concatenation of bang and/or singular arcs occurring in a finite time. However, we have not encountered this case in the experiments on the two problems studied here.

Note: we introduce in this chapter several variants of the single or multiple shooting methods, some of them involving the state-costate or control discretization. We keep for these the term of "shooting function", which may seem kind of improper for the discretized ones. However, all these formulations still use the costate p, and take into account the necessary conditions on the optimal control. So it seems to me that it is appropriate for them to keep a name associated to the indirect methods class.

4.1 Problems statement

We study in this chapter two singular arcs problems in parallel: an optimal harvesting model in fishery from Clark ([20]), and a quadratic regulator problem studied for instance by Chen and Huang in [19]. We do this to observe the similarities and/or differences of behaviour with respect to the method, so that we can better tell what is related to the method or to a particular problem. This way, we also hope that the method be not too problem dependant, as we try to apply it to both problems in the same conditions. This is especially important for the numerical resolution part, as experiments actually show that particular settings giving the best results for one of the problems can turn out to be ill-suited to the other one. Indeed, we would like to come up with a reasonably general method that could be applied to various problems, rather than to find the best tuning for a particular case.

Notations: in this chapter, we will often use the subscripts $_1$ *and* $_2$ *for notations specific to Problems 1 and 2, and keep unsubscripted notations for the general case.*

4.1.1 Optimal harvesting in fishery

The first of the two singular arcs problems we consider is a optimal harvesting model in fishery described by Clark in [20], and studied for instance by Schilling in [34][1]. The state $x(t) \in \mathbf{R}$ represents the fish (halibut) population, and the control $u(t) \in \mathbf{R}$ is the fishing activity. The objective is to maximize the harvesting over a certain fishing period, the fishing activity being capped by a certain value U_{max}. The model of the harvesting product and fish population growth is described in the original formulation of the

[1] where the BVP is interpreted in the context of fixed points for set valued operators

problem:

$$(P)\begin{cases} Max\ \int_0^{10}\ (E-\frac{c}{x(t)})\ u(t)\ dt \\ \dot{x}(t)=rx(t)\ (1-\frac{x(t)}{k})\ -\ u(t) \\ 0\leq u(t)\leq U_{max}\quad \forall t\in[0,10] \\ x(0)=70.10^6\quad x(10)\ free \end{cases}$$

with $E=1$, $c=17.5\ .10^6$, $r=0.71$, $k=80.5\ .10^6$ and $U_{max}=20\ .10^6$.

We note that the numerical values of the constants above are such that the term $(E-\frac{c}{x(t)})$ is always positive, which ensures a positive harvesting product. Incidentally, one might want to add a state constraint of the form $x(t)>0,\quad \forall t\in[0,10]$, as a negative fish population would make little sense... However, in all the numerical experiments in the following we never actually encounter this case, so we just let it as it is. Needless to say, the problem would be much more complicated if this constraint had to be taken into account explicitly.

We now slightly modify this problem statement in order to obtain a similar formulation to the other problems we study. First we transform the maximization problem into a minimization one, with the opposite objective (remember that we have $(\frac{c}{x(t)}-E<0)$, which will play a part in the formulation of the continuation approach, see 4.2 page 90 below). Then we replace the expression of the control $u(t)$ by $u(t)\ U_{max}$, with the constraint on the control rewritten as $0\leq u(t)\leq 1\quad \forall t\in[0,10]$. So the problem we actually work with in the following is:

$$(P_1)\begin{cases} Min\ \int_0^{10}\ (\frac{c}{x(t)}-E)\ u(t)\ U_{max}\ dt \\ \dot{x}(t)=rx(t)\ (1-\frac{x(t)}{k})\ -\ u(t)\ U_{max} \\ 0\leq u(t)\leq 1\quad \forall t\in[0,10] \\ x(0)=70.10^6\quad x(10)\ free \end{cases}$$

To apply the Pontryagin's Maximum Principle, we begin with the expression of the Hamiltonian

$$\mathcal{H}_1(t,x,p,u)=\left(\frac{c}{x(t)}-E\right)u(t)U_{max}+p(t)\left(rx(t)\left(1-\frac{x(t)}{k}\right)-u(t)U_{max}\right).$$

This gives the state and costate dynamics

$$\begin{cases} \dot{x}(t)=\ \ r\ x(t)\ (1-\frac{x(t)}{k})-u(t)U_{max} \\ \dot{p}(t)=\ \ \frac{c}{x^2(t)}\ u(t)U_{max}-p(t)\ r\ \left(1-\frac{2x(t)}{k}\right). \end{cases}$$

We have the boundary conditions

$$\begin{cases} x(0)=70.10^6 & p(0)\ free \\ x(10)\ free & p(10)=0. \end{cases}$$

Now concerning the optimal control u^*, the Hamiltonian minimization leads to the following switching function ψ:

$$t \in [0, 10] \mapsto \psi(t) = \frac{c}{x(t)} - E - p(t),$$

and the corresponding bang-bang optimal control

$$\begin{cases} u^*(t) = 0 & if\ \psi(t) > 0 \\ u^*(t) = 1 & if\ \psi(t) < 0 \\ u^*(t) \in [0, 1] & if\ \psi(t) = 0. \end{cases}$$

The latter case usually indicates a commutation, where the control is discontinuous and switches from one bang arc to another as ψ vanishes. It is not that annoying by itself if this happens only at isolated times, apart from the fact that a high number of commutations can make the numerical integration of the BVP a bit tricky, and also lead to difficulties in applying the single shooting method (as said in 1.1.1). Nevertheless, we have seen in chapter 3 that using a variable step integrator and a well-chosen continuation could give some good results.

However, we precisely study here the case of singular arcs, namely we expect the switching function ψ to vanish over a non-trivial interval. The necessary conditions given by the Hamiltonian minimization alone are insufficient to determine uniquely the value of the optimal control over such a singular arc, which is of course more problematic than a fixed number of discontinuities.

Then we naturally try to extract some information concerning the control from the fact that the successive derivatives of the switching function must vanish over a singular arc.

Remark: we recall that t is omitted for clarity in the following equations, as it does not appear by itself anyways. We thus note x, p, u, ψ instead of $x(t), p(t), u(t), \psi(t)$.

The first derivative of the switching function is of the form

$$\dot{\psi} = r\ \left(-\frac{c}{x} + \frac{c}{k} + p - \frac{2\ p\ x}{k} \right).$$

It happens that u has vanished from this expression, so the relation $\dot{\psi} = 0$ provides no information on the control. We then look at the second derivative

$$\ddot{\psi} = r^2\ \left(-\frac{2}{kr}\ (\frac{c}{x} - p)u\ U_{max} + \frac{c}{x} - \frac{c}{k} - p + \frac{2\ p\ x}{k} - \frac{2\ p\ x^2}{k^2} \right).$$

Luckily, the equation $\ddot{\psi} = 0$ gives the expression of the singular control

$$u^*_{singular} = \frac{k\ r}{2\ (\frac{c}{x} - p)\ U_{max}} \left(\frac{c}{x} - \frac{c}{k} - p + \frac{2px}{k} - \frac{2px^2}{k^2} \right).$$

The problem is of course that this expression is only usable if we *know* that we are over a singular arc. Another point is that it is not obvious from this formula that $u^*_{singular}$ always satisfies the constraint $0 \leq u(t) \leq 1, \forall t \in [0, 10]$. In practice however, this algebraic expression of the singular control is only used for the precise resolution of the problems described later in 4.4. In our case it turns out that the initialization provided by the continuation (see 4.2 and 4.3 below) is close enough to the solution so that we do not encounter such "out of the bounds" singular controls.

Remark: incidentally, combining the three equations $\psi, \dot{\psi}, \ddot{\psi} = 0$ yields $\dot{x} = 0$ and $\dot{p} = 0$, so the state and costate are both constant over a singular arc, and therefore so is the singular control value. This is of course quite specific of this problem, and by no means a general result.

4.1.2 Quadratic regulator

The second singular arc problem we study is a quadratic regulator problem studied for instance in [19] by Chen and Huang (who also use a perturbation approach, but with a discrete continuation based on the Davidenko differential equation). The state is this time of dimension 2, while the control is still of dimension 1. The objective is to minimize the squared L_2-norm of the state.

$$(P_2) \begin{cases} Min\ \frac{1}{2} \int_0^5 (x_1^2(t)\ +\ x_2^2(t))\ dt \\ \dot{x_1}(t) = x_2(t) \\ \dot{x_2}(t) = u(t) \\ -1 \leq u(t) \leq 1 \quad \forall t \in [0, 5] \\ x(0) = (0, 1) \quad x(5)\ free. \end{cases}$$

We have the expression of the Hamiltonian

$$\mathcal{H}_2(t, x, p, u) = \frac{1}{2}(x_1^2(t) + x_2^2(t)) + p_1(t)x_2(t) + p_2(t)u(t),$$

which gives the simple dynamics of the state and costate

$$\begin{cases} \dot{x_1}(t) = & x_2(t) \\ \dot{x_2}(t) = & u(t) \\ \dot{p_1}(t) = & -x_1(t) \\ \dot{p_2}(t) = & -p_1(t) - x_2(t). \end{cases}$$

With the boundary conditions

$$\begin{cases} x(0) = (0, 1) & p(0)\ free \\ x(5)\ free & p(5) = (0, 0). \end{cases}$$

The Hamiltonian minimization immediately indicates that the switching function is

$$\psi(t) = p_2(t),$$

and the corresponding optimal control is bang-bang again, namely

$$\begin{cases} u^*(t) = -\ sign\ p_2(t) & if\ \psi(t) \neq 0 \\ u^*(t) \in [-1, 1] & if\ \psi(t) = 0. \end{cases}$$

There again, we expect ψ to vanish over nontrivial intervals, and we try to obtain the expression of the singular control by using the successive derivatives of ψ, same as before. The first order comes directly from the dynamics:

$$\dot{\psi}(t) = -p_1(t) - x_2(t)$$

As for Problem 1, this equation is useless concerning the control, so we look at the second order

$$\ddot{\psi}(t) = x_1(t) - u(t),$$

which immediately gives the expression of the singular control (via $\ddot{\psi} = 0$)

$$u^*_{singular}(t) = x_1(t).$$

We make here the same remark than before concerning the constraint on the control, which is not automatically satisfied by this expression. However, the situation is in practice the same as for problem (P_1), and out of the bounds singular control do not occur during the numerical resolution either.

4.2 Quadratic perturbation

As for the orbital transfer studied in the previous chapter, a direct resolution of these two problems by single shooting is not possible due to the presence of singular arcs. Furthermore, the use of more evolved methods such as multiple shooting would require some qualitative knowledge of the control structure, more precisely the number and approximate location of possible singular arcs. Then we would also need a reasonably good initialization at the multiple shooting nodes, especially for the costate, than has no obvious physical interpretation and is therefore difficult to estimate a priori. So we try to use a continuation approach like before, but this time more specifically to find out what the singular control structure looks like, and to obtain an approximation of the state and costate at the solution. We can then use this information to solve the problems precisely with a multiple shooting-like method.

4.2.1 Formulation

The setting is here a bit different from the orbital transfers problems, where the main difficulty is the huge number of commutations for low thrusts. Both (P_1) and (P_2) deal with a fixed final time, and the difficulty is the presence of singular arcs. So the objective of the continuation is to regularize the problems to get rid of the singular arcs, yet we still want to acquire reliable information on the singular structure at the solution. We try to do this by adding a quadratic perturbation to the original criteria, such as done for instance by Chen and Huang for (P_2) in [19] (the difference here is that we use the simplicial method to perform the continuation, instead of a discrete continuation based on the Davidenko differential equation). A consequence of this perturbation is to make the Hamiltonian strictly convex with respect to the control, which prevents the appearance of singular arcs. We hope that by using the regularization parameter as the homotopy parameter, the continuation will allow us to predict the apparition of singular arcs.

In all the following we try to deal with both (P_1) and (P_2) in a similar way, as the objective in studying two problems in parallel is to avoid a problem dependant formulation and numerical resolution method. Here are the regularized criteria that we use for the continuation:

$$Min \int_0^{10} \left(\frac{c}{x(t)} - E\right) (u(t) - (1-\lambda)\ u^2(t))\ U_{max}\ dt, \quad \lambda \in [0,1]$$

and

$$Min\ \frac{1}{2} \int_0^5 (x_1^2(t) + x_2^2(t))\ +\ (1-\lambda)u^2(t)\ dt, \quad \lambda \in [0,1].$$

For problem (P_1), as mentioned before the term $\frac{c}{x(t)} - E$ is always negative, so the minus sign before $(1-\lambda)\ u^2(t)\ U_{max}$ actually results in "adding" a quadratic term, as for problem (P_2). We then obtain two families of boundary value problems parametrized by λ, denoted by $(BVP_1)_\lambda$ and $(BVP_2)_\lambda$ respectively. The original unperturbed problems correspond of course to the case $\lambda = 1$.

Back to the results of 1.2.1 (page 20) concerning the Hamiltonian minimization, we have the following domains for the controls:

$$U_1 = [0,1] \quad and \quad U_2 = [-1,1],$$

and the Hamiltonians are (with t omitted for clarity):

$$\mathcal{H}_1(t,x,p,u) = \left(\frac{c}{x} - E\right)(u - (1-\lambda)u^2)U_{max} + p(rx\ \left(1 - \frac{x}{k}\right)\ -\ u\ U_{max})$$

and

$$\mathcal{H}_2(t,x,p,u) = \frac{1}{2}(x_1^2 + x_2^2\ +\ (1-\lambda)u^2) + p_1 x_2 + p_2 u.$$

We can see that both $\mathcal{H}_1$ and $\mathcal{H}_2$ are continuous, and convex with respect to u (again, we have $\frac{c}{x} - E < 0$). So for both problems (P_1) and (P_2), Theorem 1 and Corollary 1 apply, thus Γ_1 and Γ_2 are upper-semicontinuous, and non empty compact convex valued. These properties will be useful for the following convergence results concerning the continuation. We can also note that for $\lambda < 1$, both Hamiltonians are strictly convex, thus the optimal controls are continuous functions, according to Corollary 1.

We recall the state-costate dynamics for $(BVP_1)_\lambda$ and $(BVP_2)_\lambda$:

$$\varphi_1(y,u,\lambda) = \begin{pmatrix} r\ x\ (1-\frac{x}{k}) - u\ U_{max} \\ \frac{c}{x^2}\ (u-(1-\lambda)u^2)U_{max} - p\ r\ (1-\frac{2x}{k}) \end{pmatrix}$$

and

$$\varphi_2(y,u,\lambda) = \begin{pmatrix} x_2 \\ u \\ -x_1 \\ -p_1 - x_2 \end{pmatrix}.$$

We consider the set valued dynamic $\Phi(y,\lambda) = \varphi(y,\Gamma(y),\lambda)$. From the expression of φ_1 and φ_2, and the fact that Γ_1 and Γ_2 are usc and non empty compact convex valued, we obtain that Φ_1 and Φ_2 are also usc and non empty compact convex valued. Now we make the assumption that the solutions of $(BVP_1)_\lambda$ and $(BVP_2)_\lambda$ remain in some fixed compacts. According to Theorem 5, we know that from a sequence of solutions (y_{λ_k}) such that λ_k tends to 1, we can extract a subsequence (y_k) that converges uniformly to a solution y of (BVP), with $(\dot{y}_k)$ that converges *-weakly to $\dot{y}$ in $L^\infty_n([0,t_f])$.

Moreover, we have the following expression of the control:

$$\begin{cases} u_\lambda(t) &=& \frac{1}{U_{max}}(r\ x_\lambda(t)\ (1-\frac{x_\lambda(t)}{k}) - \dot{x}_\lambda(t)) & for\ Problem\ 1 \\ u_\lambda(t) &=& \dot{x}_{2_\lambda}(t) & for\ Problem\ 2. \end{cases}$$

Thus Corollary 2 applies and gives the *-weak convergence for the subsequence of the control (u_k). However, the existence of these subsequences does not guarantee the actual success of the continuation, as we shall see during the numerical experiments.

Concerning Theorem 6, (see page 31), we make the assumptions on the path following that all the faces generated by the algorithm remain in $K \times [0,1]$ with K compact, and that the algorithm does not go back to 0. In both cases the homotopy H is upper-semicontinuous, compact valued and acyclic. The latter implies that H is connex valued, which gives the convexity in dimension 1 but not in dimension 2. So the assumptions on the homotopy only hold in the case of (P_1), but not (P_2) a priori. This

point probably deserves further investigation, but this would require dealing with some topological degree, however... In practice, the path following converges for both problems, but we still encounter some difficulties related to the presence of singular arcs, as detailed below.

Remark: incidentally, one might wonder what happens with problem (P_1) *if we set a plus sign before the perturbation term, "same as with the orbital transfer and problem* (P_2)*". Well, the corresponding Hamiltonian is concave, not convex, which leads to a discontinuous control anyways. The resulting continuation is not very useful in practice, to say the least...*

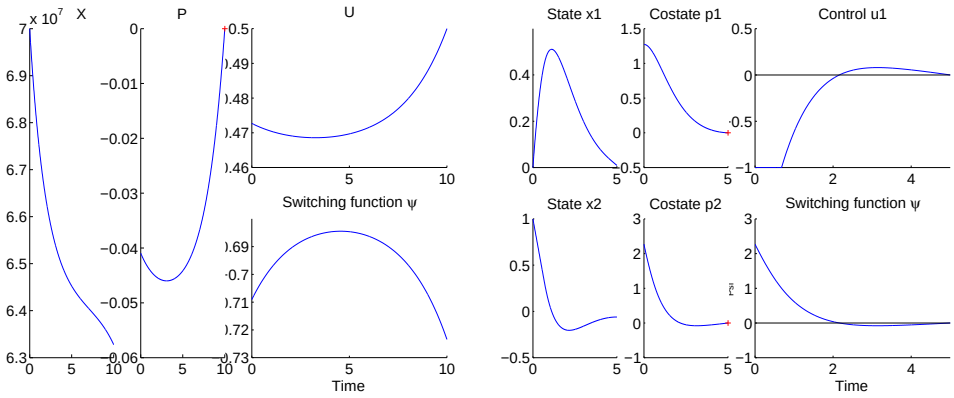
4.2.2 Path following

Initialization

In order to initialize the continuation, we need to solve both problems for $\lambda = 0$, which is easily done by single shooting. We use here a fixed step (1000) Runge Kutta4 integration, the results with variable step integrators being similar. The convergence of the single shooting is immediate for both problems, with a trivial choice of starting points, namely $z_0 = 0.1$ and $z_0 = (0, 0)$ respectively. Here are the results of the single shooting and the corresponding solutions for $\lambda = 0$.

	z_0	z^*	$\lvert S_0(z^*)\rvert$	objective	iter	time
Problem 1	0.1	$-4.0935\ 10^{-2}$	$3.6295\ 10^{-16}$	69374046	39	$< 1s$
Problem 2	$(0, 0)$	$(1.2733, 2.2715)$	$3.3596\ 10^{-14}$	0.4388	134	$< 1s$

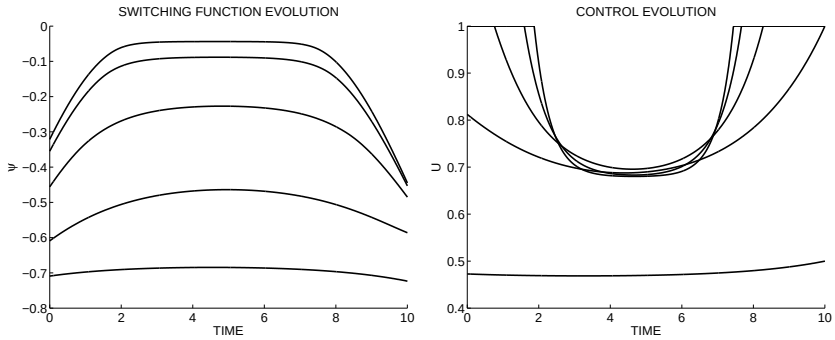
Single shooting results at $\lambda = 0$.



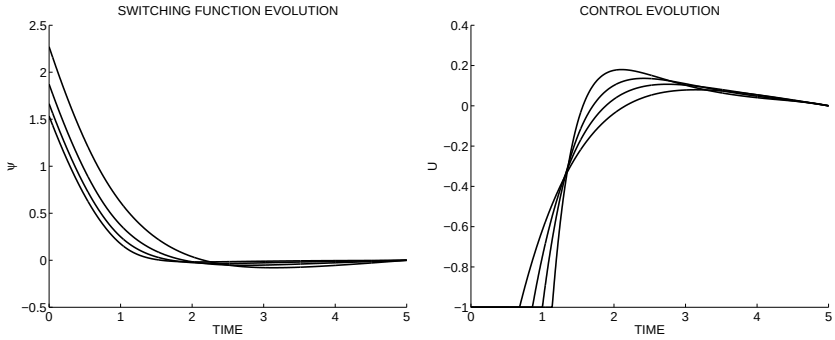
Solutions for Problem 1 and 2 at $\lambda = 0$

Singular structure detection

For both problems, the path following goes smoothly at first, and the evolution of the switching functions and controls is quite interesting, as shown below.



Problem 1: Switching function and Control for $\lambda = 0, 0.5, 0.75, 0.9$ *and* 0.95



Problem 2: Switching function and Control for $\lambda = 0, 0.5, 0.75$ *and* 0.9

We can clearly see that for both problems, the switching function ψ comes closer to 0 over certain time intervals as λ grows. This strongly suggests the presence of singular arcs over these intervals at the solution for $\lambda = 1$, and already gives some hints about the possible control structures.

For Problem 1, the form of the structure seems to be *regular-singular-regular*, with the possible singular arc located around the time interval $[2, 7.5]$. For Problem 2, we seem to have a *regular-singular* structure, with the beginning of the singular arc near $t = 1.5$.

Meanwhile, by looking at the controls evolution we observe that outside the suspected singular arcs, the control tends for both problems to a bang-bang structure, which is coherent with the necessary conditions for $\lambda = 1$. More precisely, we have two bang arcs with $u = +1$ flanking the singular arc for Problem 1, and a bang arc with $u = -1$ before the singular arc for Problem 2.

Another remarkable fact about the controls is that both of them keep on taking interior values over time intervals matching those where ψ tends to 0. As the necessary conditions for $\lambda = 1$ lead to a bang-bang control for both problems, this is an additional hint of the presence of singular arcs

over these intervals.

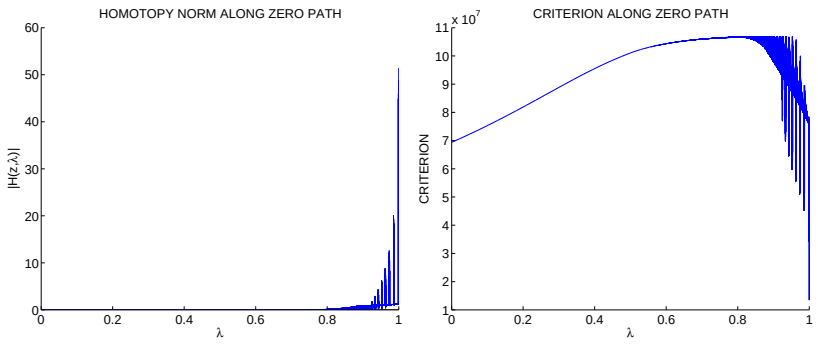
So at this stage, the continuation based on the quadratic perturbation looks quite promising. For both problems, we already have a strong indication about the singular structure, with an approximate location of the suspected singular arcs. As far as the detection of the singular arcs is concerned, this approach seems rather effective.

Numerical difficulties

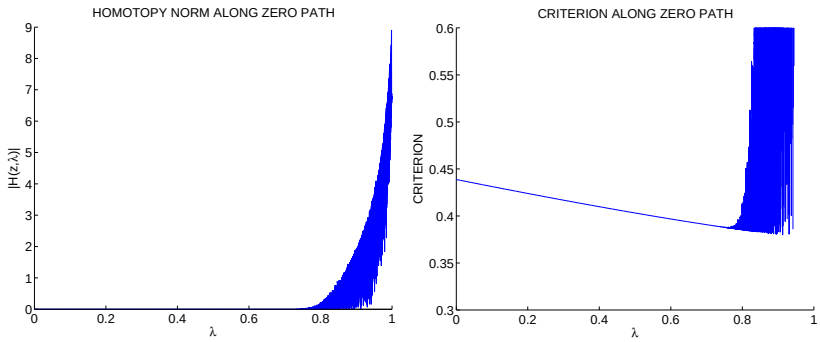
However, if we pursue the continuation, the path following encounters some difficulties as λ tends to 1. For both problems, above a certain point, the piecewise linear approximation of the homotopy seems to become increasingly inaccurate. This phenomenon is illustrated on the two left graphs below, as we see the homotopy norm along the path increase very steeply, with a very oscillatory behaviour.

Moreover, we also observe that from this point on, the objective value does not improve any longer, and displays the same kind of brutal degradation, with a lot of oscillations. We recall that on all graphs, the objectives displayed for Problem 1 and 2 correspond to the original, non-perturbed criteria

$$\begin{cases} J_1 = Max \int_0^{10} \left(E - \frac{c}{x(t)}\right)\ u(t)\ U_{max}\ dt \\ J_2 = Min \frac{1}{2} \int_0^5 (x_1^2(t) + x_2^2(t))\ dt \end{cases}$$



Homotopy norm and objective along the zero path - Problem 1



Homotopy norm and objective along the zero path - Problem 2

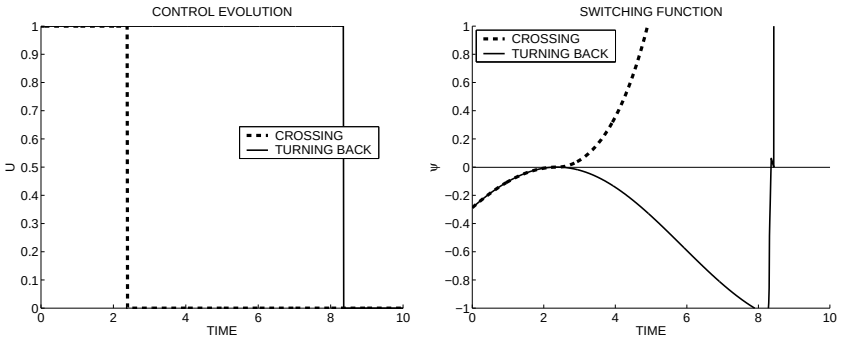
Difficulties are often expected at the end of continuation strategies, all the more so in this particular case, with the presence of singular arcs. For simplicial methods, one can think of using some refining triangulations (such as J_3 or J_4), and hope that the decreasing meshsize would help to keep an accurate path following until the convergence. In this case, however, the use of such techniques only delays this degradation a little, and does not prevent it from appearing eventually. Incidentally, we have observe during these tests that the value of λ at which the degradation occurs is problem-dependant. With the use of refining triangulations, we have located this instability threshold around $\lambda = 0.975$ for Problem 1 and a bit earlier, around $\lambda = 0.95$, for Problem 2.

4.2.3 Two different control structures

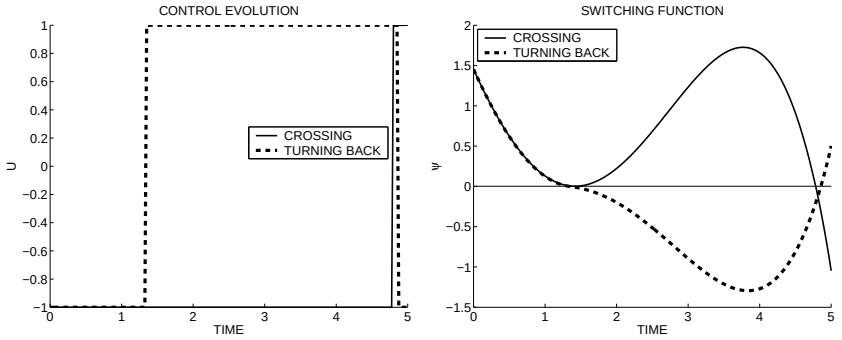
As the PL approximation of the homotopy is obtained by interpolation over the vertices labeling, we investigate the corresponding trajectories more closely. Then if we look at the control structures corresponding to the vertices, we indeed find the cause of the degradation of H_T. Over the last completely labeled face, we clearly observe two distinct control structures, depending on the vertices. This difference naturally leads to the same separation among the labels, and explains the inaccuracy of the approximation.

We show below for both problems the two kinds of control structures that are present among the vertices of the last completely labeled face for $\lambda = 1$. For each problems, both possible control structures are bang-bang, and there is no trace of singular arcs anymore.

Note: the terms "crossing" and "turning back" applied to the control structures are explained below.



Two different control and switch structures at the vertices - Problem 1



Two different control and switch structures at the vertices - Problem 2

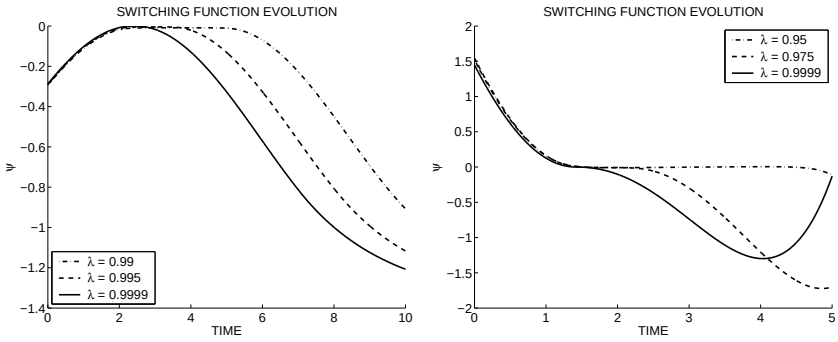
We notice, however, that the two structures are identical up to a certain time, namely when the switching function ψ vanishes. Then we have the two possibilities corresponding to the sign of ψ after this zero, with the two opposite bang arcs. More precisely, in one case the switching function keeps the same sign, and the initial bang arc continues. In the other one the sign of ψ changes, and so does the bang arc: we have a commutation on the control. After this, the trajectories are distinct and continue on their own separately. We note that these two structures keep appearing among the vertices of the completely labeled faces, however small a meshsize we use. This explains why refining triangulations are eventually unable to prevent this phenomenon.

So we can think that the path following degradation is due to the appearance, at a certain level, of two distinct control structures, numerically merged in terms of homotopy unknown z. It is remarkable that both problems share here this particular behaviour, only with a different threshold value, which is not surprising. In the following we refer to these two cases as "turning back" and "crossing", with respect to the behaviour of the switching function after the 0.

4.2.4 Pseudo-arc instability near $\lambda = 1$

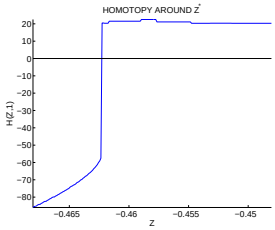
So we have seen that the degradation of the path following is due to the appearance of two distinct control structures, related to the behaviour of the switching function. As said before, we notice in the beginning of the continuation that the switching function gets close to 0 over some time intervals, that we call "pseudo singular arc". We observe that these pseudo singular arcs seem to be numerically unstable: ψ stays near 0 for a while, then at some point it leaves and increases in absolute values, either with positive or negative sign. Depending on the exit sign of the switching function at the end of the pseudo singular arc, we either obtain a "turning back" or "crossing" type control structure.

Now if we examine the evolution of the switching function beyond the instability threshold, we can get a clearer view of what happens. Basically, the numerical instability of the switching function near 0 becomes worse as λ tends to 1. The length of the pseudo singular arcs decrease, as the exit occurs earlier and earlier.

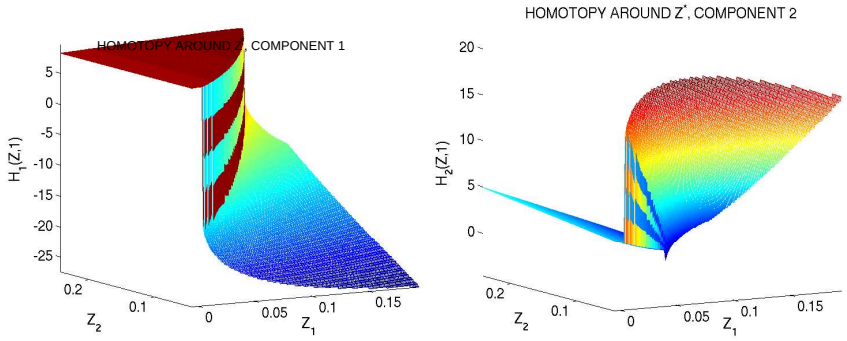


Pseudo singular arc shrinking near the convergence Problem 1 ($\lambda = 0.99, 0.995, 0.9999$) and Problem 2 ($\lambda = 0.95, 0.975, 0.9999$)

At the convergence for $\lambda = 1$, all that is left from the singular arc is a contact point where ψ reaches 0. Depending on the sign of the switching function after this point, we have the two corresponding "turning back" or "crossing" bang bang control structures. We draw here the shooting function in the neighbourhood of the solution z_1^* given by the path following, for Problem 1 and 2.



Shooting function discontinuity at the singular arc - Problem 1



Shooting function discontinuity at the singular arc - Problem 2

In both cases we clearly notice the frontier separating the regions corresponding to the two different control structures. The presence of singular arcs manifests itself in the form of a discontinuity of the shooting functions at the solution. As said in the beginning, the results here correspond to a 1000-step Runge Kutta 4 integration. However, augmenting the number of steps to 10000, or using the three variable step integrators (RKF45, DOP853 and ODEX) gives similar results. The only difference we have noticed when changing the integration is that for (P_2) the path following can converge to different points on the "frontier" at $\lambda = 1$. We also note that the convergence is generally more difficult to attain for (P_2), as we often have to use a J_4 triangulation instead of the more precise J_3. This may be related to the fact that Theorem 6 applies for (P_1), but not for (P_2), as mentioned before.

As a conclusion, we can say that this first continuation has earned a mitigate success. Sure, observing the switching function and control evolution before the instability threshold gives strong hints about the presence of singular arcs. However, due to this instability, we have lost the singular structure by the time we arrive at the convergence, where we obtain an incorrect and useless solution.

4.3 Discretized BVP approach

4.3.1 Formulation

So we have seen that the continuation using a quadratic perturbation of the criterion looks promising, but suffers from numerical instabilities during the IVP integration when the problem becomes nearly singular. We would like to circumvent this difficulty, and we try a new approach, inspired by the multiple shooting principle. The idea is basically to discretize the equations of the Boundary Value Problem (as for instance in [7], with a symplectic Runge Kutta method). We use here a basic Euler scheme for the state and costate, and consider a piecewise constant control, still derived from the

necessary conditions. The values of the state and costate at the interior discretization nodes thus become additional unknowns of the shooting function, while we have the following matching conditions at these nodes, from the Euler scheme:

$$Match_{cond} \left\{ \begin{array}{l} x_i - (x_{i-1} + h\frac{\partial x}{\partial t}(t_{i-1}, x_{i-1}, p_{i-1}, u^*_{i-1})), \quad \forall i \in [1..d] \\ p_i - (p_{i-1} + h\frac{\partial p}{\partial t}(t_{i-1}, x_{i-1}, p_{i-1}, u^*_{i-1})) \end{array} \right.$$

where the optimal control u^*_i is obtained from (x_i, p_i) by the usual necessary conditions, and gives the constant value of the control over the interval $[t_i, t_{i+1}]$. The idea is, that even if the control obtained on the singular arc is irrelevant, we hope to have a good approximation of the state and costate values.

In terms of convergence properties, due to this choice of a very simple integration scheme, the discretized version of the shooting function is compact convex valued. This allows us to hope a good behaviour of the path following, according to Theorem 6 (see 1.2.2).

The "shooting function" associated to this discretized BVP is of the form

$$S_D : \mathbf{R}^{n+(2n)\times d} \rightarrow \mathbf{R}^{n+(2n)\times d}$$

where n denotes the state (and costate) dimension, and d the number of discretization nodes. We summarize here the discretized shooting function unknown and value layouts:

Unknown z

- IVP unknown at t_0 (same as in single shooting method)
- values of (x^i, p^i) at interior times $t_i,\ i \in [1..d]$ (note that $t_d = t_f$)

IVP unknown at t_0	(x^1, p^1)	...	(x^d, p^d)

Value $S_D(z)$

- matching conditions at interior times
- terminal and transversality conditions at t_f (same as single shooting)

$Match_{cond}(t_1)$	$Match_{cond}(t_2)$	...	Conditions at t_f

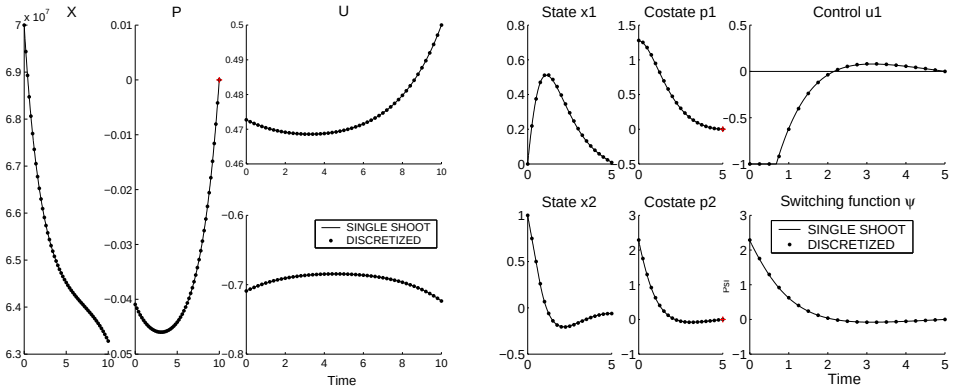
A major drawback of this formulation is that the full state and costate are discretized. This drastically limits the number of discretization nodes, else the high dimension of the unknown leads to prohibitive execution times. As a side effect, this also puts some restrictions on the use of small meshsizes or refining triangulations, for the same computational cost reasons.

Note: In all the following, the discretized BVP formulations use 50 nodes for Problem 1, and only 20 for Problem 2, due to the higher dimension of the state and costate. Even these rather rough discretizations lead to a total dimension of 101 *and* 82 *respectively for the path following...*

4.3.2 Path following

We perform with this new formulation the same continuation as before, based on the quadratic perturbation of the criterion. Once again, solving both problems for $\lambda = 0$ is immediate (less than 1s, with a solution norm below 10^{-15}), even with trivial starting values for the interior state and costate

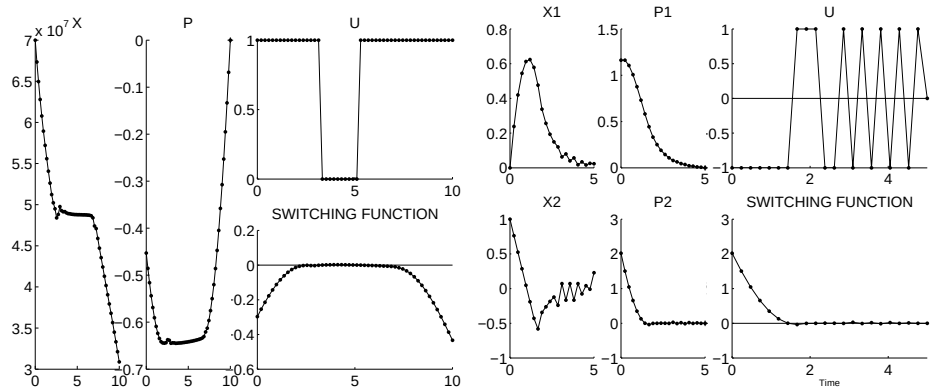
$$\begin{cases} \textit{Problem } 1 & p(0) = 0, & (x^i = x_0 = 7\ 10^7, p^i = 0), & \forall i \in [1..d] \\ \textit{Problem } 2 & p(0) = (1,1), & (x^i = (0,0), p^i = (0,0)), & \forall i \in [1..d] \end{cases}$$



Discretized (Euler) BVP solutions for Problem 1 and 2 at $\lambda = 0$ (plotted over the single shooting solution at $\lambda = 0$)

We note that the solutions for the discretized BVP formulation coincide very well with the previous results from the single shooting formulation (see page 93), despite the rough integration scheme we use in the discretized BVP formulation.

Concerning the path following, things go much more smoothly here, and we attain $\lambda = 1$ without noticing the instability that plagued the single shooting continuation. We plot below the solutions obtained at $\lambda = 1$, with a $K_1(10^{-2})$ triangulation for the path following.



Discretized (Euler) BVP solutions for Problem 1 and 2 at $\lambda = 1$

At first glance, the most interesting part is the allure of the two switching functions. For both problems the presence of the singular arcs is obvious, with ψ nearly vanishing around the time intervals $[2, 7]$ and $[1.5, 5]$ respectively. It is remarkable that despite the little number of discretizations nodes, the switching functions are numerically rather close to 0. For Problem 1, in particular, the singular arc detection is obviously more accurate than with the single shooting formulation just before the instability threshold.

As expected with this formulation, the control is bang bang, according to the necessary conditions at each discretization node. Over the singular arcs intervals, it randomly oscillates between the two extremal possible values ($0, 1$ and $-1, 1$, respectively), depending on the sign of the switching function evaluated at the nodes.

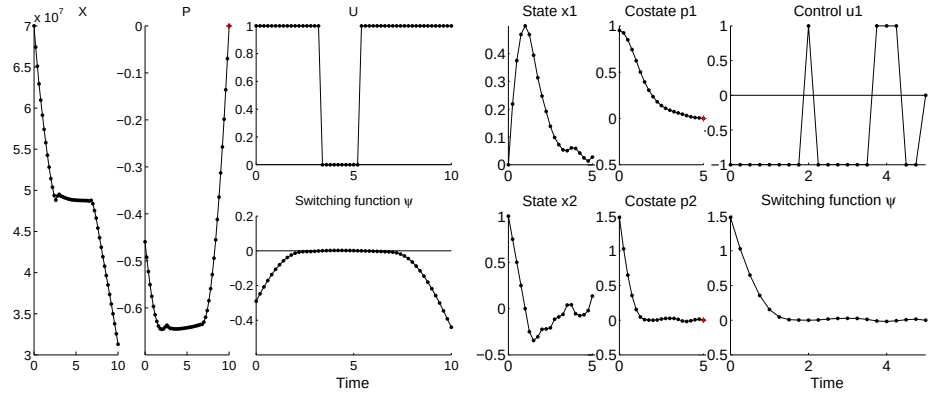
Actually, if we compare a posteriori these trajectories with the precise solutions obtained later (see 4.4), we find that this discretized approximation of x and p is not too bad. For Problem 1, the differences are localized on the singular arc, where the discretized state and costate are nevertheless still close to the solution. For Problem 2 however, the discretized solution is less accurate: the state and costate have the same general allure, but there is a kind of shifting from the solution for x_1, p_1 and p_2, and some strong oscillations for x_2, the latter can be seen on the graph above.

Implicit trapeze formulation

Nevertheless, there is still a part of the discretized formulation that leaves room for improvement, namely the integration scheme itself. The original formulation uses an Euler formula, which is quite crude. This choice is driven by the intent to apply Theorem 6, so we need the shooting function be compact convex valued. However, an implicit trapeze formula also satisfies this requirement, and could be expected to do better than a simple Euler. The only change in the formulation is the new expression of the matching conditions:

$$Match_{cond} \begin{cases} x_i - (x_{i-1} + \frac{h}{2}\left(\frac{\partial x}{\partial t}(t_{i-1}, x_{i-1}, p_{i-1}, u^*_{i-1}) + \frac{\partial x}{\partial t}(t_i, x_i, p_i, u^*_i)\right)) \\ p_i - (p_{i-1} + \frac{h}{2}\left(\frac{\partial p}{\partial t}(t_{i-1}, x_{i-1}, p_{i-1}, u^*_{i-1}) + \frac{\partial p}{\partial t}(t_i, x_i, p_i, u^*_i)\right)) \end{cases}$$

As before with the Euler discretization, the continuation initialization (ie solving the problems for $\lambda = 0$) is immediate. Similarly, the path following itself goes smoothly, and the convergence at $\lambda = 1$ is attained without any particular difficulties, with the solutions below.



Discretized (Trapeze) BVP solutions for Problem 1 and 2 at $\lambda = 1$

The solution for Problem 1 looks nearly identical to the Euler one, but the annoying oscillations on Problem 2 are reduced. More interestingly, the state, costate and switching function of these solutions are a posteriori closer to the accurate solution obtained later. A hint of this can be found by looking at the initial value of p, which is not the same as before (roughly $(1, 1.5)$ versus $(1.15, 2)$ for Euler, the accurate solution being near $(0.9422, 1.4419)$). So it would seem that the flaws of the previous solution for Problem 2 is partly due to the Euler integration, and can be improved with a better choice, which is rather comforting. Indeed, this second solution can be further improved by the use of adaptive meshsize and solution refinement (see 4.3.3), contrary to the Euler solution.

Remark: one can think of trying more precise integration, even if the do not verify the assumptions for Theorem 6. In practice, using Runge Kutta 2nd and 4th order actually gives less good solutions, with a shifting from the correct solution observed on x_2 (this is maybe related to the fact that the convexity assumption does not hold for (P_2)). We have also made very basic experiments with some symplectic integrators, see Appendix B.3.2 page 147.

These solutions are actually already sufficient to detect the singular structure, and to initialize properly the accurate resolution of the problems, as explained in 4.4.

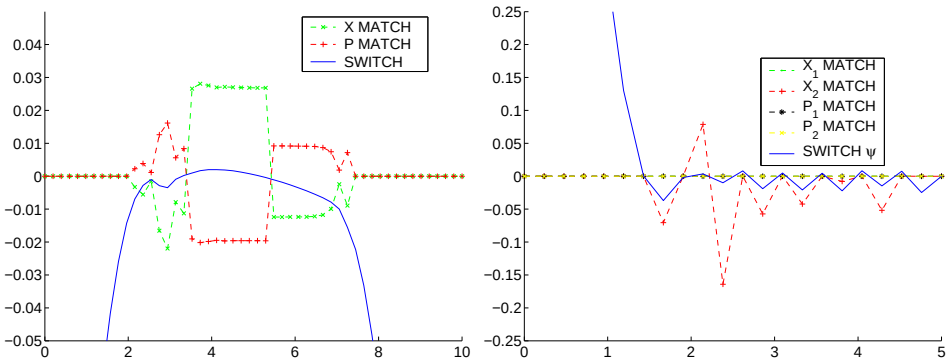
Matching condition errors

Besides the replacement of the Euler formula, one might think of simply using more discretization nodes in order to improve the solutions. However, the experiments we have done in this direction have not been very convincing, plus this strategy soon meets the limitation of the problem size anyways. Similarly, using refining triangulations has not given better results, and the shooting attempts at $\lambda = 1$ have also been unsuccessful (solution refining,

however, gives some positive results, see page 106).

These difficulties might come from the expression of the control, which is still given by the necessary conditions. On a time interval $[t_i, t_{i+1}]$ located within a singular arc, let us assume that the continuation has found the correct values of the state and costate (x^i, p^i) and (x^{i+1}, p^{i+1}). As we are supposed to be on a singular arc, the switching function value $\psi(t_i)$ should be near 0, but of course, numerically, it will be a small positive or negative value (the rough discretization even makes it quite unlikely that this value be a "numerical zero" below the machine precision). According to this sign the necessary conditions then give a bang-bang control u_i, that is of course different from the actual singular control u_i^*. Therefore the matching conditions on the state and costate at t_{i+1} may not be satisfied, even though we have indeed their correct values. The numerical consequences on the solution of this flaw inherent to the discretized BVP formulation are hard to predict, but the continuation seems to find a middle ground between the correct state-costate and the resulting matching errors.

We now have a closer look at the value of the matching conditions on (x, p) for the solution at $\lambda = 1$ (Euler formulation). On the following graphs are drawn, for both problems, the switching function ψ (plain line), and each component of the matching conditions on the state and costate (dashed line, with symbols marking the nodes). We thus have a total of 3 curves for Problem 1 ($(x, p) \in \mathbf{R}^2$), and 5 for Problem 2 ($(x, p) \in \mathbf{R}^4$).



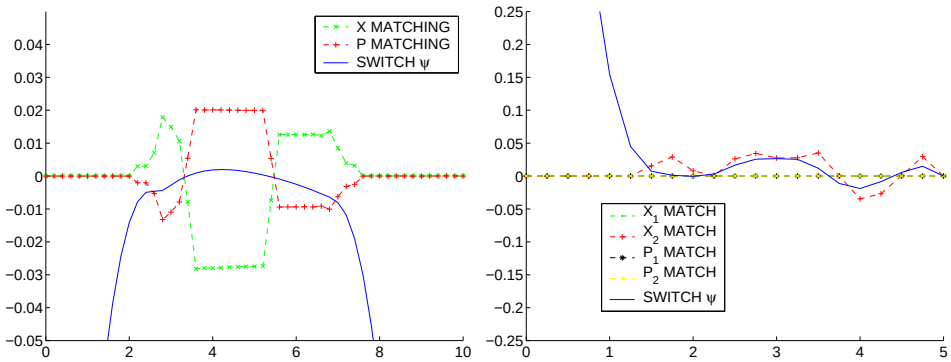
Matching conditions for Problem 1 and 2 at $\lambda = 1$ (Euler)

The most interesting thing about these two graphs is the localization of the matching errors. We immediately notice that non-zero components of the matching conditions are found only for discretization nodes located inside the singular arcs. Over the bang arcs, where the control given by the necessary conditions is correct, the matching errors numerically vanish (below 10^{-14}) for all components, for both problems.

Moreover, for Problem 2 we see that matching errors only occur for the second component of the state (x_2). Other components (x_1, p_1, p_2) always have numerically verified matching conditions, even over the singular arcs. Now if we look at the expression of the state-costate dynamics for Problem 2 (see page 89), we observe that the component x_2 is precisely the only one in which the control u appears. For Problem 1, both derivatives of x and p involve the control (see page 86), so it is not surprising that both matching conditions are non-zero on the singular arc.

Furthermore, we also note that the sign of the matching errors changes accordingly to the sign of the switching function. This is consistent with the previous point, as this sign changes also correspond to the switchings of the incorrect bang-bang control from the necessary conditions. Given the actual values of the singular control, these switchings typically lead to a change of sign of the "error" on the control, namely $u^i - u^i_{singular}$.

Here are now the corresponding matching conditions at the solution, with the switching function, for the trapeze formulation.



Matching conditions for Problem 1 and 2 at $\lambda = 1$ (Trapeze)

We observe the same qualitative behaviour than with the Euler discretization, only with lower matching errors for Problem 2. So it would seem that the link between the matching errors at the solution and the wrong control on singular arcs is not a consequence of the integration scheme used, but indeed a characteristic of the discretized BVP formulation.

At this point, it seems safe to assume that these persistent matching errors at the solution are directly related to the wrong optimal control value given by the necessary conditions on the singular arcs.

Using the algebraic expression of the singular control ?

So it seems that the reason that prevents us from obtaining better solutions is the wrong control value given by the necessary conditions over the singu-

lar arcs. But after all, we do know the algebraic expression of the singular control (see p.89 and 90) ! Then a legitimate idea would be to use this expression in the following way: if at a given node t_i we suspect that we are within a singular arc, then we use the exact expression of the singular control $u_{singular}$ instead of the incorrect optimal control U_i^* given by the necessary conditions. This idea seems interesting at first, but its practical implementation turns out to be quite problematic.

The crucial point is the detection of a suspected singular arc, and we have tried several formulations involving the norm of the switching function ψ and its derivatives (that are supposed to vanish over singular arcs). However, numerical tests with conditions of the form

$$Sing_{cond} \begin{cases} \|\psi(t_i)\| < \epsilon \\ \psi^2(t_i) < \epsilon \\ \psi^2(t_i) + \dot{\psi}^2(t_i) < \epsilon \\ \psi^2(t_i) + \dot{\psi}^2(t_i) + \ddot{\psi}^2(t_i) < \epsilon \end{cases}$$

have all shown an extreme sensitiveness with respect to the choice of the tolerance ϵ, to say the least.

Indeed, changing the value of this parameter leads to completely different solutions, sometimes better, but sometimes far worse than with the classical formulation. As we have not found any consistent link (for both problems at the same time) between the value of ϵ and the quality of the solutions, we have finally abandoned this idea. The root of the problem seems to be that this formulation tends to "force" singular arcs too abruptly, so maybe a softer way of introducing $u_{singular}$ (or a much clever way of detecting the singular case) could make this idea usable in practice.

4.3.3 Adaptive triangulation and solution refinement effect

For the discretized BVP formulation, we have observed that the final shooting does not improve the solution at $\lambda = 1$, regardless of the integration used. This is probably due to the matching errors on singular arcs described before, which are inherent to the formulation we use. So we try to see how the adaptive meshsize and solution refining mechanisms perform in this case, in terms of solution precision. We first compare the basic, fixed uniform meshsize with the full adaptive mode (see 2.3 page 37). Then we add to the adaptive mode the solution refinement (see 2.4.2 53) for the third test, with 10 refinement attempts (allowing 20 refinements does not improve the solutions).

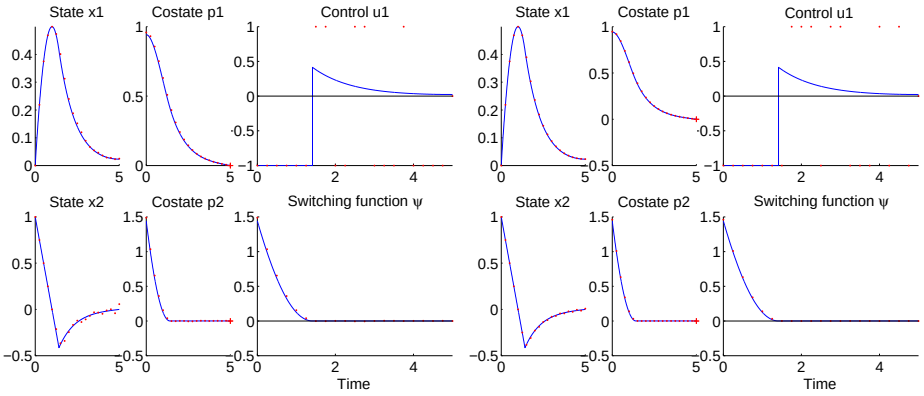
Summary: the three configurations compared in the tests are:
- fixed uniform meshsize, no solution refinement, aka "uniform".

- full adaptive meshsize, no solution refinement, aka “adaptive”.
- full adaptive meshsize, 10 refinement attempts at $\lambda = 1$, aka “refined”.

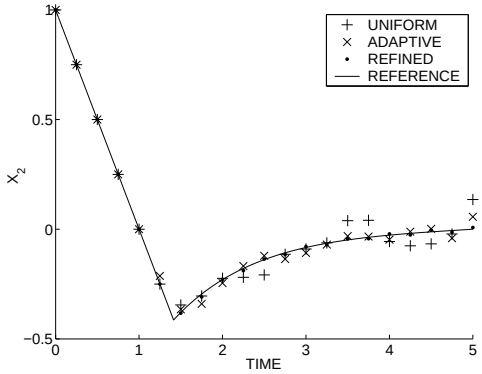
All tests use a K_1 triangulation, with an initial meshsize of 10^{-2}, and the implicit trapeze integration (the same tests with the Euler integration have not been convincing, failing in particular to reduce the oscillations observed before). As we look here for a better final precision rather than performance[2], the deviation control has been set to the more cautious value of 10^{-2}, that has turned out to be superior in this case to the default value (10^{-1}).

The ”uniform” solutions are the ones we have already obtained, and we show here the “adaptive” and “refined” solutions for Problem 2, plotted over the reference solution obtained by structured shooting (see 4.4 just below).

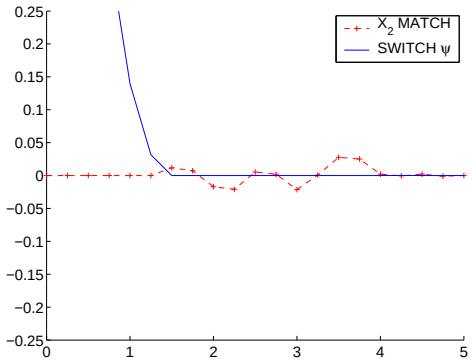
Remark: Solutions for Problem 1 are not represented, as the three of them are very close and nearly identical visually.



Solutions for adaptive and refined triangulations, over reference solution.



[2]these paths take only a few seconds to complete anyway



x_2 comparison for all configurations, and x_2 matching for refined.

We can see (with good eyes) that the "adaptive" and "refined" solutions are indeed closer to the reference, which is reassuring somehow. The comparison is more significant for the x_2 component, which is drawn separately on the third graph above for all three test configuration, in addition to the reference solution in solid line. We recall that x_2 is the only one whose derivative involves the control u, which probably explains why it is the one displaying the most irregularities.

We notice that the refined solution still displays some stray points, which cannot seem to be suppressed by increasing the number of refinement attempts. We probably reach here the limits of the discretized formulation, as the matching errors described before prevent us to find the exact solution anyways. Nevertheless, as we can see on the comparison on page 117 with the reference solution, the refined solution is actually quite good a posteriori. We have a very close approximation of the state and costate, and also especially of the switching function, with numerical values over the singular arcs as low as $10^{-5} - 10^{-6}$ for both problems. The difference with the "uniform" solution is quite clear on the graphs showing the matching conditions, where we used to have some oscillations around 0 for the switching function (see page 105).

Here are the details about the path followings: in addition to the final norm, we indicate the number of simplices for the main path (**splx**), the junction simplices including refinement junctions (**J splx**), and the total homotopy calls (**H calls**).

Problem 1	splx	J splx	H calls	$\lvert S_1(z^*)\rvert$
uniform	107 360	0	107 578	1.69
adaptive	34 920	754	37 404	$8.81\ 10^{-1}$
refined	34 920	1 282 374	249 154	$7.19\ 10^{-1}$

Problem 2	splx	J splx	H calls	$\lvert S_1(z^*)\rvert$
uniform	25161	0	25 837	$9.68\ 10^{-2}$
adaptive	9347	8 656	27 152	$8.21\ 10^{-2}$
refined	9347	220 868	91 271	$5.51\ 10^{-2}$

We observe that the adaptive mode does well in terms of performance, with a quite faster path following for Problem 1. Then we see that if the solution refinement manages to further improve the solutions, it is at a cost however[3]. Incidentally, we see that the final norms decrease -slightly- in accordance to the quality of the solutions. However, it should be kept in mind that, due to the remaining matching errors, this norm is not always a reliable indicator of the quality of a solution. In particular, the "refined" Euler solution for Problem 2 actually has a low final norm, even though it still displays strong oscillations as before, and has a very poor objective value.

Preliminary conclusion

The discretized BVP formulation has been introduced in an effort to avoid the numerical instability of the single shooting continuation as the problem becomes nearly singular. The numerical experiments have indeed shown that this particular difficulty is avoided, and have also pointed out some limitations of the formulation.

First, the roughness of the integration used, with only few nodes and an extremely simple scheme, which has been improved a little by replacing the original Euler formula with an implicit trapeze rule. Then, a more subtle point, the matching errors induced by the incorrect expression of the control over the singular arcs, which we have been unable to suppress completely at this time. Nevertheless, even with these flaws we already obtain a reliable detection of the singular structure and a good initialization for a precise resolution, as described below.

Besides, with the use of both the adaptive meshsize and solution refinement mechanisms presented in 2.1, we also manage to come up with a quite close approximation of the state, costate and especially switching function, even if this is not really needed in the following.

4.4 Structured shooting

As we have seen, the first approach with the quadratic perturbation of the criterion already gives a hint about the singular structure of the two problems. Then with the continuation on the discretized formulation of the BVP, we obtain a clearer idea of this structure, with an accurate detection of the singular arcs. Now we want to solve the problems precisely, by using this knowledge of the structure. We use a variant of the classical multiple shooting method that we call "structured shooting". It shares the same principle

[3]but once again, performance is not our primary concern here, and the refined configurations still only take about 2 minutes to converge...

as the well known code BNDSCO from Oberle (see [31]), slightly simplified and adapted to the singular case instead of the state constraints. The first of the two main differences is that singular arcs, unlike state constraints, do not cause costate discontinuities, therefore there are no jump conditions to worry about during the integration. The second simplification is that instead of having a fixed time subdivision for the multiple shooting in addition to the switching times delimiting the structure changes, we only keep the switching times. We integrate the IVP arc by arc, with the usual optimal control on bang (non-singular) arcs, and singular control (given by the relation $\ddot{\psi} = 0$, cf 4.1.1 and 4.1.2 pages 89 and 90).

Remark: if we look more closely at the expressions of the singular control mentioned above, we notice that these values do not verify the constraint $u \in U$ a priori. It turns out that the initializations provided by the two previous continuations are precise enough to prevent this from happening in practice. However, the numerical experiments on perturbed initializations indeed show that some "out of the bounds" singular control values can occur if we are far from the solution, see page 116.

4.4.1 Formulation

We describe here two formulations for the structured shooting method, first the complete one then a simplified version.

Complete structured shooting

So in addition to the classical IVP unknown z (which for the two problems is the initial costate $p(t_0)$), we now have the interior switching times that delimit the singular arcs, and the corresponding state and costate at these times. On the other hand, we have a set of new conditions that must be verified by these unknowns, so that the shooting function for this formulation is of the form

$$S_{Struct} : \mathbf{R}^{n+(2n+1)\times s} \rightarrow \mathbf{R}^{n+(2n+1)\times s}$$

where n is the state-costate dimension and s the number of interior switching times. The conditions on the switching times reflect the fact that these times are either the beginning or end of a singular arc, hence the switching function and its derivatives must vanish. The conditions on the interior state-costate value here merely ensure the state and costate continuity at the switching times (this is of course much simpler than the state constraints case, where jump conditions have to be handled, as in BNDSCO). We will note these conditions $Switch_{cond}$ and $Match_{cond}$ in the following. Here are the structured shooting function unknown and value layouts.

Unknown: z
- IVP unknown at t_0 (same as in the single shooting method)
- values of (x^i, p^i) at interior switching times t_i , $i = 1..s$
- switching times intervals Δ_i, such that $t_i = t_{i-1} + \Delta_i \quad , \quad \forall i \in [1..s]$

IVP unknown at t_0	(x^1, p^1)	...	(x^s, p^s)	Δ_1	...	Δ_s

Value: $S_{Struct}(z)$
- switching and matching conditions at interior times
- terminal and transversality conditions at t_f (noted TC(t_f), same as in the single shooting method)

$Switch_{cond}(t_1)$	$Match_{cond}(t_1)$	...	$Switch_{cond}(t_s)$	$Match_{cond}(t_s)$	TC(t_f)

Now let us detail the practical implementation of switching and matching conditions. As said before, in our case matching conditions merely check the continuity of the state and costate at the discretization nodes. Hence the following expression:

$$Match_{cond}(t_i) = (x^i, p^i) - (\hat{x}^i, \hat{p}^i) \quad \forall i \in [1..s]$$

where $(\hat{x}^i, \hat{p}^i)$ denotes the state and costate obtained by the integration at the end of the $[t_{i-1}, t_i]$ arc.

As for the switching conditions, we use

$$Switch_{cond}(t_i) = \psi^2(t_i, x^i, p^i, u_i).$$

Remark: it is possible to add the derivative of ψ in this switching condition, with an expression of the form $Switch_{cond}(t_i) = \psi^2(t_i, x^i, p^i, u_i) + \dot{\psi}^2(t_i, x^i, p^i, u_i)$. However, this turns out to be numerically more fragile than the above formulation.

Light structured shooting

We present now a simplified version of the structured shooting formulation, that we call, quite originally, "light structured shooting". This modification stems from the fact that as said before, in the absence of state constraints, the matching conditions merely enforce the state and costate continuity at the beginning/end of singular arcs. So one could want to just pursue the integration at these times with the values of (x, p) obtained after the integration of the previous arc (singular or not). Therefore, it is no longer necessary to add the interior values (x^i, p^i) to the problem unknown. Now only the switching time intervals Δ_i remain, in addition to the usual IVP unknown. The new shooting function is of the form

$$S_{LightStruct} : \mathbf{R}^{n+s} \to \mathbf{R}^{n+s}$$

with s the number of interior switching times.

This lighter formulation does present a gain over the complete one in terms of dimension, however this advantage would be more significant with a higher number of singular arcs. For the two problems we study her, there are only 2 and 1 interior switching times respectively, so there is not much of a difference.

Besides, pursuing the integration all the way also raises a theoretical difficulty. At a singular arc exit, the switching function is supposed to be 0, thus the -non singular- optimal control just after the exit cannot be given properly by the necessary conditions. In practice, of course, ψ only "numerically" vanishes, up to a certain precision. At the exit, we therefore expect a very small, but either positive or negative value. If this sign is correct, then we have the right value of the control at the beginning of the bang-bang arc following the singular arc.

The numerical experiments tend to show that with a good initialization, we are in this favorable case and the light formulation converges well. This highlights the importance of knowing the correct control structure completely, meaning not only the number and localization of the singular arcs, but also the nature of the bang-bang arcs. It is therefore all more interesting that the previous continuation can provide such information, with a quite good approximate solution.

We summarize here the light structured shooting function unknown and value layouts.

Unknown: z
- IVP unknown at t_0 (same as in the single shooting method)
- switching times intervals Δ_i, such that $t_i = t_{i-1} + \Delta_i \quad , \quad \forall i \in [1..s]$

IVP unknown at t_0	Δ_1	...	Δ_s

Value: $S_{LightStruct}(z)$
- switching conditions at interior times
- terminal and transversality conditions at t_f (noted TC(t_f), same as in the single shooting method)

$Switch_{cond}(t_1)$	...	$Switch_{cond}(t_s)$	TC(t_f)

4.4.2 Numerical results

We now proceed with the numerical resolution of the problems. To begin with, we determine the singular structure and initialization from the results

of the two previous continuations, namely single shooting and discretized BVP. Then we solve the problems with the complete and light structured shooting methods presented just above. This resolution does not imply any continuation by itself, we just try to solve the equations $S_{Struct}(z) = 0$ or $S_{LightStruct}(z) = 0$ for both problems (P_1) and (P_2).

Initializations

First we determine the singular structure we expect for each problem, based on the information provided by the two continuations. Here the switching functions clearly indicate the same structure for both continuations: *regular-singular-regular* for Problem 1, and *regular-singular* for Problem 2. Then we extract the required values for the switching times and interior state-costate from the solutions obtained with the continuations, as summarized below. We therefore have two initialization sets (from single shooting and discretized BVP continuations), and two resolution methods (complete and light structured shooting).

Remarks:
• *Finding the initialization corresponding to the single shooting continuation is a bit tedious. In order to find a solution just before the instability threshold, we have in practice to try several path following with refining triangulations.*
• *The initialization for the discretized BVP is taken from the most basic formulation, Euler with an uniform meshsize path following. After all, we only want an initialization, and this is sufficient, as we shall see.*

We summarize here the initializations for Problem 1.
Note: in the following we use the arc lengths Δ_i instead of the switching times t_i, in accordance with the implementation. We recall that we have here $t_1 = \Delta_1$ and $t_2 = \Delta_1 + \Delta_2$.

Problem 1: control structure *regular-singular-regular*
• **Unknown z:** $p(0)$, Δ_1, Δ_2, $x(t_1)$, $p(t_1)$, $x(t_2)$, $p(t_2)$

Continuation	$p(0)$	Δ_1	Δ_2	x^1	p^1	x^2	p^2
$BVP_{0.95}$	-0.429	2.5	4.5	4.996 10^7	-0.600	4.825 10^7	-0.587
BVP_D	-0.453	2.55	4.55	4.839 10^7	-0.637	4.741 10^7	-0.621

We observe that both initializations are rather close, and that for both of them we have $(x^1, p^1) \approx (x^2, p^2)$. This latter point is coherent with the fact that for (P_1), the state and costate are constant over singular arcs (see 4.1.1 page 86).

Now are the initializations for Problem 2.
Problem 2: control structure *regular-singular*
• **Unknown z:** $p(0)$, Δ_1, $x(t_1)$,$p(t_1)$

Continuation	$p(0)$	Δ_1	x^1	p^1
$BVP_{0.925}$	(0.974,1.512)	1.5	$(0.398, -0.309)$	$(0.401, 0.00358)$
BVP_D	(1.167,2.024)	1.429	$(0.578, -0.429)$	$(0.586, 0.000505)$

There are only two arcs and one switching time here, but the state and costate are of dimension 2. The initializations are here more distinct, which is due to the particular "shifting" of the state and costate observed with the Euler formulation on Problem 2, as mentioned before. Besides, the fact that p_2^1 is near 0 in both cases is coherent with the fact that the switching function is supposed to vanish at the beginning of a singular arc (we recall that $\psi = p_2$ for Problem 2, so we expect $p_2(t_1) = 0$).

Resolution

As said before, there is no path following, we directly perform the shoot (at $\lambda = 1$ technically). We use a Runge Kutta 4th order integration, with 1000 steps. In the structured shooting formulations, in case of a fixed step integration, the steps are divided among the arcs proportionally to their lengths (there is no distinction between the regular and singular case). The convergence is immediate (1 second or below) for all test configurations.

Here are the full and light structured shooting details for Problem 1, for both initialization sets:
- Initial norm is the norm of the shooting function at the initial point
- Final norm indicates the shooting function norm at the solution
- objective indicates the objective value at the solution
- H calls indicates the number of shooting function evaluations for the shoot
- time indicates the time taken by the shoot (on a 2.8GHz Pentium 4).

Initialization	Shoot	Initial norm	Final norm	objective	H calls	time
$BVP_{\lambda=0.95}$	Full	$1.08\ 10^{-1}$	$6.70\ 10^{-12}$	$1.069\ 10^{8}$	53	$< 1s$
$BVP_{\lambda=0.95}$	Light	$1.69\ 10^{-1}$	$1.40\ 10^{-12}$	$1.069\ 10^{8}$	59	$< 1s$
BVP_D	Full	$8.44\ 10^{-2}$	$1.17\ 10^{-12}$	$1.069\ 10^{8}$	111	$1s$
BVP_D	Light	$1.13\ 10^{-1}$	$4.47\ 10^{-14}$	$1.069\ 10^{8}$	62	$< 1s$

Structured shooting (full and light) convergence results for Problem 1 Initialization sets from single shooting and discretized BVP continuations

And a more detailed comparison of the shoot solutions:
-p(0) indicates the initial costate value at the solution
-Δ_1, Δ_2 indicate the arc lengths (with $t_1 = \Delta_1, t_2 = \Delta_1 + \Delta_2$)

Initialization	Shoot	$p(0)$	Δ_1	Δ_2
$BVP_{\lambda=0.95}$	Full	$-4.622547\ 10^{-1}$	2.370416	4.618367
$BVP_{\lambda=0.95}$	Light	$-4.622547\ 10^{-1}$	2.370416	4.618367
BVP_D	Full	$-4.622547\ 10^{-1}$	2.370416	4.618367
BVP_D	Light	$-4.622547\ 10^{-1}$	2.370415	4.618366

Structured shooting (full and light) solution comparison for Problem 1
Initialization sets from single shooting and discretized BVP continuations

We see that the norm of the shooting function at the initial point is slightly better for the discretized BVP initialization. As for the four solutions, the final norm is similar for the first three solutions, the fourth one being slightly better. The objective is actually the same up to 11 digits[4], and $p(0), \Delta_1, \Delta_2$ up to 6 digits, except for the fourth solution (5 digits). So we can safely assume that these solutions are numerically identical.

Now we move on to Problem 2.

Initialization	Shoot	Initial norm	Final norm	objective	H calls	time
$BVP_{\lambda=0.925}$	Full	1.37	$3.07\ 10^{-14}$	$3.7699\ 10^{-1}$	52	$< 1s$
$BVP_{\lambda=0.925}$	Light	1.99	$6.85\ 10^{-14}$	$3.7699\ 10^{-1}$	43	$< 1s$
BVP_D	Full	$3.14\ 10^{-1}$	$1.01\ 10^{-15}$	$3.7699\ 10^{-1}$	57	$< 1s$
BVP_D	Light	$1.01\ 10^{-1}$	$1.56\ 10^{-14}$	$3.7699\ 10^{-1}$	49	$< 1s$

Structured shooting (full and light) convergence results for Problem 2
Initialization sets from single shooting and discretized BVP continuations

Initialization	Shoot	$p_1(0)$	$p_2(0)$	Δ_1
$BVP_{\lambda=0.925}$	Full	$9.421734\ 10^{-1}$	1.441910	1.413764
$BVP_{\lambda=0.925}$	Light	$9.421734\ 10^{-1}$	1.441910	1.413764
BVP_D	Full	$9.421734\ 10^{-1}$	1.441910	1.413764
BVP_D	Light	$9.421734\ 10^{-1}$	1.441910	1.413764

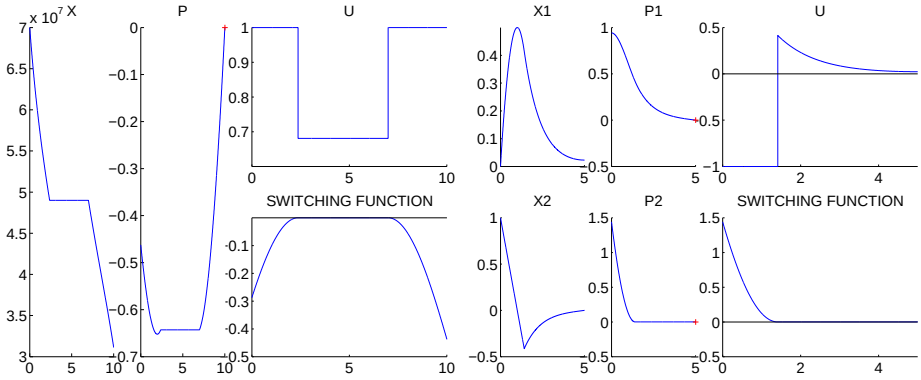
Structured shooting (full and light) solution comparison for Problem 2
Initialization sets from single shooting and discretized BVP continuations

Here we see more clearly the distinction between the two initialization sets, accordingly to the previous remarks. Indeed, the initial norms show a difference of one order of magnitude, in favor of the discretized BVP initialization. As for Problem 1, the final norms are very close, the objectives are identical up to 14 digits[5], and $p(0), \Delta_1$ up to more than 6 digits. Once again, we can assume that we have converged to the same solution.

We note that using the variable step integrators RKF45, DOP853 and ODEX leads to the same solutions. The final norms are slightly better (between 10^{-14} and 10^{-16}), which is not surprising, and the objective value remains the same, up to 8 significant digits. Finally, we represent here the corresponding trajectories, on which the singular arcs are clearly visible.

[4] $1.06905997911\ 10^8$ to be exact.

[5] 0.37699193029464...



Structured shooting solutions - Problem 1 and 2

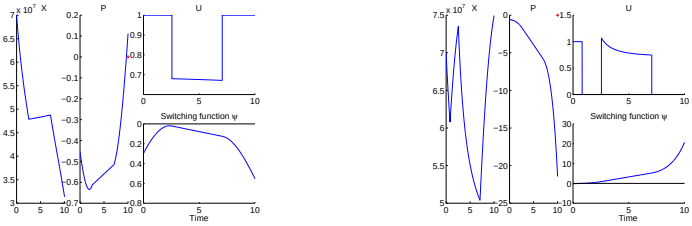
We might wonder about the sensitiveness of the complete and light structured shooting formulations. So we study now the convergence stability by applying some perturbations to the initialization. The reference settings are the following: discretized BVP initialization, RK4 (1000) integration. We use the following perturbations on the initial costate and switching times:
- $p(0)$ by $-1, -0.5, -0.1, +0.1, +0.5, +1$ for Problem 1
- $p(0)$ by $(-2,-2), (-1,-1), (+1,+1), (+2,+2)$ for Problem 2
- (t_1, t_2) by $(-1,+1), (+1,-1)$ for Problem 1
- t_1 by $-1, +1$ for Problem 2

For each shooting attempt we only indicate here the final norm and homotopy calls. The * after the number of homotopy calls indicates that out of bounds values of the singular control (as mentioned on page 110) are encountered during the shoot.

Perturbation	Complete	Light
$p(0) - 1$	$1.07\ 10^{-13}$ / 112*	6.02 / 19*
$p(0) - 0.5$	0.26 / 125*	4.59 / 19*
$p(0) - 0.1$	0.27 / 114*	$9.57\ 10^{-3}$ / 75*
$p(0) + 0.1$	$6.11\ 10^{-14}$ / 87	$1.07\ 10^{-13}$ / 68
$p(0) + 0.5$	$1.28\ 10^{-13}$ / 128	5.68 / 17*
$p(0) + 1$	4.06 / 25*	2.18 / 39*
$(t_1, t_2) + (-1, 1)$	$9.80\ 10^{-13}$ / 60	0.28 / 37
$(t_1, t_2) + (1, -1)$	$1.83\ 10^{-13}$ / 121	$1.37\ 10^{-13}$ / 92

Convergence stability - Problem 1

We observe that the shooting methods are rather sensitive with respect to the initial costate, as a perturbation of -0.1 is enough to prevent the convergence. Actually, this is explained by the fact that this small perturbation is enough to change the control structure, as shown below



The perturbation modifies the control structure - Problem 1

Therefore we are in the same situation as with the simple example in chapter 1: the initial point does not lie in the correct domain, and the shooting does not converge.

We also notice that the full structured shooting manages slightly better than the light one, with 5 acceptable convergence out of 8 tests, versus only 2 (and a half at 10^{-3}...). So it seems that the problem size gain of the light formulation is counterbalanced by a reduced stability. Moreover, we notice the frequent appearance of values of the singular control outside of $[0, 1]$, marked by the * symbol. These seem to coincide quite well with the non convergent cases, with only one exception (first line for Complete formulation) where the shoot manages to converge despite this phenomenon. Technically, it is possible here to truncate these values to 0 or 1 respectively. We have not made it part of the formulation because this case is actually not supposed to happen if we have a good initialization, which is all what the continuation part is about. Indeed, this phenomenon have never occurred during the "normal" structured shooting attempts to solve the problems.

Perturbation	Complete	Light
$p(0) - (2, 2)$	$9.79\ 10^{-14}$ / 58	$2.27\ 10^{-14}$ / 54
$p(0) - (1, 1)$	$1.84\ 10^{-14}$ / 58	$1.26\ 10^{-14}$ / 50
$p(0) + (1, 1)$	$1.97\ 10^{-14}$ / 70	$5.37\ 10^{-15}$ / 58
$p(0) + (2, 2)$	$9.79\ 10^{-15}$ / 73	$3.24\ 10^{-14}$ / 54
$t_1 - 1$	$3.03\ 10^{-15}$ / 76	$2.88\ 10^{-14}$ / 52
$t_1 + 1$	$7.28\ 10^{-15}$ / 69	$4.50\ 10^{-15}$ / 73

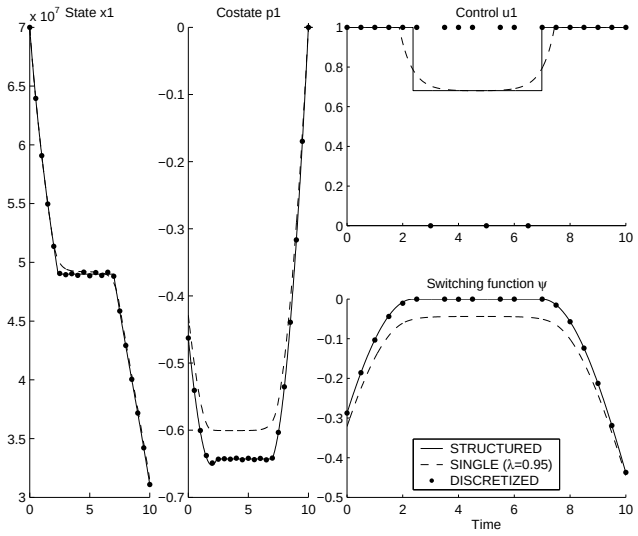
Convergence stability - Problem 2

For Problem 2 we observe that the shooting method is much more robust, as we always converge despite the perturbations. Incidentally, we never encounter any "out of bounds" singular control values, which maybe explains this stability.

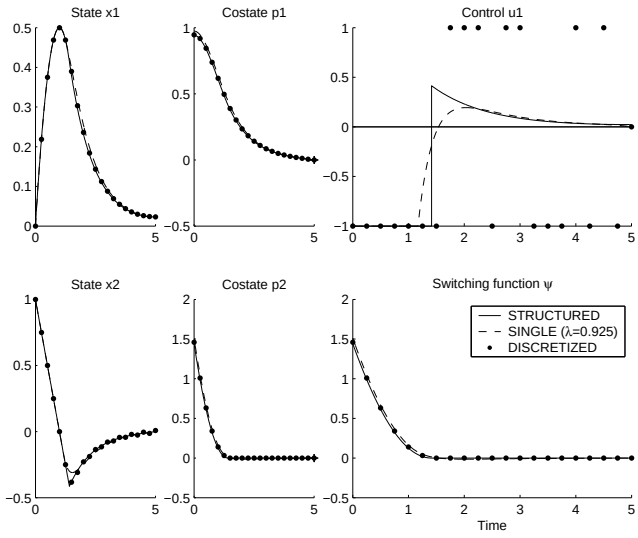
Comparison of the three formulations

To finish with this part, we give here a comparison of the solutions obtained by the single and discretized shooting continuations, with respect to the reference solution from the structured shooting. We take for the single

shooting (dashed line) the solution we have obtained just before the instability threshold, namely for $\lambda = 0.95$ for Problem 1, and $\lambda = 0.925$ for Problem 2. As for the discretized BVP formulation (dots), we represent the "refined" solution corresponding to the implicit trapeze integration, which is the best we have obtained. Then the reference solution from structured shooting is drawn with plain line.



Discretized BVP, Single and Structured shooting solutions for (P_1)



Discretized BVP, Single and Structured shooting solutions for (P_2)

As we can see, the state and costate trajectories from the discretized BVP continuation are quite close to the reference ones. For the single shooting solutions, p is less close to the reference over the singular arc for Problem 1,

but then, we are only at $\lambda = 0.95$ here, so the difference is not that surprising. Actually, we should probably think the other way around, that we are lucky with the good solution at $\lambda = 0.925$ for Problem 2.

This accuracy has probably a lot to do with the easy convergence of the structured shooting, as we have a quite good initial guess. We also observe that the discretized switching function is really close to the reference one, which would confirm that this formulation is well suited to the singular structure detection (and structured shooting initialization).

Concerning the control, the discretized bang-bang values are irrelevant, and merely correspond to the sign of the nearly-0 switching function values over the singular arcs. On the other hand, the control from the single shooting continuation (at $\lambda = 0.95$) is not that bad, being rather close to the reference optimal control. Indeed, we observe that this proximity holds both inside bang-bang arcs, but also inside singular arcs. The differences are actually localized around the switching times (we mean here the *regular/singular* switching times, in the sense of the structured shooting formulation), which is not that surprising.

4.5 Discretized Control formulation

We have seen that the discretized BVP formulation can give quite good approximate solutions. However, it is a bit frustrating that the corresponding control be useless, and indeed even impairs the method by inducing matching errors over singular arcs. So we would like, still in the discretization idea, to obtain an acceptable approximation of the control as well. We take inspiration after the semi direct methods, where the discretized control is the sole unknown (full direct methods discretize both the state and control).

4.5.1 Formulation

In this formulation, the state and costate (x, p) are basically integrated from the initial time t_0, as in the classical IVP of the single shooting method. The difference is that this integration does not use the expression of the optimal control given by the necessary conditions ($u^* = Argmin_{u \in U} \mathcal{H}(t, x, p, u)$ directly. We consider instead a piecewise constant approximation of the control, whose values u^i at the discretization nodes $t_i, i = 1..d$ become additional unknowns. These control values must satisfy some conditions, noted $Control_{cond}$, that are described below. For now, suffice to say that these conditions are of the same dimension as the control u, hence this "shooting function" for the discretized control formulation looks like

$$S_{DC} : \mathbf{R}^{n+m\times d} \to \mathbf{R}^{n+m\times d}$$

with d the number of discretization nodes (and m the dimension of the control). As usual, here is the detailed description of S_{DC}:

Unknown z
- IVP unknown at t_0 (same as in single shooting method)
- values of (u^i) at interior times $t_i,\ i \in [1..d]$ (here t_f is not a node[6])

IVP unknown at t_0	(u^1)	...	(u^d)

Value $S_{DC}(z)$
- control conditions at interior times
- terminal and transversality conditions at t_f (same as single shooting)

$Control_{cond}(t_1)$	$Control_{cond}(t_2)$	...	Conditions at t_f

Note: As the control is generally of lower dimension than the couple (x, p), this formulation can often afford more discretization nodes than the discretized BVP.

Now we come to the core of this formulation, the expression of the control conditions. On a discretization node at t_i, the condition on the control u^i depends on the control structure. If we are over a bang-bang (non singular) arc, then u^i must coincide with the optimal control u_i^* given by the necessary conditions. In the singular case, conversely, we must not enforce the necessary conditions, unless we face the same consequences as in the discretized BVP formulation. However, the control condition should not rely on the detection of the singular case, which typically involves the comparison of the switching function ψ (and optionally of its derivatives) to a certain threshold value. As mentioned previously on page 105, this is numerically very instable depending on the threshold value. So in the end we try the following formulation, which uses the fact that U is an interval $[u_{low}, u_{up}]$ for the two problems:

If $u^i \in U = [u_{low}, u_{up}]$
Then $Control_{cond}(t_i) = (u^i - u_i^*) \times Sing_{cond}(t_i)$
Else if $u^i > u_{up}$, then $Control_{cond}(t_i) = u^i - u_{up}$,
else $Control_{cond}(t_i) = u^i - u_{low}$

with $Sing_{cond} = \psi^2 + \dot{\psi}^2$

The first part translates the fact that we want to enforce the necessary conditions only outside of singular arcs. When $Control_{cond}(t_i) = 0$, we either have $u^i = u_i^*$ and the necessary conditions are satisfied (this should correspond to the bang-bang arcs), or $Sing_{cond}(t_i) = 0$, meaning a singular

[6] As we only use explicit integration formulas, the control at t_f is not needed.

arc. The crucial part is that we do not chose which term vanishes, and never actually force the necessary conditions: if the $Sing_{cond}$ term is close enough to 0, then u^i is allowed to be different from u_i^*. Concerning the expression of $Sing_{cond}$, the formula above with the first derivative of ψ here works better than the simpler formula $Sing_{cond} = \psi^2$ used in the discretized BVP formulation.

Remark: it is actually possible that both terms in the control condition vanish, in which case what numerically happens is not clear. This would correspond to the beginning/end of a singular arc (inside the arc, the correct singular control is supposed different from the incorrect optimal control u_i^*, therefore $u^i - u_i^*$ should not vanish). However, the coincidence of such a time with a discretization node is probably quite unlikely in practice.

The last part, which corresponds to the constraint $u^i \in U$, is such that this constraint violation can take both negative and positive values. It is in practice superior to the maybe more intuitive $Control_{cond}(t_i) = d(u^i, U)$, possibly because of this sign property. Of course, in case the control is of dimension greater than 1, this part has to be adapted, but it is uneasy to predict how to keep the stability of the above formulation.

Remark: as with the discretized BVP formulation, we have tried to use the algebraic expression of the singular control. The idea is to enforce a constraint of the form "$u^i = u^i_{singular}$ when t_i is inside a singular arc" as part of the control conditions. However, we have encountered the same difficulties as before, namely an unacceptable sensitiveness with respect to the practical implementation of the singular case detection.

Before we move on to the numerical results, we would like to emphasize that the convergence properties for this discretized control formulation are still an open question.

4.5.2 Numerical results

We apply the same continuation as with the single shooting and discretized BVP. As the control is of dimension 1 in both cases, we use 50 discretization nodes for the two problems (with a total dimension of 51 and 52 respectively). We observe a rather fast[7] convergence without any particular difficulties, except the fact that the first junction at $\lambda = 0$ curiously takes a large number of simplices, especially for Problem 2 (see the tables below).

[7] depending on the integrator, from 15 to 30 seconds for Problem 1, and 10 to 20 seconds for Problem 2.

The path following uses here for both problems a basic $K_1(10^{-1})$ triangulation, with normal scaling, and a shooting attempt both at $\lambda = 0$ and $\lambda = 1$. The initialization before the first shoot is extremely rough: we set $p(0) = 0$ for (P_1) and $p(0) = (1, 1)$ for (P_2), with the control values $u^i = 0,\ i = 1..d$ in both cases. The initial shoot converges nevertheless almost instantly to a solution with a norm around 10^{-15} in all cases, with the exception of the Runge Kutta 2 integration for (P_2), where the norm is only around 10^{-4} (which does not seem to affect the following).

We detail here the path followings:
- **Main path** indicates the number of simplices for the main path following.
- **1st Junction** indicates the number of simplices for the first junction.
- **H calls** indicates the global number of homotopy evaluations.
- **Final norm** indicates the norm of the solution after the shoot at $\lambda = 1$.

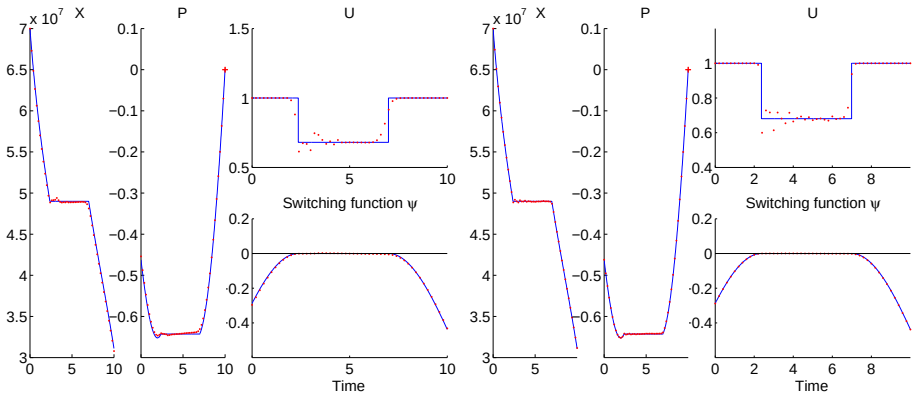
Integrator	Main path	1st Junction	H calls	Final norm
Euler	23 209	3499	26 268	$5.39\ 10^{-5}$
Midpoint	24 149	6474	29 071	$6.87\ 10^{-6}$
RK2	24 186	6464	29 289	$4.43\ 10^{-6}$
RK4	24 173	6488	28 974	$3.88\ 10^{-6}$

Problem 1 - Path following for the Discretized Control Formulation.

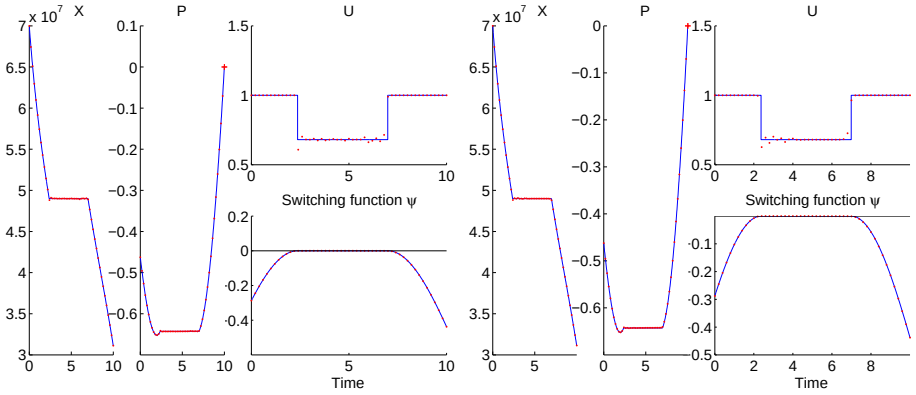
Integrator	Main path	1st Junction	H calls	Final norm
Euler	5071	27 033	23 792	$4.03\ 10^{-3}$
Midpoint	5144	31 336	27 459	$1.05\ 10^{-2}$
RK2	4678	42 142	33 568	$6.57\ 10^{-4}$
RK4	5150	31 396	28 158	$5.58\ 10^{-4}$

Problem 2 - Path following for the Discretized Control Formulation.

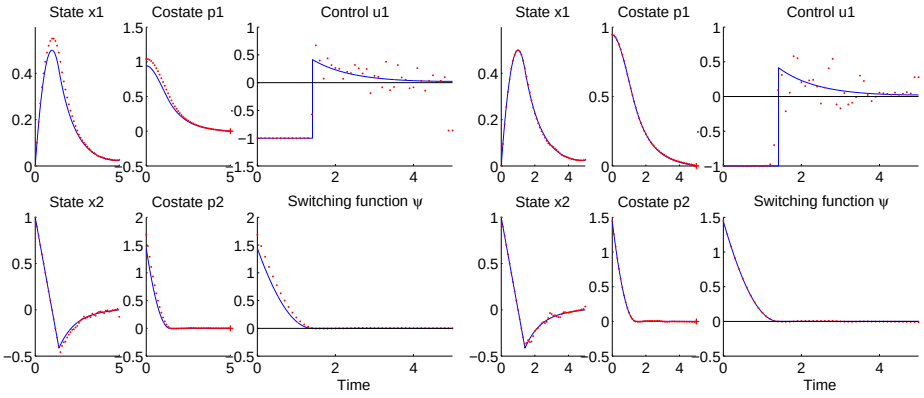
Here are the corresponding trajectories for various integration choices, namely Euler, explicit midpoint, and basic Runge-Kutta of 2nd and 4th order. Once again, the discretized solutions are plotted over the structured shooting reference in solid line for the sake of comparison.



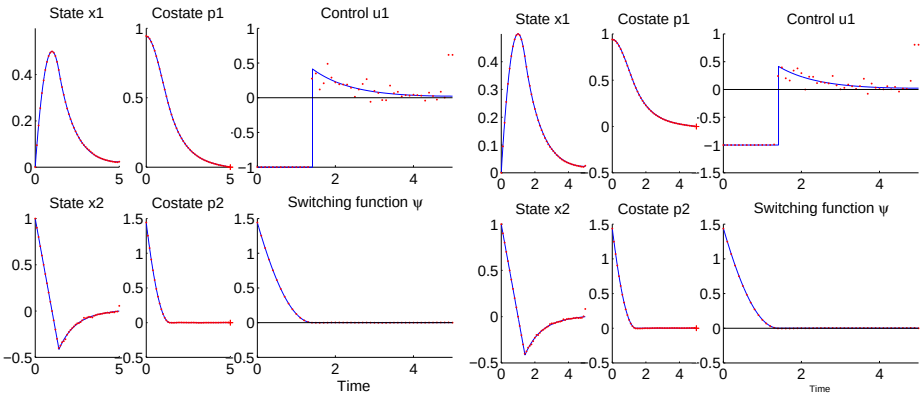
Discretized control for Problem 1 - Euler and Midpoint



Discretized control for Problem 1 - RK2 and RK4



Discretized control for Problem 2 - Euler and Midpoint



Discretized control for Problem 2 - RK2 and RK4

The first constatation is that for both problems and all integrators, the approximation of the state and costate is surprisingly good. An exception is the Euler case for Problem 2, which displays the same kind of "shifting" from the solution mentioned on page 102 for the discretized BVP formulation. The discretized switching function ψ is also quite accurate, and clearly

indicates the singular structure for both problems.

Yet the most interesting point is the approximation of the control, which is the unknown in this formulation. In all cases the discretized control matches the reference over the bang-bang arcs, which tends to indicate that the necessary conditions are taken into account properly. Then over the singular arcs, we observe that the control is not that far from the reference singular control, especially with the two Runge Kutta integrators. This means that the $\psi^2 + \dot{\psi}^2$ term in the control conditions indeed manages to guide the control towards the correct singular value, even though we do not use the expression of $u_{singular}$ at all[8]! This is an interesting point, as it would suggest that this method might be applied to problems where the algebraic expression of the singular control is not available.

4.6 Direct method approach

We would like here to make a quick comparison with the direct methods, which basically discretize the problem. In their case the control is part of the unknowns, and is not derived from necessary conditions, so it is interesting to see how they handle the singular arcs.

4.6.1 Formulation

We use here for our direct experimentations the software KNITRO, that solves nonlinear optimization problems (NLP). Detailed information about the algorithms used can be found in [13, 14], we just recall here the general principle of the method. Basically, KNITRO handles the resolution of the original NLP problem by solving a sequence of barrier subproblems. Each subproblem is solved with a low precision by SQP iterations (with the use of trust regions), until the barrier parameter is nullified. This is an interior point method, which share some ties with continuation or penalty methods.

The transformation of the optimal control problem into a nonlinear optimization involves the discretization of the state and control, who become the unknown of the problem. The integration of this discretized system introduces a set of constraints related to the dynamics, as the value of the state at a discretization node must match the value provided by the integration from the last node.

[8] we recall that for the two problems studied, the expression of the singular control is obtained from the second order equation $\ddot{\psi} = 0$, as the control vanishes from $\dot{\psi} = 0$.

Note: a variant from this direct approach is to discretize the control only, and merely integrate the state from the beginning. This "semi direct" approach allows an important gain in terms of problem size, as the state is not part of the unknown anymore and the dynamics constraints vanish.

Concerning the integration, we limit here ourselves to fixed step schemes, as we want to be able to use automatic differentiation to compute the derivatives of the dynamics constraints and integrated objective.

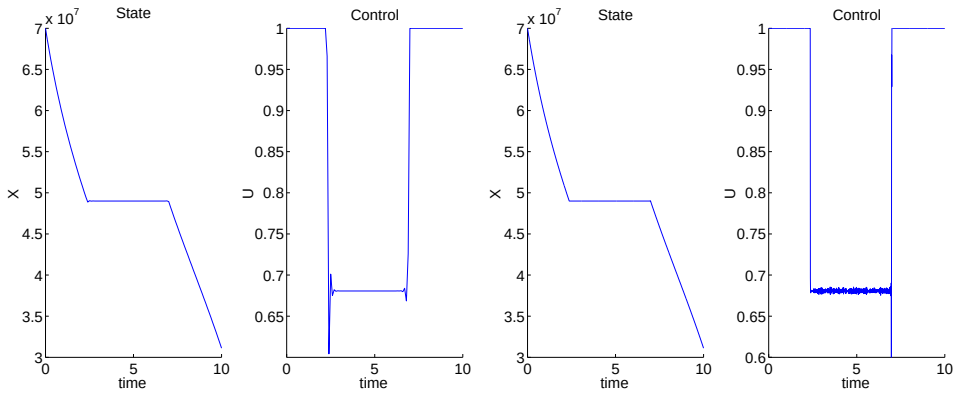
Remark: there is no costate as such in direct methods, even if it can be linked to the Lagrange multipliers of the discretized problem.

4.6.2 Numerical results

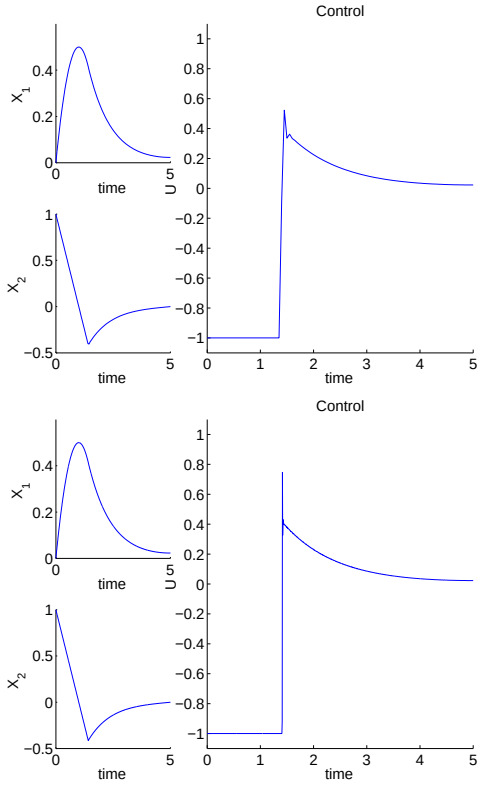
The integration for both problems is a basic 4th order Runge Kutta rule, with 100 and 1000 steps. The feasibility and optimality tolerances are set to 10^{-12} and 10^{-9} respectively. We use the following trivial initialization, in the same spirit as for our continuation approach: $x^i = x_0$, and $u^i = 0$, $\forall i = 1..ns$, with ns the number of discretization steps.

With these settings, the execution times are of the same order as our method:
- 2 seconds (100 steps) and 81 seconds (1000 steps) for Problem 1
- 26 seconds (100 steps) and 237 seconds (1000 steps) for Problem 2

Here are the solutions (state and control) for 100 and 1000 steps:



Direct method - Solutions with 100 and 1000 nodes for Problem 1



Direct method - Solutions with 100 and 1000 nodes for Problem 2

We converge obviously to the same solutions that we have found with the structured shooting. However, we observe some irregularities over the singular arcs, and especially at the switching times. The few tests we have made would indicate that getting rid of these artifacts completely (to obtain a "smooth" solution as with the structured shooting) might not be easy. Increasing the number of steps is soon limited by the execution times, and is not always convincing. For instance, see Problem 1 with 1000 steps, where the oscillations on the singular arc have actually increased compared to the 100 steps formulation. On the other hand, setting more strict tolerances than the ones above rapidly makes the convergence much more difficult to attain.

However, we must insist on the fact that our experiments with KNITRO are very preliminary, so there are probably better settings to be found. Another way that should be explored is the semi direct approach, in which the control only is the unknown. Finally, we may also be limited by the choice of a fixed step integration, which follows from the use of automatic differentiation.

Conclusion

The first continuation with the single shooting shows the progressive apparition of the singular structure, by the evolution of the control and switching function as λ increases along the path. However, near the convergence the problems become nearly singular, and the singular structure is lost due to the numerical instability of the switching function near 0.

To circumvent this difficulty, we discretize the equations of the BVP, the values of the state and costate at the discretization nodes becoming part of the homotopy unknown. We then observe on the numerical experiments an indication of a flaw in this formulation, related to the incorrect application of the necessary conditions over the singular arcs. This numerically leads to persistent matching errors and some irregularities of the discretized state and costate. Nevertheless, a good choice of the integration formula, combined with the adaptive meshsize and solution refinement mechanisms described in chapter II, allows us to obtain a quite accurate approximation of the state, costate and switching function. We have therefore a clear detection of the singular structure, as well as a good initialization for a precise resolution.

It is worth noting that for these two continuations, the simplicial algorithm manages to reach the convergence at $\lambda = 1$, despite the presence of singular arcs.

Then we move on to the precise resolution, with a method adapted from multiple shooting to the singular case. Thanks to the information provided by the previous continuation, we are able to converge instantly to an accurate solution for both problems (coherent with the results presented in [34, 19]).

So the problems are solved, yet the very good results of the discretized formulation concerning the state and costate encourage us to try to obtain an approximation of the singular control as well.

We introduce then a new discretized formulation, in which the control replaces the state and costate in the unknown. We try to enforce the necessary conditions on this control, while respecting the singular structure of the problem to avoid the pitfalls of the discretized BVP. Although very sensitive to the practical implementation of these conditions on the control, this idea gives surprising numerical results. Indeed we obtain for both problems a very close approximation of the state and costate, and most interestingly good control values as well, including over the singular arcs, without using the algebraic expression of the singular control. However, we lack theoretical

results concerning this formulation, and the handling of a control of dimension greater than 1 would likely require a careful implementation. Also, the conditions on the control can probably be improved for more stability, maybe by using a barrier.

To finish with, it is worth noting that all the approaches described in this chapter are rather fast, with convergence times not exceeding 2 minutes (for the refined discretized BVP continuation). In particular, the complete resolution of the problems (basic discretized BVP continuation plus structured shooting) takes less than 15 seconds.

Conclusions and Perspectives

The central theme of this work is related to the difficulties, especially the critical choice of the initial point, encountered by shooting methods for problems with a low regularity. In order to deal with these, we introduce a continuation approach, and choose in practice to use simplicial methods due to their robustness. In this context, some additions have been made to the simplicial algorithm, such as adaptive meshsize and solution refining.

We first consider a low thrust orbital transfer problem family, involving a bang-bang control structure with several hundreds of commutations. The simplicial method leads to the same results that have already been obtained by the differential continuation (but was slower, as expected), and the comparison of three variable step integrators highlights again some conservation properties related to the maximal thrust. Besides, these experiments validate the qualitative behaviour of the adaptive meshsize mechanism, with both a gain in the simplices followed and a faster final shoot. However, they also reveal the significant numerical cost of the adaptive junctions, leading to a mixed quantitative performance.

Then, the second part of the experiments is devoted to the study of two singular arcs problems in parallel. The continuations based on the single shooting and discretized boundary value problem formulations provide us with a reliable knowledge of the control structure, both singular and bang-bang arcs. We also obtain a good approximate solution, which allow us to initialize properly the precise resolution of the problems, performed by methods derived from multiple shooting. Besides, we also introduce a new formulation inspired by the direct methods, in which the discretized control is part of the unknown. The first numerical results are interesting, even though the practical implementation requires some care and the theoretical properties of this formulation remain an open question.

Finally, the future perspectives of this work include the study of state constraints problems, that are also difficult to solve by indirect methods with no a priori assumptions on the structure. In particular, we note that some interesting results have already been obtained by J. Laurent-Varin with an

interior point method applied to a symplectic Runge Kutta discretization of the optimality conditions, see [6].

Concerning the simplicial algorithm, a first objective would be to improve the numerical cost of the adaptive meshsize, either with a better junction formulation, or by finding another robust (and cheaper) way of performing the meshsize changes. Some additional directions to explore would be other triangulations, labelings (integer labeling for instance), integrators (symplectic integrators in particular), and nonlinear solvers. Also, we would like to make a more in-depth comparison with the direct methods on the studied problems.

Appendix A

Simplicial package overview

This package was used to conduct all the numerical experiments of this work, except the preliminary tests for the direct methods (made with Knitro[1]). It implements a simplicial method adapted to resolution of optimal control problems, as described in chapter 2, and includes the various formulations presented in chapter 4 for the singular arcs problems. The complete package (source, Matlab scripts, sample problems and user guide) is available online.

Web page: **http://www.enseeiht.fr/lima/apo/simplicial**
Contact: **pierre.martinon@enseeiht.fr**

The code itself is written in Fortran9x (about 7000 lines, standard F90 mostly), and uses some Fortran77 third party codes: the nonlinear solver for the shooting methods, the three variable step integrators, and matrix inversion subroutines. The Simplicial package should contain the following files:

User guide

Simplicial core files

- *Simplicial.f90*: main program
- *Levels.f90*: path following module
- *Cell.f90*: simplices manipulation
- *Homotopy.f90*: homotopy interface
- *Shoot.f90*: shooting functions module
- *Integrators.f90*: integrators interface
- *Defs.f90*: global variables
- *Makefile*

[1] see www.ziena.com/knitro.html

Sample problem files

- *ProblemFuns.f90*: user supplied problem specific subroutines
- *Problem.cfg*: sample configuration file
- *Problem.in*: sample initialization file

Matlab scripts

- *init.m*: problem file generation
- *config.m*: configuration file generation
- *sol.m*: solution visualization
- *path.m*: zero path visualization
- *simpath.m*: zero path simplices visualization
- *valley.m*: path valley visualization
- *plateau.m*: homotopy plateau visualization
- *control.m*: control and switching function evolution for several solutions

Third party codes

- *auxsubs.f*: miscellaneous auxiliary subroutines
- *inverse.f*: matrix inversion (LU factorization)
- *rkf45.f*: Runge Kutta Fehlberg 4-5th order integrator
- *dop853.f*: Dormand Prince integrator
- *odex.f*: Gragg Bulirsch Stoer extrapolation integrator

- Nonlinear solver: Quasi Newton (Powell method) solver (MINPACK)
  Source: hybrd.f, by B.S. Garbow, K.E. Hillstrom and J.J More
  http://netlib.bell-labs.com/netlib/minpack/hybrd.f.gz

- Integrator: Runge Kutta Fehlberg (4,5) method
  Source: rkf45.f, by H.A. Watts and L.F. Shampine
  http://netlib.bell-labs.com/netlib/ode/rkf45.f.gz

- Integrator: Dormand Prince (8,5-3) method
  Source: dop853.f, by E. Hairer and G. Wanner
  http://elib.zib.de/pub/elib/hairer-wanner/nonstiff/dop853.f

- Integrator: Gragg Bulirsch Stoer extrapolation method
  Source: odex.f, by E. Hairer and G. Wanner
  http://elib.zib.de/pub/elib/hairer-wanner/nonstiff/odex.f

- Matrix inversion: via LU factorization (LAPACK-BLAS)
  Source: dgetri.f, dgetrf.f
  http://netlib.bell-labs.com/netlib/lapack/double/dgetr{i,f}.f.gz

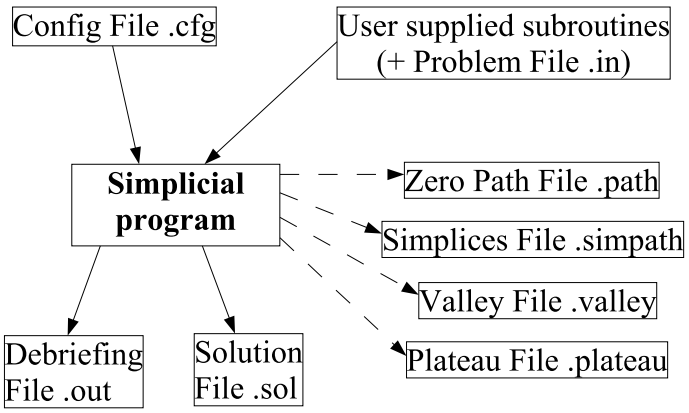
A.1 Using the Simplicial package

In order to solve an optimal control problem with the Simplicial package, the user has to complete the following subroutines, located in *ProblemFuns.f90*:
• **InitPar**: performs specific problem initializations, if any.
• **Control**: optimal control and switching function evaluation.
• **Dynamics**: state, costate and objective dynamics.

Then for each problem, two input files are used as well:
• **Problem initialization file .in**: contains problem data.
• **Configuration file .cfg**: contains settings for the simplicial algorithm.

After execution, the Simplicial code will produce various output files:
• **Debriefing file .out**: details about the execution.
• **Solution file .sol**: state, costate, control (and switching function) generated at the solution (visualization with the Matlab script *sol.m*).
• Optional output files: **.path**, **.simpath**, **.valley**, **.plateau** (see page 141 for more details).

Here is an overview of the various input and output files used by the Simplicial package, which are described more in details below.



Note: for each problem, a single prefix name is associated to all input and output files. Thus if input files are named *Demo1.in* and *Demo1.cfg* for instance, the corresponding output files will be *Demo1.out* and *Demo1.sol* (and so on for optional output files). This makes it easier to keep track of which problem the files correspond to.

A.1.1 User supplied subroutines

These three subroutines are located in the file *ProblemFuns.f90*, specific to each problem.

InitPar

This subroutine performs the various initializations required by the problem. It is called at the beginning of the program, after the initialization file .in has been read, and then each time the homotopy is computed.

The input variables are
- *mode*: 0 for first call, 1 for subsequent calls
- *lambda*: homotopic parameter used for continuation

Subroutine interface

```
Subroutine InitPar(mode, lambda)
  implicit none

  integer, intent(in) :: mode
  real(kind=8), intent(in) :: lambda
```

Control

This subroutine provides the value of the optimal control, according to necessary conditions. The switching function is also typically evaluated, but this is not mandatory.

The input variables are
- *lambda*: homotopic parameter used for continuation
- t: time
- x: state (dimension ns)
- p: costate (dimension nc)

And the output variables
- u: optimal control (dimension m)
- *psi*: optional switching function and its derivatives (total dimension *dimpsi*)

Subroutine interface

```
Subroutine Control(lambda,t,x,p,u,psi)
  implicit none

  real(kind=8), intent(in) ::  lambda, t
  real(kind=8), intent(in), dimension(ns) :: x
  real(kind=8), intent(in), dimension(nc) :: p
  real(kind=8), intent(out), dimension(m) :: u
  real(kind=8), intent(out), dimension(dimpsi) :: psi
```

Dynamics

This subroutine provides the dynamics for the state, costate, and objective if needed.

The input variables are
- *dimphi*: dynamics dimension (> ns+nc when objective is required)
- *lambda*: homotopic parameter used for continuation
- t: time
- x: state (dimension ns)
- p: costate (dimension nc)
- u: the optimal control (dimension m)

And the output variable
- *phi*: dynamics (state, then costate, and optionally objective)

Subroutine interface

```
Subroutine Dynamics(dimphi,lambda,t,x,p,u,phi)
  implicit none

  integer, intent(in) :: dimphi
  real(kind=8), intent(in) :: lambda, t
  real(kind=8), intent(in), dimension(ns) :: x
  real(kind=8), intent(in), dimension(nc) :: p
  real(kind=8), intent(in), dimension(m) :: u
  real(kind=8), intent(out), dimension(dimphi) :: phi
```

A.1.2 Input files

A.1.3 Initialization file .in

This file contains problem initializations, such as dimensions, initial and terminal conditions, initial guess, and integrator choices. Configuration files can either be edited manually, or automatically generated by the *init.m* script. Here is a brief summary of the parameters set in this file.

- *Homotopy choice and problem class*
  Integer [1+]: tells which homotopy is to be used by the program. Set to 1 by default, this permits to use several different homotopies for a given problem with the same *ProblemFuns.f90*.
  Integer [0..3]: sets the kind of shooting method used (see page 145).
  0: Discretized shooting
  1: Single shooting
  2: Structured shooting
  3: Light Structured shooting
  Note: The homotopy choice and problem class values given in the .cfg and .in files must match.

- *Unknown, State, Costate and Control dimensions*
  Integer Integer Integer Integer: here are specified the dimensions

of the shooting function unknown, and of the state, costate and control of the problem (the state and costate dimensions are usually the same, but can differ if some components are known to be constant, and therefore do not need to be integrated).

- *Objective and Switch dimension*
  Integer [1+]: usually set to 1, as the objective is scalar, but additional objectives can also be computed, if the corresponding derivatives are included in subroutine **Dynamics**.
  Integer [0+]: like the objective, the switching function has scalar values, but its derivatives can also be computed, if set in subroutine **Control** (observing the switching function is especially useful when suspecting the presence of singular arcs).

- *Number of IVP unknown values*
  Integer: dimension of IVP unknown.
  IVP unknown indices
  Integer(): indices of IVP unknown values in $y = (x, p)$ vector.

- *Number of initial values*
  Integer: number of values known at t_0 (given by initial and transversality conditions).
  Initial values indices
  Integer(): indices of these values in $(x(t_0), p(t_0))$.
  Initial values
  Real(): initial values.

- *Number of terminal values*
  Integer: number of values known at t_f (given by terminal and transversality conditions).
  Terminal values indices
  Integer(): indices of these values in $(x(t_f), p(t_f))$.
  Terminal values
  Real(): terminal values.

- *Initial and final time*
  Real Real: the initial and final times t_0 and t_f

- *Starting point (z0,lambda0) for zeropath*
  Real(n+1): initial guess z_0 for shooting function unknown, and ini-

tial value λ_0 for homotopic parameter λ (λ_0 is typically 0).

- *Scaling mode*
  Integer [0..1]: selects the scaling mode.
  0: no scaling
  1: standard scaling
  2: soft scaling (to $[0.01, 10]$ instead of $[0.1, 1]$)

- *Path and solution integrator choice*
  Integer [0..7]: selects the integrator used for path following and solution (cf page 146)
  0: Fixed step - Euler
  1: Fixed step - Midpoint
  2: Fixed step - Runge Kutta 2nd order
  3: Fixed step - Runge Kutta 3rd order
  4: Fixed step - Runge Kutta 4th order
  5: Variable step - Runge Kutta Fehlberg 4-5th order (**rkf45**)
  6: Variable step - Dormand Prince 8-5-3 (**dop853**)
  7: Variable step - Gragg Bulirsch Stoer extrapolation method (**odex**)

- *Steps for fixed step integrators*
  Integer Integer: sets the number of steps for fixed steps integrators (path and solution).
  Variable step integrator abserr and relerr
  Real Real Real Real: absolute and relative error tolerances for variable step integrators (idem).
  Fixed point minimal progress and maximal iterations
  Real Integer: sets the stopping criterion for the fixed point iterations.

- *Number of parameters*
  Integer [0+]: number of problem specific parameters.
  Parameters
  Real(): problem specific parameters, if any.

A.1.4 Configuration file .cfg

Here we find a list of parameters for the simplicial algorithm. Configuration files can either be edited manually, or automatically generated by the *config.m* script. Here is a brief summary of the parameters set in this file.

- *Homotopy choice and problem class*
  Integer [1+]: tells which homotopy is to be used by the program. Set to 1 by default, this permits to use several different homotopies for a given problem with the same *ProblemFuns.f90*.
  Integer [0..3]: sets the kind of shooting method used (see page 145).
  0: Discretized BVP
  1: Single shooting
  2: Structured shooting
  3: Light Structured shooting
  4: Discretized Control
  Note: The homotopy choice and problem class values given in the .cfg and .in files must match.

- *Triangulation choice*
  Integer [1..5]: selects the triangulation used (cf 2.1).
  1: Freundenthal's uniform K_1[1]
  2: Todd's uniform J_1[1]
  3: Todd's refining J_3
  4: Alternate refining J_4
  5: Dang's uniform D_1 (cf [21]) (experimental)

- *Labeling choice*
  Integer [0..1]: selects the labeling used.
  0: Traditional vector labeling by the homotopy
  1: Integer labeling (experimental, cf page 145)

- *Zero Path follow mode*
  Integer [-1..3]: sets the adaptive meshsize mode (cf 2.3).
  -1: do not perform level checks at all
  0: perform only level checks, no adaptive meshsize
  1: adaptive meshsize, deviation control only
  2: adaptive meshsize, anisotropic refinement only
  3: full adaptive meshsize

- *Initial and final solver calls*
  Integer [0..1] Integer [0..1]: selects whether solver calls are made at the beginning / end of zero path.
  0: no solver call

[1]Note: we use for the K_1 and J_1 the "interchange permutations" pivot rules, instead of the "reflexion" pivot rules (see [2], 13.3), as they were slightly better in terms of cumulative simplex errors.

1: solver call

- *Refinement mode and maximum refinement attempts*
  Integer [0..3]: solution refinement method used (see 2.4).
  0: no refinement
  1: J_3 refinement between refinement bound and upper bound
  2: J_4 refinement between refinement bound and upper bound
  3: Merril-like junctions refinement at upper bound
  Integer [0+]: number of solution refinement attempts (for mode 3).

- *Deviation tolerance*
  Real: tolerance for adaptive meshsize (cf 2.3).
  Anisotropic thresholds
  Real Real Real: thresholds for adaptive meshsize (cf 2.3).

- *Maximum simplices for main and junction homotopies, check frequency*
  Integer Integer: maximum number of simplices allowed for main and junction homotopies. *Junction homotopies (see page 32 for more details) take place to find starting completely labeled faces, or when changing the triangulation meshsize, and should not require more than 1000 or 10000 simplices (quick junctions can often be completed in less than 100 simplices). In case of difficulties at the starting point, or when using solution refinement, however, longer junctions may occur, and the maximum number of simplices should be increased accordingly.*

  Integer: this parameter sets the number of simplices after which the current simplex and labeling are periodically checked and reset in order to limit the accumulation of roundoff errors. Set to 0 to disable checks (usual value is 10000 or 100000).

- *Initial triangulation meshsize for (z,lambda)*
  Real Real: size vector δ for initial meshsize, given by $\delta(i), i = 1, n$ and $\delta(n+1)$. (Note: the second value is meaningless for J_3 triangulation, as the stepsize with respect to the homotopic parameter λ is here predetermined.)

- *Bounds for homotopic parameter lambda*
  Real Real Real: lower, refinement (between lower and upper) and upper bounds for homotopic parameter lambda. Lower and upper bounds are often set to 0 and 1, but this is not mandatory. Actually,

the algorithm works internally with a homotopic parameter $\lambda \in [0, 1]$, which is re-parametrized to [lower bound, upper bound] for actual homotopy evaluation. These values can be used to change the beginning and end of the zero path, for instance to split the homotopy into several passes.

- *Zero Path, Simplices, Valley and Plateau output files*
  Integer[0..1] Integer[0..1] Integer[0..1] Integer[0..1]: selects whether output files .path, .simpath, .valley and .plateau are to be generated (see page 141 for details).
  0: no output file generated
  1: output file generation

- *Ratio for saved paths*
  Integer Integer: selects the ratio for saved points/simplices in .path/.simpath files, for main and refinement homotopy. Set to 1 to save full paths (*caution: this can lead to huge output files if the number of simplices followed is high...*).

- *Homotopy Norm and Criterion output in Path*
  Integer[0-1] Integer[0-1]: see page 141 for description.
  0: do not include homotopy norm/criterion in zero path file
  1: include homotopy norm/criterion in zero path file
  (*these two options require that Zero Path output file be set to 1.*)

- *Valley range and step*
  Integer Real: see page 141 for description.

- *Plateau range and step*
  Integer Real: see page 141 for description.

- *Solution generation at final face vertices, barycentric control*
  Integer[0-1]:
  0: do not generate solutions at final face vertices
  1: generate .vertxx.sol solution files for final face vertices
  Integer[0-1]: (note: barycentric control is experimental, and on its way to oblivion...)
  0: do not generate barycentric control
  1: reintegrate solution with barycentric control

Warning: some options are not compatibles, and the code will check for inconsistencies in the input files. In case some conflicting options are found, a warning is issued and changes are made so that the conflict is solved. Nevertheless, in this case the input files should be corrected manually, as the automatic changes made by the code may not match the user's wishes...

A.1.5 Output files

Here is a summary of the output files, that can be visualized with the provided Matlab scripts.

Execution and debriefing file

Once user-supplied subroutines and input files are completed, just run the executable **Simplicial**. The program will ask the name (without the extension) of the input files, for instance "Problem1" for *Problem1.cfg* and *Problem1.in*. Keep in mind that all input and output files associated to a given problem have the same name, only with different extensions. Thus it is quite easy to keep track of several problems at the same time, each one having its set of files with the same prefix.

At the end of the execution, a debriefing file **.out** file is created, which contains some informations about the continuation, as well as a copy of the input files used for the run. A solution file **.sol** is also generated (see below), as well as some other optional output files if specified in the **.cfg** file.

Solution file

The **.sol** output file contains the solution generated at the end of the continuation. The script *sol.m* traces the state, costate and control with respect to time, as well as the switching function optionally.

Zero Path file

The **.path** output file contains the PL approximation of the followed zero path. The ratio (total simplices / saved zeros) is set in the .cfg file, and should be set greater than 1 if a large number of simplices is expected. The path can be traced by the script *path.m*, which displays the evolution of each component of z with respect to λ. Optionally, the norm of the homotopy and the objective value along the path can also be shown, if these options are set to 1 (possibly greater than 1 for the objective).

Simplices Path file

The **.simpath** output file contains the transverses simplices followed, which can be drawn by the script *simpath.m* (for dimension 2 or 3).

Note: this file can become quite huge if there are a lot of simplices followed, so the saved path ratio should be set accordingly.

Homotopy Valley file

This file can be used to visualize the norm of the homotopy around the followed path, with a certain deviation along each component of the path. For each dimension $j \in [1..n]$, for each point z of the zero path, the norm of the homotopy will be computed at *range* points on both sides of z, with a stepsize equal to *step* (the parameters *step* and *range* are set in the configuration file):

Note: same remark here, especially with large simplices/range values.

Homotopy Plateau file

In dimension 1 or 2, the "plateau" output file shows the value and norm of the homotopy around the solution at the convergence. More precisely, it will contain the value and norm of the homotopy on grid points around the solution z^*. The total grid width is $1 + 2 \times range$, with a stepsize between points equal to *step* (*range* and *step* are set in the configuration file).

A.2 General code structure

The Simplicial code organization respects a layer structure, from the simplicial algorithm to the user supplied subroutines for homotopy evaluation.

Layer 1: Simplicial module

The first layer contains the main program, and basically implements the simplicial algorithm, along with some input and output files manipulations.

File name: **Simplicial.f90** Main program: **Simplicial** Module name: **Simplicialmod**, *uses* **Levelsmod** (Layer 2)

Main Subroutines

- *Simplicial*: main program
- *Start*: first simplex and labeled face (optional first junction)
- *Follow*: zero path following
- *Refine*: solution refinement (via homotopy)
- *Solve*: solution refinement (via Powell solver call)

Layer 2: Levels module

The second layer contains the subroutines related to junction homotopies, and adaptive meshsize refinement, as well as some data scaling.

File name: **Levels.f90** Module name: **Levelsmod**, *uses* **Cellmod** (Layer 3)

Main Subroutines

- *Junction*: performs junction homotopy (usually to new meshsize)
- *NewMesh*: computes the new meshsize for the triangulation
- *DataScale*: data scaling, initial or dynamic

Layer 3: Cell module

The third layer implements the cell manipulations, such as simplex construction and pivoting, labeling operations, and next simplex choice (lexicographic test).

File name: **Cell.f90** Module name: **Cellmod**, *uses* **Hmod** (Layer 4)

Main Subroutines

- *FirstSimplex*: builds first transverse simplex.
- *FirstLabel*: computes labeling for the first simplex.
- *Lex*: lexicographic test, determines the exit (completely labeled) face of the current simplex.
- *Piv*: pivoting rules for next simplex.

Third party code: inversion subroutines **dgetri,dgetrf** (in *inverse.f*).

Layer 4: Homotopy module

The fourth and last layer deals with the homotopy value computation. In the current implementation, the homotopy is actually a shooting function related to an optimal control problem, so this layer is merely an interface to the Shooting function module described below.

File name: **Homotopy.f90** Module name: **Hmod**, *uses* **GenFuns**

Main Subroutines

- *Homotopy*: evaluates homotopy H at given point (x, λ)
- *InitJunction*: junction homotopy initialization

Layer 4bis: Shooting function

This implementation of the fourth layer in the case of shooting functions is divided into two separate files, so that all user supplied subroutines are in the same place.

Generic functions: common to all problems

File name: **Shoot.f90**
Module name: **GenFuns**, *uses* **SpecFuns**

Main subroutines

- *Hom*: interface for homotopy evaluation
- *IVP*: Shooting function evaluation
- *Single, Structured, LightStructured, Discrete*: shooting methods
- *Hyb*: interface for quasi newton solver (**hybrd**)
- *Sol*: solution file generation (.sol)

Problem-specific functions, user supplied

File name: [**Problem**]**Funs.f90**
Module name: **SpecFuns**

Main subroutines

- *InitPar*: initializations
- *Control*: computes optimal control
- *Dynamics*: state, costate and objective dynamics

Appendix B

Main options

We recall here the main options of the Simplicial package, without going into the details (we refer the interested reader to the User Guide[1] for a more complete description of the available options).

B.1 Problem class

This setting indicates the general problem formulation, among the five implemented at the moment:

0: Discretized BVP: introduced in chapter 4 for the singular arcs problems. Can provide an approximate solution to initialize the Structured shooting below.

1: Single shooting: the standard one.

2: Structured shooting: introduced in chapter 4 for the precise resolution of singular arcs problems. Requires proper intialization, that can be provided by the Discretized BVP/Control formulations.

3: Light structured shooting: simplified version of the above.

4: Discretized Control: introduced in chapter 4, still experimental.

The formulation choice is nearly invisible in all the simplicial part of the code, and only shows in the file *Shoot.f90*, where the "shooting function" is actually evaluated.

B.2 Labelings

The Simplicial code actually features an integer labeling

[1] or the author...

0: Standard vector labeling.

1: Integer labeling, see below.

We have implemented the integer labeling described in [2], page 168, which is defined by:

$$L : \mathbf{R}^{n+1} \to [1..n+1]$$

$$v^i = (x^i, \lambda^i) \mapsto 1 + m$$

where m is the number of strictly positive components of $H(x^i, \lambda^i)$.

A face is completely labeled if and only if the labels of its vertices exactly match the set $[1..n+1]$

$$f = \{v^1, .., v^{n+1}\} \textit{ completely labeled } \Leftrightarrow \{L(v^1), .., L(v^{n+1})\} = \{1, .., n+1\}$$

Finding the exit face is here extremely simple: the vertex of the face that has to be pivoted is the one having the same label as the remaining vertex of the simplex (therefore the new face has the same label set as the current one, and is completely labeled too).

The main difficulty with this integer labeling is finding the first labeled face. Using (directly at least) a junction homotopy like in the classical labeling case is here ineffective, as we have no trivial starting labeled face, even in the affine case (see page 32 for more details).

Again, the code has been structured so that adding a new labeling would be as simple as possible. Only three subroutines (located in *Cell.f90*) actually deal with the implementation of the labeling, and have therefore to be provided for a new one (here are given the subroutines corresponding to the two implemented labelings, classical and integer):

- Finding the exit face of the current simplex (**Lex / PseudoLex**)
- Labeling on the first face evaluation (**FirstHLabel / FirstIntLabel**)
- Labeling check (**LabCheck / IntLabCheck**)

B.3 Integrators

B.3.1 Variable step integrators

The variable step integrators available at the moment are:

• RKF45 (code by Shampine and Watts)
Embedded Runge Kutta formulas, Fehlberg method, with local extrapolation (ie 5th order used for the integration instead of 4th)

• DOP853 (code by Hairer and Wanner)
Embedded Runge Kutta formulas, improved version of Dormand-Prince 8(6) (with the 6th order error estimation replaced by a 5th-3rd order)

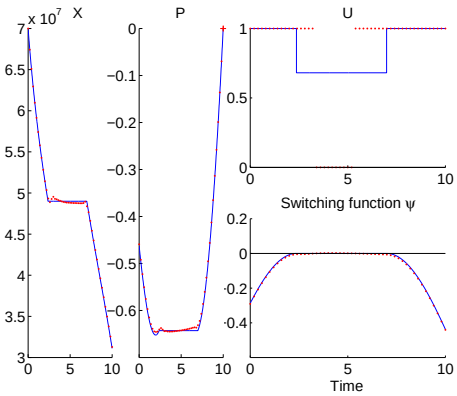
• ODEX (code by Hairer and Wanner)
Gragg-Bulirsch-Stoer extrapolation method (Aitken-Neville, symmetric case, using Gragg scheme as 2nd order base formula), both order and stepsize are variable.

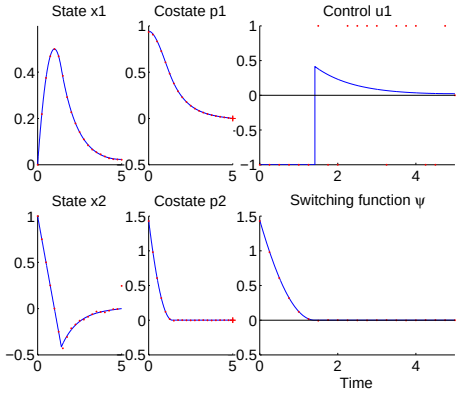
The links to the source files of these integrators are indicated among the third party references (page 132). A complete description of the algorithms can be found in [27].

B.3.2 Symplectic integrators

As we integrate a Hamiltonian system, we have also tried two symplectic integrators, the Stormer-Verlet and Gauss methods described in [27]. It should be noted that our implementation of these methods is extremely basic, especially for the fixed point part (both methods are implicit). Anyway, some of the preliminary results are interesting.

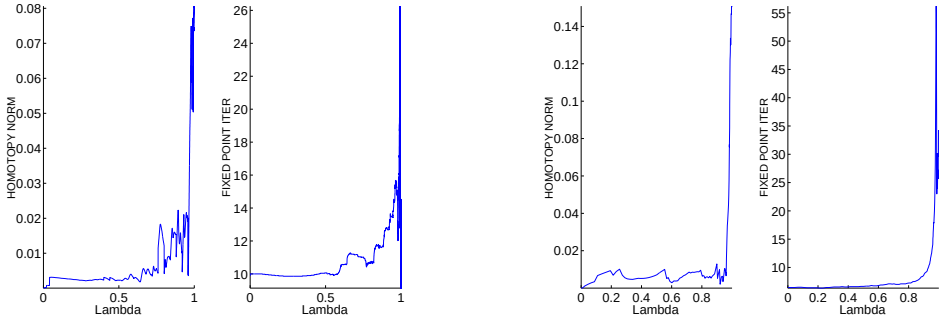
As a first example, we represent the Stormer-Verlet method applied to the discretized BVP formulation of the singular arc problems studied in chapter 4.





Discretized BVP for Problems 1 and 2 - Symplectic Stormer-Verlet

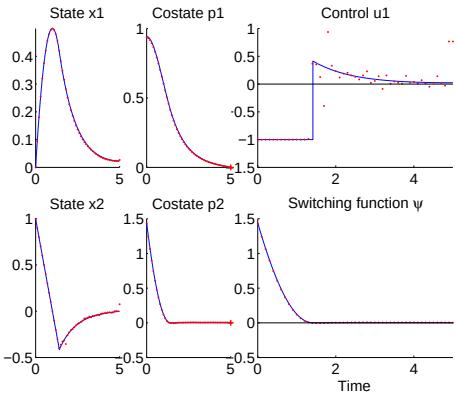
The solutions are quite good, especially for Problem 2. Now we can look at the evolution of the average fixed point iterations along the path.

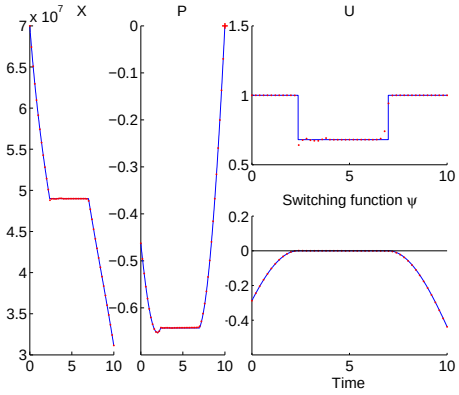


Evolution of $|H|$ and FP iterations for Problems 1 and 2 (S-V)

It is here interesting to note that the steep increase in the number of iterations seems to correspond to the degradation of the homotopy norm. Both are probably related to the regularity loss when coming close to the singular arcs.

The second example is with the Gauss method on the discretized control formulation.





Discretized control for Problems 1 and 2 - Symplectic Gauss

The solutions are here again quite good, on the same level as the best ones we obtain by RK4, but with a longer execution time however. Also, the numerical settings of the fixed point is here a bit tricky for Problem 2. Anyway, these are likely due to our crude implementation of the methods, which should be improved before conducting further experiments.

Appendix C

Sample files for some studied problems

We reproduce here some of the input files used in the experiments, to show that it is easy to modify the settings for the simplicial algorithm or the problem itself. We would like to emphasize that all tests were conducted with the same Simplicial code (with only one separate executable for each problem studied, regardless of the formulation in particular).

C.1 Demonstration problem

This is the first simple example described in chapter 1 and 2.

C.1.1 InitPar

We begin with the parameters initialization. Nothing much to say, except that we can consider several continuations for a given problem: we have here the continuation on the terminal conditions mentioned on page 30, as well as the first continuation on the objective introduced on page 15. The variable "homotop" corresponds to the "Homotopy choice" parameter in the .cfg and .in input files, and indicates the continuation used.

```
Subroutine InitPar(mode, lambda)
  implicit none

  integer, intent(in) :: mode
  real(kind=8), intent(in) :: lambda

  !mode = 0: first call
  !mode = 1: homotopy call

  if (mode == 0) then

     tf0 = tf
     cf0 = cf
     ci0 = ci
```

```
   elseif (homotop == 2) then

      cf(1) = 0.5d0 - lambda

   end if

end subroutine InitPar
```

C.1.2 Control

Here is the subroutine giving the optimal control, which corresponds to the expressions given in chapters 1 and 2.

```
Subroutine Control(lambda,t,x,p,u,psi)
  implicit none

  real(kind=8), intent(in) ::  lambda, t
  real(kind=8), intent(in), dimension(ns) :: x
  real(kind=8), intent(in), dimension(nc) :: p
  real(kind=8), intent(out), dimension(m) :: u
  real(kind=8), intent(out), dimension(dimpsi) :: psi

  select case (homotop)

  case (1)

     if (lambda < 1d0) then
        if (abs(p(2)) <= lambda .or. p(2)==0) then
           u(1) = 0d0
        elseif (abs(p(2)) >= 2d0 - lambda) then
           u(1) = -p(2) / abs(p(2))
        else
           u(1) = -p(2)*(abs(p(2))-lambda) / (2d0*(1d0-lambda)*abs(p(2)))
        end if
     else
        if (abs(p(2)) < 1d0 .or. p(2)==0) then
           u(1) = 0d0
        else
           u(1) = -p(2) / abs(p(2))
        end if
     end if

  case (2)

     if (abs(p(2)) < 1d0 .or. p(2)==0) then
        u(1) = 0d0
     else
        u(1) = -p(2) / abs(p(2))
     end if

  case default
     write(outputfid,*) 'ERROR : Control >>> Unknown homotop...',homotop
     stop
```

```
    end select

    !switch value and successive derivatives
    psi(1) = 1d0 - abs(p(2))

  end subroutine Control
```

C.1.3 Dynamics

Now the state and costate (and objective) dynamics (we can recognize the continuation on the first objective).

```
  Subroutine Dynamics(dimphi,lambda,t,x,p,u,phi)
    implicit none

    integer, intent(in) :: dimphi
    real(kind=8), intent(in) :: lambda, t
    real(kind=8), intent(in), dimension(ns) :: x
    real(kind=8), intent(in), dimension(nc) :: p
    real(kind=8), intent(in), dimension(m) :: u
    real(kind=8), intent(out), dimension(dimphi) :: phi

    select case (homotop)

    case (1)

       phi(1) = x(2)
       phi(2) = u(1)
       phi(3) = 0d0
       phi(4) = -p(1)

       !objective dynamic
       if (dimphi > ns+nc) phi(ns+nc+1) = lambda * abs(u(1)) + (1d0-lambda)*u(1)**2

    case (2)

       phi(1) = x(2)
       phi(2) = u(1)
       phi(3) = 0d0
       phi(4) = -p(1)

       !objective dynamic
       if (dimphi > ns+nc) phi(ns+nc+1) = abs(u(1))

    case default
       write(outputfid,*) 'ERROR : Dynamics >>> Unknown homotop...',homotop
       stop
    end select

  end subroutine Dynamics
```

C.1.4 Input files

Here are the input files corresponding to the graph on page 52, with a K_1 triangulation until $\lambda = 0.9$ and a J_3 refinement to $\lambda = 1$ (hence the refinement mode set to 1 and the refine bound for λ at 0.9). We notice that the path and simpath output are both set to 1, to generate the files used by the visualization scripts.

Configuration file

```
Simplicial algorithm parameters
Homotopy choice and problem class
1 1
Triangulation choice
1
Labeling choice
0
Zero path follow mode
-1
Initial and final solver call
1 0
Refinement mode and Max refinement attempts
1 1
Deviation tolerance
1d-1
Anisotropic thresholds
2.0 0.2 0.1

Maximum simplices for main and junction homotopy, check frequency
500000 10000 100000
Starting triangulation size
1d-1 1d-1
Lower, refine and upper bounds for lambda
0d0 0.9d0 1d0

Path, simplices,valley and plateau output files
1 1 0 0
Ratio for saved paths
1 1
Homotopy Norm and Criterion output in Path
1 0
Valley range and step
50 0.1
Plateau range and steps
50 0.1

Solution generation at final face vertices, Barycentric control
0 0
```

Problem file

```
Homotopy choice and problem class
```

```
1 1
Unknown, State, Costate and Control dimensions
2 2 2 1
Objective and Switch dimension
1 1

Number of IVP unknown values
2
IVP unknown indices
3 4

Number of initial values
2
Initial values indices
1 2
Initial values
0 0

Number of terminal values
2
Terminal values indices
1 2
Terminal Values
0.5 0

Initial and final time
0.0 2.0

Starting point (x0,lambda0) for zeropath
0
0
0

Scaling mode
1
Path and solution integrator choice
4 4
steps for fixed steps integrators
1000 1000
Variable step integrator abserr and relerr
1d-16 1d-14 1d-16 1d-14
Fixed Point minimal progress and maximal iterations
1d-2 25

Number of parameters
0
Parameters
```

C.2 Orbital transfer problems

Concerning the orbital transfer problems studied in chapter 3, the optimal control and dynamics are much more complicated (the costate dynamics in

particular...), and instead of copy/pasting four pages of code, we rather refer the interested reader to the thesis of Thomas Haberkorn ([26]), who wrote the original formulas. Here are the input files corresponding to the transfer with $T_{max} = 0.1N$, with a basic uniform following (follow mode set to 0).

Configuration file

```
Simplicial algorithm parameters
Homotopy choice and problem class
1 1
Triangulation choice
1
Labeling choice
0
Zero path follow mode
0
Initial and final solver call
0 1
Refinement mode and Max refinement attempts
0 0
Deviation tolerance
1d-1
Anisotropic thresholds
2 0.2 0.1

Maximum simplices for main and junction homotopy, check frequency
10000 500 5000
Starting triangulation size
1d-1 1d-1
Lower, refine and upper bounds for lambda
0 0.9 1

Path, simplices, valley and plateau output files
0 0 0 0
Ratio for saved paths
1 1
Homotopy Norm and Criterion output in Path
0 0
Valley range and step
10 0.1
Plateau range and step
10 0.1

Solution generation at final face vertices, barycentric control
0 0
```

Problem file

This one corresponds to a DOP853 shooting attempt, as indicated by the second integrator choice set to 6 (the preceding 5 selects RKF45 as the path integrator). We can also notice the use of the soft scaling (scaling mode set to 2).

```
Homotopy choice and problem class
1 1
Unknown, State, Costate and Control dimensions
8 8 8 3
Objective and Switch dimension
1 2

Number of IVP unknown values
8
IVP unknown indices
8 9 10 11 12 13 14 15

Number of initial values
8
Initial values indices
1 2 3 4 5 6 7 16
Initial values
11.625 0.75 0 0.0612 0 0 1500 0

Number of terminal values
8
Terminal values indices
1 2 3 4 5 6 15 16
Terminal Values
42.165 0 0 0 0 1 0 0

Initial and final time
3.14159 2678.52159

Starting point (x0,lambda0) for zeropath
 1.53157896392868E+04
-7.82647827592651E+02
-2.13866401262451E+04
-3.97023354861161E+00
 3.56466373170132E+03
-1.22948602187706E+00
 4.17593485715192E+00
 6.08318784116258E+00
 0.00000000000000E+00

Scaling mode
2
Integrator choice
5 6
Fixed step integrator steps
1500 2000
Variable step integrator abserr and relerr
1e-8 1e-6 1e-12 1e-10
Fixed point min prog and max iter
1d-2 25

Number of parameters
5
Parameters
```

```
0.1 1.77 0.0142 5165.86 12.96
```

Note: T_{max} is the first parameter on the last line, the others are various coefficients used in the transfer formulation.

C.3 Optimal harvesting in fishery

Here is the first of the two singular arc problems studied in chapter 4. We give here the files for the discretized BVP continuation, as indicated by the problem class set to 0. More specifically, it is the implicit trapeze formulation (integrator choice set to 1 in the .in file), with full adaptive path following (follow mode set to 3), and 10 refinement attempts at the solution (the refinement modes 1 and 2 correspond to the J3-J4 refinement before the convergence, and 3 is for the refinements after the convergence).

Configuration file

```
Simplicial algorithm parameters
Homotopy choice and problem class
1 0
Triangulation choice
1
Labeling choice
0
Zero path follow mode
3
Initial and final solver call
1 0
Refinement mode and Max refinement attempts
3 10
Deviation tolerance
1d-2
Thresholds for Anisotropic
2.0 0.2 0.1

Maximum simplices for main and junction homotopy, check frequency
1000000 100000 100000
Starting triangulation size
1d-2 1d-2
Lower, refine and upper bounds for lambda
0 0.9 1.0

Path, simplices, valley and plateau output files
0 0 0 0
Ratio for saved paths
1 1
Homotopy Norm and Criterion output in Path
0 0
Valley range and step
10 0.1
Plateau range and step
```

```
10 0.1

Solution generation at final face vertices, Barycentric control
0 0
```

Problem file

```
Homotopy choice
1 0
Unknown, State, Costate and Control dimensions
101 1 1 1
Objective and Switch dimension
1 1

Number of IVP unknown values
1
IVP unknown indices
2

Number of initial values
1
Initial values indices
1
Initial values
7e+07

Number of terminal values
1
Terminal values indices
2
Terminal Values
0

Initial and final time
0 10

Starting point (x0,lambda0) for zeropath
0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
7e7 0 7e7 0 7e7 0 7e7 0 7e7 0
0.0

Scaling mode
1
Path integrator choice
```

```
1 1
Steps for fixed step integrator
1 1
Variable step integrator abserr and relerr
1d-16 1d-14 1d-16 1d-14
Fixed point minimal progression and maximal iterations
1d-4 100

Number of parameters
7
Parameters
1 1.75d+07 0.71d0 8.05d+07 2d+07 0d0 1d0
```

C.4 Quadratic regulator

To finish with, here are the files corresponding to the discretized control continuation (problem class set to 4), for the second singular arc problem.

Configuration file

```
Simplicial algorithm parameters
Homotopy choice and problem class
1 4
Triangulation choice
1
Labeling choice
0
Zero path follow mode
0
Initial and final solver call
1 1
Refinement mode and Max refinement attempts
0 0
Deviation tolerance
1d-1
Anisotrpic thresholds
2.0 0.2 0.1

Maximum simplices for main and junction homotopy, check frequency
5000000 500000 100000
Starting triangulation size
1d-1 1d-1
Lower, refine and upper bounds for lambda
0d0 0.9d0 1.0d0

Path, simplices, valley and plateau output files
0 0 0 0
Ratio for saved paths
1 1
Homotopy Norm and Criterion output in Path
0 1
Valley range and step
```

```
10 0.1
Plateau range and step
10 0.1

Solution generation at final face vertices, barycentric control
0 0
```

Problem file

```
Homotopy choice
1 4
Unknown, State, Costate and Control dimensions
52 2 2 1
Objective and Switch dimension
1 2

Number of IVP unknown values
2
IVP unknown indices
3 4

Number of initial values
2
Initial values indices
1 2
Initial values
0 1

Number of terminal values
2
Terminal values indices
3 4
Terminal Values
0 0

t0 tf
0d0 5d0

Starting point (x0,lambda0) for zeropath
1 1
0 0 0 0 0   0 0 0 0 0
0 0 0 0 0   0 0 0 0 0
0 0 0 0 0   0 0 0 0 0
0 0 0 0 0   0 0 0 0 0
0 0 0 0 0   0 0 0 0 0
0.0

Scaling mode
1
Path integrator choice
4 4
fixed step integrators
1 1
```

```
Variable step integrator abserr and relerr
1d-16 1d-14 1d-16 1d-14
fpminprog and fpmaxiter
1d-6 100

Number of parameters
3
Parameters
1d0 -1d0 1d0
```

Bibliography

[1] E. Allgower and K. Georg. Simplicial and continuation methods for approximating fixed points and solutions to systems of equations. *Siam review*, 22(1):28–85, 1980.

[2] E. Allgower and K. Georg. *Numerical Continuation Methods*. Springer-Verlag, Berlin-Heidelberg-New York, 1990.

[3] E. Allgower and K. Georg. Piecewise linear methods for nonlinear equations and optimization. *Journal of Computational and Applied Mathematics*, 124:245–261, 2000. Special Issue on Numerical Analysis 2000: Vol. IV: Optimization and Nonlinear Equations.

[4] U.M. Ascher, R.M. Mattheij, and R.D. Russel. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. Prentice Hall, 1988.

[5] J.P. Aubin and A. Cellina. *Differential Inclusion*. Springer-Verlag, 1984.

[6] N. Berend, F. Bonnans, M. Haddou, J. Laurent-Varin, and C. Talbot. A preliminary interior point algorithm for solving optimal control problems. November 2003. 5th International Conference on Launcher Technology (Madrid).

[7] N. Berend, F. Bonnans, M. Haddou, J. Laurent-Varin, and C. Talbot. On the refinement of discretization for optimal control problems. June 2004. 16th IFAC Symposium on Automatic Control in Aerospace (St Petersburg).

[8] C. Berge. *Espaces topologiques*. Dunod, Paris, 1959.

[9] H. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In W. Jiger K. Ebert, P. Deuflhard, editor, *Modelling of Chemical Reaction Systems*, pages 102–125. Springer, Berlin, 1981.

[10] J.F. Bonnans. The shooting algorithm for optimal control problems: a review of some theoretical and numerical aspects. Technical re-

port, Université d'El Manar (Tunis), 2002. Lectures notes, DEA de Mathématiques Appliquées de l'ENIT.

[11] Brezis. *Analyse fonctionnelle.* Masson, 1983.

[12] A.E. Bryson and Y.C. Ho. *Applied Optimal Control - Optimization, Estimation, and Control.* Blaisdell publishing company, 1969.

[13] R.H. Byrd, J.C. Gilbert, and J. Nocedal. *A trust region method based on interior point techniques for nonlinear programming.* 1998.

[14] R.H. Byrd, M.E. Hribar, and J. Nocedal. *An interior point algorithm for large scale nonlinear programming.* 1997.

[15] JB. Caillau, R. Dujol, J. Gergaud, T. Haberkorn, P. Martinon, J. Noailles, and D. Preda. Mise au point d'une méthode de résolution efficace pour les problèmes de contrôle optimal à solution "bang-bang" - application au calcul de trajectoires à poussée faible. Technical report, ENSEEIHT-IRIT, UMR CNRS 5505, 2 rue Camichel, F-31071 Toulouse, January 2004. Rapport de fin de phase 2 - Contrat 02/CNES/0257/00 - DPI 500.

[16] JB. Caillau, R. Dujol, J. Gergaud, T. Haberkorn, P. Martinon, J. Noailles, and D. Preda. Mise au point d'une méthode de résolution efficace pour les problèmes de contrôle optimal à solution "bang-bang" - application au calcul de trajectoires à poussée faible. Technical report, ENSEEIHT-IRIT, UMR CNRS 5505, 2 rue Camichel, F-31071 Toulouse, January 2005. Rapport de contrat - Contrat 02/CNES/0257/00 - DPI 500.

[17] JB. Caillau, J. Gergaud, and J. Noailles. 3D Geosynchronous Transfer of a Satellite: Continuation on the Thrust. *Journal of Optimization Theory and Applications*, 118(3):541–565, 2003.

[18] L. Cesari. *Optimization theory and application. Problems with ordinary differential equations.* Springer-Verlag, New York, 1983.

[19] Y. Chen and J. Huang. A numerical algorithm for singular optimal control synthesis using continuation methods. *Optimal Control Applications & Methods*, 15:223–236, 1994.

[20] C.W. Clark. *Mathematical Bioeconomics.* John Wiley & Sons, 1976.

[21] C. Dang. The d_1 triangulation of $\mathbf{R}^n$ for simplicial algorithms for computing solutions of nonlinear equations. *Mathematics of Operations Research*, 16(1):148–161, 1991.

[22] A.F. Filippov. *Differential Equations with Discontinuous Righthand Sides.* Kluwer Academic Publishers, Dordrecht-Boston-London, 1988.

[23] W.H. Flemming and R.W. Rishel. *Deterministic and Stochastic Optimal Control*. Springer-Verlag, 1975.

[24] J. Gergaud. *Résolution numérique de problèmes de commande optimale à solution Bang-Bang par des méthodes homotopiques simpliciales*. PhD thesis, Institut National Polytechnique de Toulouse, 1989.

[25] J. Gergaud, T. Haberkorn, and P. Martinon. Low thrust minimum-fuel orbital transfer: an homotopic approach. *Journal of Guidance, Control and Dynamics*, 27(6):1046–1060, 2004.

[26] T. Haberkorn. *Transfert orbital à poussée faible avec minimisation de la consommation: résolution par homotopie différentielle*. PhD thesis, Institut National Polytechnique de Toulouse, 2004.

[27] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Comput. Mathematics*. Springer-Verlag, Berlin-Heidelberg-New York, 1993. Second Revised Edition.

[28] R. Hartl, S. Sethi, and R. Vickson. A survey of the maximum principles for optimal control problems with state constraints. *Siam review*, 37(2):181–218, 1995.

[29] P. Martinon and J. Gergaud. An application of pl continuation methods to singular arcs problems. In A. Seeger, editor, *Recent Advances in Optimization*, Lectures Notes in Economics and Mathematical Systems. Springer-Verlag, 2005 (second semester).

[30] H. Nikaido. *Convex structures and economic theory*. Academic Press, 1968.

[31] H.J. Oberle and W. Grimm. BNDSCO - A Program for the Numerical Solution of Optimal Control Problems. Technical Report 515, Institut for Flight System Dynamics, Oberpfaffenhofen, German Aerospace Research Establishment DLR, 1989.

[32] L. Pontriaguine, V. Boltianski, R. Gamkrelidze, and E. Michtchenko. *Théorie Mathématique des Processus Optimaux*. Editions Mir, Moscou, 1974.

[33] R. Robert. *Contributions à l'analyse non linéaire*. PhD thesis, Université Scientifique et Médicale de Grenoble et Institut National Polytechnique de Grenoble, 1976.

[34] K. Schilling. An algorithm to solve boundary value problems for differential inclusions and applications in optimal control. *Numerical Functional Analysys and Optimization*, 10(7-8):733–764, 1989.

[35] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, Berlin-Heidelberg-New York, 1983.

[36] M.J. Todd. The computation of fixed points and applications. In *Springer Lectures Notes in Economics and Mathematical Systems*, volume 124 (VII). Springer-Verlag, Heidelberg-New York, 1976.

[37] M.J. Todd. Union jack triangulations. In *Fixed Points: Algorithms and Applications*, pages 315–336. Karamadian, Academic Press, New York, 1977.

[38] L.T. Watson, M. Sosonkina, R.C. Melville, A.P. Morgan, and H.F. Walker. Algorithm 777: HOMPACK90: A suite of fortran90 codes for globally convergent algorithms. *ACM Transactions on Mathematical Software*, 23:514–549, 1997.