A Preliminary Interior Point Algorithm For Solving Optimal Control Problems[1]

5th International Conference on Launcher Technology
Madrid, November 25-27-2003

Nicolas BEREND[1], Frédéric BONNANS[2], Mounir HADDOU[3], Julien LAURENT-VARIN[4], Christophe TALBOT[5]

[1]ONERA Long-Term Design and System Integration Department BP 72 - 29 avenue de la Division Leclerc 92322 Chatillon Cedex Nicolas.Berend@onera.fr

[2]Projet SYDOCO, INRIA Rocquencourt B.P. 105 78153 Le Chesnay Cedex France frederic.bonnans@inria.fr

[3]Maître de Conférences UMR 6628-MAPMO B.P. 6759 45067 Orléans cedex 2 mounir.haddou@labomath.univ-orleans.fr

[4]Projet SYDOCO, INRIA Rocquencourt B.P. 105 78153 Le Chesnay Cedex France julien.laurent-varin@inria.fr

[5]CNES : launcher directorate Rond-point de l'Espace - 91023 Evry Cedex christophe.talbot@cnes.fr

Abstract

In the frame of future launcher design studies, CNES and ONERA are studying possible improvements of tools and methods for trajectory optimization. Trajectory optimization of reusable launch vehicle (RLV) is a very complex task calling for a versatile tool, which should be able to address - either simultaneously or separately - ascent and reentry trajectory phases. Improvement can be made in comparison with existing methods and tools regarding issues such as the global processing of ascent and reentry phases, or specific constraints of reentry phases. In this context, a first candidate for a new optimization method is currently being studied jointly with INRIA. This method is an adapted interior point algorithm, associated with a Runge-Kutta discretization scheme of the optimal control problem. This method seems to be very promising especially with regard to constraints processing, because no assumption about the current set of active constraints is necessary.

[1]Sponsored by CNES and ONERA

Introduction

This paper presents a new method for the trajectory optimization of reusable launch vehicles. We would like to make improvements in various fields of optimization : in time of calculation, in path constraint treatment and in global trajectories mission resolution.

First, future reusable space launcher trajectories include reentry phases with dynamic pressure constraint, thermal flux constraint and some other path constraints like for the space shuttle reentry (see [2]). But in that case some problems of treatment appear, for example we have to assume some hypothesis on active and nonactive constraints and then optimize; interior point methods don't make any kind of assumptions. The part 1 of this paper will present this aspect of the method.

In addition, we would like to gain time during optimization execution. When a new space launcher is being designed, it can be usefull to calculate many different trajectories for different kinds of concepts and design parameters. For this reason, a fast optimization method is needed. In an other hand, we have to discretize the continuous optimality conditions to be able to compute it. Then we choose to minimize the number of points of this discretization for the first moment of optimization procedure, and when we get closer to the solution, we refine the mesh in order to reach the solution as fast as possible. An error estimation is then required and a refinement procedure is needed too. The part 2 of this paper will present this aspect of the method.

An other main topic is to solve the global mission trajectories for a totally reusable space launcher with different phases of trajectories connected by physical conditions. This kind of mission can be seen as a particular case of a more general problem defined on a scenariograph that we will deal with in the part 3.

1 Interior point method

Launcher trajectories, especially RLV ones, must take into account path constraints. In most classical optimization methods, assumptions have to be made conserning active or not active constraint. Here, interior point method will not make any assumption and then, a trajectory might be obtained faster.

The idea of this method is to add to the cost fonction a penalty term called a barrier function. The first persons who have studied logarithmic barrier methods were Fiacco and McCormick in [6]. In an other hand, the active set method frequently used in linear programming with the simplex method shows an exponential worst case complexity. In linear optimization, the first algorithms with polynomial complexity was the ellipsoid method by Khachiyan [7] in 1979, but this one was slower than simplex for practical problems. The breakthrough has been achieved by Karmarkar [12] in 1984 who proved polynomial complexity for his algorithm. That achieved competitive performance for linear programming and turned out to be a logarithmic barrier method afterwards.

In nonlinear programming, Gilbert and Nocedal study interior points algorithms (see [5, 1, 3]) and Wright studies the application to the control optimal problems (see [13, 14, 15]).

Here, we would like to have the same peformance gap between the old active constraints methods and this interior point approch.

1.1 Formulation of the optimal control problem using logarithmic penalization

The problems of launcher trajectory optimization always have the same form, with a cost function, a dynamic for the state f, initial and final condition for some state variables and some continuous path constraints g.

The problem of RLV trajectory optimization can be stated as the following optimal control problem (P). We want to find the optimal control $u(t)$ such that :

$$\left\{\begin{array}{l} \text{Min} \displaystyle\int_0^T \ell(u(t),y(t))dt + \ell_f(y(T)); \\ \dot{y}(t) = f(u(t),y(t)), \quad t \in [0,T]; \\ \\ y_i(0) = y_i^0 \quad \forall i \in I_0, \\ y_i(T) = y_i^T \quad \forall i \in I_T \\ \\ a \leq g(u(t),y(t)) \leq b, \quad t \in [0,T]. \end{array}\right. \tag{P}$$

where y is the state vector and f its dynamics, ℓ and ℓ_f are the integral and final terms of the cost function, g is the path constraint function, and a and b its bounds.

We introduce the logarithmic penalization with this new integral term of the cost function :

$$\begin{array}{l} \ell_\varepsilon(u,y) = \ell(u,y) \\ \quad -\varepsilon \displaystyle\sum_i [\log(g_i(u,y) - a_i) \\ \qquad + \log(b_i - g_i(u,y))] \end{array}$$

Then we obtain the following unconstrainted problem :

$$\left\{\begin{array}{l} \text{Min} \displaystyle\int_0^T \ell_\varepsilon(u(t),y(t))dt + \ell_f(y(T)); \\ \\ \dot{y}(t) = f(u(t),y(t)), \qquad t \in [0,T]; \\ \\ y_i(0) = y_i^0 \quad \forall i \in I_0, \\ y_i(T) = y_i^T \quad \forall i \in I_T. \end{array}\right. \tag{P_ε}$$

Then, the optimization strategy consists in solving the unconstrained problem (P_ε) with decreasing values for the penalization factor ε, thus making the cost function more and more associated to constraint satisfaction.

The next step is to describe the optimality conditions to solve, an ε being given, and how to do it.

1.2 Optimality conditions

We set the Hamiltonian function of the initial problem (P) :

$$\begin{array}{l} H(y,u,p,\lambda,\mu) = \ell(u,y) + p^T f(u,y) \\ \quad +\lambda^T(g(u,y) - a) + \mu^T(b - g(u,y)) \end{array} \tag{1}$$

Optimal condition of (P) is :

$$\left\{\begin{array}{rcl} \dot{y}(t) & = & H_p(y(t),u(t),p(t),\lambda(t),\mu(t)) \\ \\ \dot{p}(t) & = & -H_y(y(t),u(t),p(t),\lambda(t),\mu(t)) \\ \\ u(t) & = & \text{Argmin}_w\, H(y(t),w,p(t),\lambda(t),\mu(t)) \\ \\ 0 & = & \lambda^T(g(u,y) - a), \quad \lambda \geq 0 \\ \\ 0 & = & \mu^T(b - g(u,y)), \quad \mu \leq 0 \end{array}\right. \tag{2}$$

We set the Hamiltonian function of the penalized problem (P_ε) :

$$H^\varepsilon(y,u,p) = \ell_\varepsilon(u,y) + p^T f(u,y) \tag{3}$$

The Pontryaguin principle gives us the well known necessary optimality conditions :

$$\left\{\begin{array}{rcl} \dot{y}(t) & = & H_p^\varepsilon(y(t),u(t),p(t)) \\ \\ \dot{p}(t) & = & -H_y^\varepsilon(y(t),u(t),p(t)) \\ \\ u(t) & = & \text{Argmin}_w\, H^\varepsilon(y(t),w,p(t)) \end{array}\right. \tag{4}$$

Now we transform this formulation into a primal-dual one like it is done in linear optimization. The optimality conditions become :

$$\left\{\begin{array}{rcl} \dot{y}(t) & = & H_p(y(t),u(t),p(t),\lambda(t),\mu(t)) \\ \dot{p}(t) & = & -H_y(y(t),u(t),p(t),\lambda(t),\mu(t)) \\ u(t) & = & \mathrm{Argmin}_w\, H(y(t),w,p(t),\lambda(t),\mu(t)) \\ \mathbf{1}\varepsilon & = & \lambda\cdot(g(u,y)-a), \quad \lambda>0 \\ -\mathbf{1}\varepsilon & = & \mu\cdot(b-g(u,y)), \quad \mu<0 \end{array}\right. \tag{5}$$

Where $\mathbf{1}$ is the vector in $\mathbb{R}^{n_g}$ where each component is 1, n_g being the constraint dimention and $x\cdot y$ the vector $(x_i y_i)_{i=1..n_g}$.

Now we just need to solve this problem for a decreasing ε. After the discretization that we will expose in part 2, we have a set of non linear equations to solve.

Here we remark that a solution of optimal conditions (5) when ε decrease to 0 is a solution of (2).

Let us see an example.

1.3 Example

We consider the very simple academic problem :

$$\mathrm{Min}\int_0^3 x(t)\mathrm{d}t;$$

with the dynamics :

$$\dot{x}(t)=u(t),$$

the boundary conditions :

$$x(0)=x(3)=1$$

and under path constraints :

$$\left\{\begin{array}{rcccl} -1 & \leq & u & \leq & 1 \\ 0 & \leq & x & \leq & 2 \end{array}\right.$$

This example is interesting because the state-constraint makes the optimal solution a singular arc, so the optimal control has non-saturated values, unlike the so-called "bang-bang" solution.

Figures 1, 2 and 3 show the evolution of the solution when ε decreases, for respectively the state, the control and the co-state.

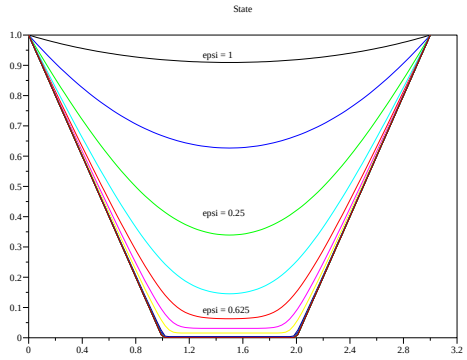

Figure 1: State as a function of time

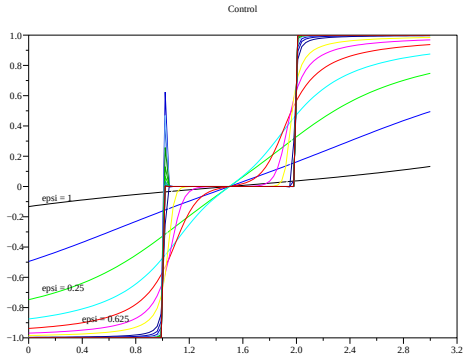

Figure 2: Control as a function of time

The graphics show the convergence of the solution of the unconstrained problem (P_ε) towards the exact solution (thick line) when ε decreases from 1 to 1.90735×10^{-6}.

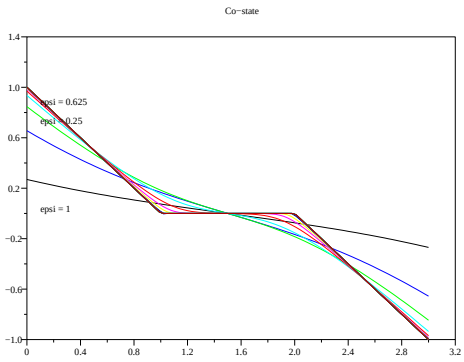

Figure 3: Co-state as a function of time

2 Discretization of optimality conditions

To compute the solution of the continuous optimality conditions, we first discretize them. This transforms the optimal control problem into a set of non-linear equations, which has to be solved for the discretized controls, state and adjoint vectors using a Newton method. Many discretization schemes exist. First, we eliminate multi-step methods because of the co-state discontinuity at the moments of state-constraint activation. So we choose a one step method. Secondly, we would like to minimize the number of discretization points in order to have a fast algorithm, then a high-order method is necessary. Indeed, high-order methods will allow us to optimize the efficiency of each discretization points with a good precision. Finally, the collocation method appears as a particular case of Runge-Kutta methods. For these reasons, we choose the Runge-Kutta method.

In the following, we will give a reminder of the Runge-Kutta ideas, then we will briefly present the extention, the method of error estimation and the policy of refinement of the discretization grid.

2.1 Runge-Kutta method for ODE

An Ordinary Differential Equation (ODE) is an equation of the form (6).

$$\dot{y}(t) = f(y(t), t), \forall t \in [0, T] \quad y(0) = y^0 \quad (6)$$

After a discretization into small subintervals, we sequentialy solve the set of non-linear equations :

$$\left\{ \begin{array}{rcl} y_{k+1} & = & y_k + h_k \sum_{i=1}^{s} b_i f(y_{ki}, t_{ki}) \\ y_{ki} & = & y_k + h_k \sum_{j=1}^{s} a_{ij} f(y_{ki}, t_{ki}) \\ t_{k+1} & = & t_k + h_k \\ t_{ki} & = & t_k + c_i h_k \end{array} \right. \quad (7)$$

A Runge-Kutta method is defined by a number of stage s, two vector c and b and a matrix a. It is usually presented under the form of a Butcher array (see [4]) :

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array}$$

2.2 Optimal control and symplectic integration

In order to certify the optimality, we have to ensure a good precision for the co-state. We want to choose the RK coefficients so that the integration has a pre-determined order (the order characterizes the method's precision). The order of the method can be assessed from the coefficients through equations called "order conditions".

In his paper [8], Hager gives en explicit formula for dual Runge-Kutta methods to be used for the transformed adjoint system. The order conditions for Hager's partitionned RK methods can be computed in an easier way by noting that these methods are symplectic (see [11]).

After using the interior point formulation explained before, the original constrained problem becomes an unconstrained problem of the form :

$$\begin{cases} \text{Min } \Phi(y(T)); \\ \dot{y}(t) = f(u(t), y(t)), \quad t \in [0, T]; \\ y(0) = y^0. \end{cases} \qquad (P)$$

The first order necessary optimality conditions are :

$$\begin{cases} \dot{y}(t) &= f(u(t), y(t)) \\ y(0) &= y^0 \\ \dot{p}(t) &= -f_y(u(t), y(t))^T p(t) \\ p(T) &= \Phi'(y(T)) \\ 0 &= f_u(u(t), y(t))^T p(t) \end{cases} \qquad (OC)$$

Then, we have a Hamiltonian system to solve.

We can either discretized the original control problem (P) or the optimality conditions (OC), then solve the corresponding discretized problem.

The discretization of (P) yields the following discretized problem (DP) :

$$\begin{cases} \text{Min } \Phi(y_N); \\ y_{k+1} = y_k + h_k \sum_{i=1}^s b_i f(u_{ki}, y_{ki}), \\ \quad k = 0, \ldots, N-1, \\ y_{ki} = y_k + h_k \sum_{j=1}^s a_{ij} f(u_{kj}, y_{kj}), \\ \quad k = 0, \ldots, N-1, \quad i = 1, \ldots, s, \\ y_0 = y^0. \end{cases} \qquad (DP)$$

A specific discretization (DOC) of the optimality conditions (OC) is obtained after some algebraic computation (see [8]) :

$$\begin{cases} y_{k+1} &= y_k + h_k \sum_{i=1}^s b_i f(u_{ki}, y_{ki}) \\ y_{ki} &= y_k + h_k \sum_{j=1}^s a_{ij} f(u_{kj}, y_{kj}) \\ p_{k+1} &= p_k + h_k \sum_{i=1}^s \hat{b}_i f_y(y_{ki}, u_{ki})^T p_{ki} \\ p_{ki} &= p_k + h_k \sum_{j=1}^s \hat{a}_{ij} f_y(y_{kj}, u_{kj})^T p_{kj} \\ 0 &= f_u(y_k, u_k)^T p_k \\ 0 &= f_u(y_{ki}, u_{ki})^T p_{ki} \\ k &= 0, \ldots, N-1, \quad i = 1, \ldots, s, \\ y_0 &= y^0, \qquad p_N = \Phi'(y_N). \end{cases} \qquad (DOC)$$

Where :

$$\begin{cases} \hat{b}_i = b_i; \quad \hat{a}_{ij} = \dfrac{b_i b_j - b_j a_{ij}}{b_i} \end{cases} \qquad (8)$$

Due to (8), this partitioned RK method happens to be symplectic. (see [11, Theorem 4.6]). In particular, the following diagram commutes when we use this discretization scheme. In other words computing optimality conditions of disretized problem is equivalent to discretize continuous optimality condition. (cf. the presentation of Runge-Kutta and symplectic methods in the books [9, 10]) :

$$\begin{array}{ccc} (P) & \xrightarrow{\text{discretization}} & (DP) \\ \text{optimality conditions} \downarrow & & \text{optimality conditions} \downarrow \\ (OC) & \xrightarrow{\text{discretization}} & (DOC) \end{array} \qquad (D)$$

The order r' of the discretization scheme of (DOC) is equal to the one r of (PD) if $r \leq 2$, but any less if $r > 2$. Hager gives the four first order conditions for having $r = r'$ when $r = 3$ and 4.

2.3 Error estimation and Refinement policy

Here, we are looking for a discrete solution near the continuous one. However, the Runge-Kutta scheme leads to a numerical error that

we would like to estimate in order to decrease the step size where this error is too high. Then we build $(\hat{y}_k)_{0\leq k\leq n}$ and $(\hat{p}_k)_{0\leq k\leq n}$ to estimate the error made on the interval $[t_k, t_{k+1}]$. The error estimates are based on a RK formula of order $(r, r+1)$,i.e. we compute :

$$\begin{cases} \hat{y}_{k+1} = y_k + h_k \sum_{i=1}^{s} e_i f(u_{ki}, y_{ki}) \\ \hat{p}_{k+1} = p_k + h_k \sum_{i=1}^{s} e_i f_y(u_{ki}, y_{ki})^T p_{ki} \end{cases}$$

The coefficient e_i chosen so that (a, e) is a symplectic Runge-Kutta of order $r+1$ [8].

Then we calculate $(e_k)_{0\leq k\leq N}$:

$$e_k := max(|\hat{y}_k - y_k|, |\hat{p}_k - p_k|) \qquad (9)$$

The global error is : $E_{\text{error}} := \sum_{k=0}^{N} e_k$

Now we have an estimation of the error on each subinterval. This estimation will be used to define which policy we should take to refine efficiently the discretization. We first choose a simple policy to improve the method : a rate $\tau \in]0, 1]$ being given, we will add one point in the middle of each interval which is doing the biggest error.

For that, we divide by two the q subintervals which have the largest local errors, with $q := \lceil \tau N \rceil$.

$\lceil x \rceil$ denotes the integer part of x.

Then we solve again the discretized equations with a Newton method and restart the error estimation. We stop when the error estimation is lower than a given value.

2.4 Example

We consider the following example which has different phases of variation :

$$\text{Min} \int_0^{10} \frac{1}{2}(y_1(t) - z(t))^2 + \frac{N}{2} u^2(t) \mathrm{d}t;$$

with

$$\begin{cases} \dot{y_1}(t) = y_2(t), \\ \dot{y_2}(t) = u(t) + \alpha y_1^3(t) + f(t) \end{cases}$$

and

$$y_1(0) = 1, \qquad y_2(0) = 0$$

This example is parametrized by the two functions $z(t)$ and $f(t)$, and the constants α and N. We can choose these parameters in order to have a special solution. Here, we consider a problem with different variations phases for the solution. The optimal solution will be made up of three phases : two phases where the control is changing, and one, between the other, where the control is constant. We expect that the refinement will mostly take place when the control is not constant.

Figure 4 gives the solution and the refined mesh obtained by the algorithm. The refinement histogram shows the point density after this refinement policy. We can see that the density is high when the control is changing, and low when the control is constant.

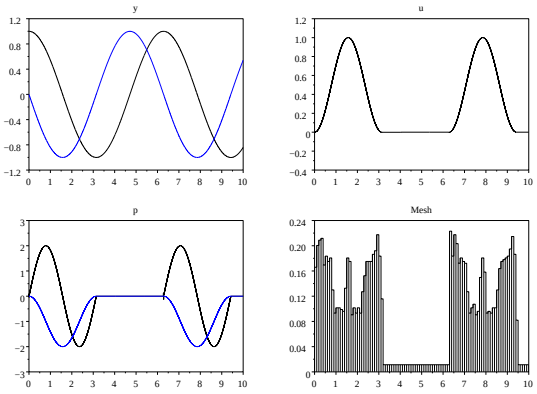

Figure 4: Refinement example

y	: state vector
p	: adjoint vector
u	: control
Mesh	: mesh density

3 Optimality conditions for multi-phase trajectories

The method described above applies to a single-phase optimal control problem. Actually, a RLV trajectory leads to a multi-phase optimal control problem. In the following, we will give a formalism to describe such a problem.

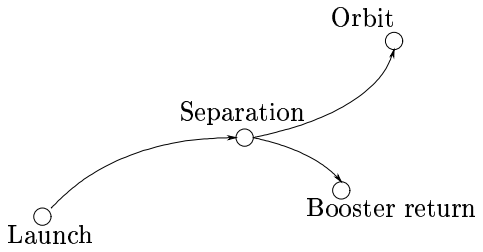

Figure 5: Example for TSTO scenario

We can represent a mission scenario with a graph, composed with nodes and edges. We need to calculate the optimality conditions for such a problem, and to find a structure for solving the discretization of optimality conditions.

A global mission scenario can be modelized with phases of trajectories linked by junction points. It seems natural to interpret this scenario as an oriented graph where each node will represent the departure conditions, the separation and final constraints, and each edges the corresponding phase oriented following time evolution. (see example in Figure 5). We can remark that this formalism can easily be extended to more complexe scenarios, for example with a rendez-vous with another vehicle.

A graph is made up of a set of nodes N, and a set of edges $E \subset N \times N$. We will denote (i,j) the edge linking node i to node j, the orientation being important. The RLV scenario of Figure 5 can be expressed by the graph $G = (N, E)$, with :

$$\begin{array}{l} N = (L, S, B, O) \\ E = ((L,S),(S,B),(S,O)) \end{array} \tag{10}$$

The nodes are: L for the launch moment, S for the separation one, B the booster landing and O the injection into orbit.

We associate one state y_{ij} and one control u_{ij} to each edge, and we consider the following problem for the trajectory optimization of the whole mission :

$$\left\{ \begin{array}{l} \text{Min} \displaystyle\sum_{(i,j)\in E} \int_{t_i}^{t_j} \ell(y_{ij}, u_{ij}) \mathrm{d}t \\ \forall (i,j) \in E, \quad \dot{y}_{ij} = f_{ij}(y_{ij}, u_{ij}) \end{array} \right. \tag{11}$$

For simplification purpose, we have omitted the depency on time for y and u. We define the boundary conditions and the conditions of state transition :

$$\left\{ \begin{array}{l} y_{LS}(t_L) = \Phi_L \\ y_{SE}(t_E) = \Phi_E \\ y_{SB}(t_B) = \Phi_B \\ y_{SE}(t_S) = \Phi_S^1(y_{LS}(t_S)) \\ y_{SB}(t_S) = \Phi_S^2(y_{LS}(t_S)) \end{array} \right. \tag{12}$$

We can remark that we choose to exactly define the initial and end states, but a very small extention of the theory could include all cases.

Now we can give the optimality conditions for such a problem.

Let us set the hamiltonians for all (i,j) in E :

$$H_{ij}(y,u,p) = \ell_{ij}(y,u) + f_{ij}(y,u)^T p \tag{13}$$

And then the Pontryaguin principle gives :

$$\left\{\begin{array}{rcl} \dot{y}_{ij} & = & \dfrac{\partial H_{ij}}{\partial p}(y_{ij},u_{ij},p_{ij}) \\ \dot{p}_{ij} & = & -\dfrac{\partial H_{ij}}{\partial y}(y_{ij},u_{ij},p_{ij}) \\ u_{ij} & = & \underset{w}{\text{Argmin}}\, H_{ij}(y_{ij},w,p_{ij}) \end{array}\right. \tag{14}$$

After some calculation, we obtain the transversality conditions :

$$\left\{\begin{array}{l} p_{LS}(t_L), p_{SE}(t_E), p_{SB}(t_B) \text{ free} \\ p_{LS}(t_S) = \dfrac{\mathrm{d}\Phi_S^1}{\mathrm{d}y}^T p_{SE}(t_S) \\ \qquad\qquad + \dfrac{\mathrm{d}\Phi_S^2}{\mathrm{d}y}^T p_{SB}(t_S) \end{array}\right. \tag{15}$$

These equations generalize our method to any multi-phase scenario. This feature will be introduced later in the software.

4 Application to a single stage launcher trajectory

We take the example of a single-stage RLV, like the Venture Star concept. For simplification purpose, we choose a flat earth model with no atmosphere and a uniform gravity field. The vehicle is expected to reach a given altitude with a given horizontal velocity, while minimizing the propellant consumed during the mission (the thrust and the mass flow-rate are constant). The final time is free. We will first study this problem with no other constraint, then we will add a path constraint which is a non-linear function of state and control.

The control variable is the pitch θ, and we assume that the thrust is colinear to the vehicle's axis (so the pitch defines the angle between the thrust and the horizontal axis).

h is the altitude, $\vec{V}$ the velocity composed by his projections V_x, V_h, $\vec{T}$ is the thrust vector, θ is the pitch and α the angle of attack.

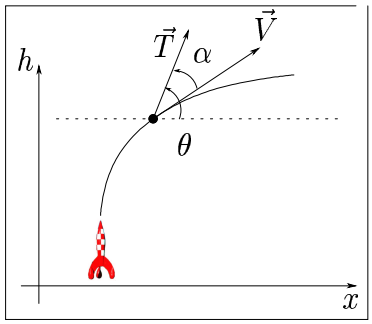

Figure 6: Notation used for the 2D trajectory

4.1 Boundary conditions

	Initial conditions	Final conditions
h	$0[m]$	$250[km]$
x	$0[m]$	free
V_h	$10[ms^{-1}]$	$0[ms^{-1}]$
V_x	$0[ms^{-1}]$	$7.75 \times 10^3[ms^{-1}]$

The launcher lifts-off vertically at altitude zero. We set an initial vertical speed greater than zero in order to avoid singularity issues. At the end of the mission, the vehicle should reach an altitude of 250 km with a horizontal velocity corresponding to the injection into a circular orbit with this altitude.

4.2 Launcher characteristics

$$\begin{array}{rcl} m_0 & = & 10^6[kg] \\ \text{Isp} & = & 420[s] \\ (T/W)_0 & = & 1.2 \\ q & = & m_0(T/W)_0/\text{Isp} \\ & = & 2857.1[kgs^{-1}] \\ T & = & qg_0\text{Isp} = m_0g_0(T/W)_0 \\ & = & 1.1810^7[N] \end{array}$$

m_0 is the total initial mass, Isp is the specific impulse, q is the mass flow rate, T the thrust, and $(T/W)_0$ is the initial thrust-to-weight ratio.

4.3 Dynamics of problem

The final time t_f, which is free, is a parameter of the problem. We choose to consider it as

an additional state variable (with a null dynamic). Using the normalized time $s = t/t_f$ (comprised between 0 and 1), we can use the following equations of dynamics :

$$\begin{aligned}
\frac{dx}{ds} &= t_f V_x \\
\frac{dh}{ds} &= t_f V_h \\
\frac{dV_x}{ds} &= \frac{t_f T sin(\theta)}{m} \\
&= \frac{t_f g_0 (T/W)_0 sin(\theta)}{1 - (T/W)_0 s t_f / Isp} \\
\frac{dV_h}{ds} &= \frac{t_f T cos(\theta)}{m} - t_f g_0 \\
&= t_f g_0 \left(\frac{(T/W)_0 cos(\theta)}{1 - (T/W)_0 s t_f / Isp} - 1 \right) \\
\frac{dt_f}{ds} &= 0
\end{aligned}$$

4.4 Cost

Since the ejected mass-flow rate is constant, the objective of minimizing the propellant consumed is equivalent to minimizing the flight time, so we take the parameter t_f as the cost function.

4.5 Path constraint

In the constrained problem, we choose $|Q\alpha| \leq (Q\alpha)_{\max}$ as a path constraint.

Q is the dynamic pressure :

$$Q = \frac{1}{2}\rho V^2 \tag{16}$$

with the exponential model of atmospheric density :

$$\rho = \rho_0 e^{-h/h_{ref}}$$

α is the angle of attack, given by :

$$\alpha = \arctan \frac{V_h}{V_x} - \theta \tag{17}$$

On a launcher's trajectory in the atmosphere, the $Q\alpha$ product is is representative of the transverse aerodynamic forces. In our example, we actually do no take into account the aerodynamic forces in the equations of dynamics. Nevertheless, this path constraint is interesting to study as a test case because it can be predicted that the optimization process should induce very small values of the angle of attack around the peak value of dynamic pressure. Moreover, $Q\alpha$ is a mixed constraint function, i.e. it is a non-linear function of state and control vectors. In our example, we have arbitrarily chosen $(Q\alpha)_{\max} = 500$ which is a very small value compared to the peak value of $Q\alpha$. This allows us to verify the efficiency of the method on the constrained problem.

4.6 Numerical application

To solve the constrained problem, we compute the optimality conditions for three values of ε : 1,10^{-2} and 10^{-5}. Figure 7 shows the constrained and unconstrained trajectories. We see that the constrained one is the steepest.

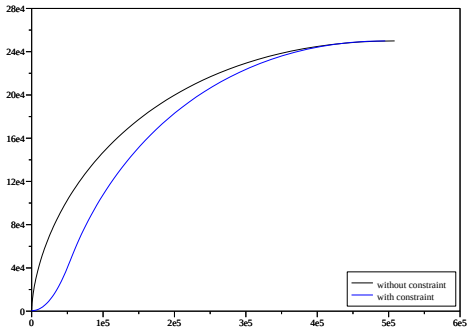

Figure 7: Trajectory h [m] $= f(x)$ [m]

The Figure 8 shows the control variable θ as a function of time. In the unconstrained trajectory, the pitch decreases continuously.

On the contrary, the control history of the constrained trajectory includes an increase phase We notice that this phase is roughly the one on which $Q\alpha$ has the maximum value allowed by the constraint. The explanation comes from the angle of attack history: as it was predicted, the angle of attack is reduced to a very small value (near zero) during this phase, in order to have a low $Q\alpha$ when the dynamic pressure is high. This is the reason of the decrease of the pitch angle.

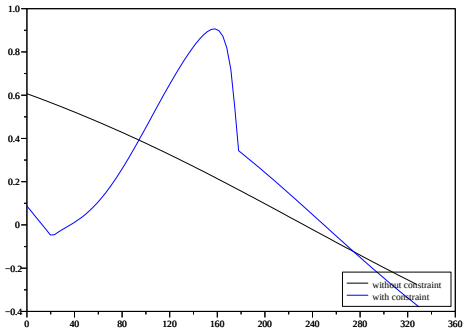

Figure 8: Pitch θ [rad] as a function of time [s]

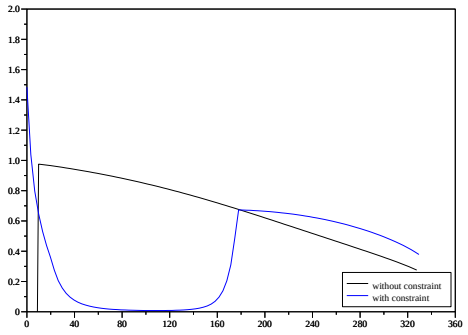

Figure 9: Angle of attack α [rad] as a function of time [s]

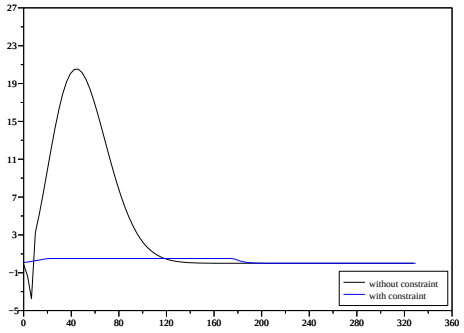

Figure 10: $Q\alpha$ [kPa.rad] as a function of time [s]

We also verify that the introduction of the path constraint increases the propellant consumption, i.e. the mission time, with reference to the unconstrained problem : the final time is of 327.574 seconds for the unconstrained problem and 329.472 seconds for the constrained one.

The numerical results show the efficiency of the method for both the unconstrained problem and the constrained one.

Conclusion

This paper presents a preliminary work on an interior point optimization method designed to solve RLV trajectory optimization problems. Two different aspects of the method have been presented : The interior point formulation of the optimal control problem itself, as well as the solving of the optimality conditions using a RK discretized method with mesh refinement.

The efficiency of the technique has been verified on simple, mono-phase test case. In addition, the equations for general multi-arc trajectory problems have been given. The next step of this work will be to validate this

approach with more realistic RLV trajectory optimization problems.

References

[1] P. Armand, J.Ch. Gilbert, and S. Jan-Jégou. A feasible BFGS interior point algorithm for solving convex minimization problems. *SIAM Journal on Optimization*, 11:199–222 (electronic), 2000.

[2] J.T. Betts. *Practical methods for optimal control using nonlinear programming*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.

[3] J.F. Bonnans, J. Ch. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization: theoretical and numerical aspects*. Universitext. Springer-Verlag, Berlin, 2002. To appear.

[4] J. C. Butcher. *The numerical analysis of ordinary differential equations*. A Wiley-Interscience Publication. John Wiley & Sons Ltd., Chichester, 1987. Runge-Kutta and general linear methods.

[5] R.H. Byrd, J.Ch. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185, 2000.

[6] Anthony V. Fiacco and Garth P. McCormick. *Nonlinear programming*. John Wiley and Sons, Inc., New York-London-Sydney, 1968.

[7] L. G. Hačijan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244(5):1093–1096, 1979.

[8] W. Hager. Runge-Kutta methods in optimal control and the transformed adjoint system. *Numer. Math.*, 87(2):247–282, 2000.

[9] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2002. Structure-preserving algorithms for ordinary differential equations.

[10] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993. Nonstiff problems.

[11] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1996. Stiff and differential-algebraic problems.

[12] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[13] S.J. Wright. *Primal-dual interior-point methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[14] S.J. Wright. On the convergence of the Newton/log-barrier method. *Mathematical Programming*, 90:71–100, 2001.

[15] S.J. Wright and F. Jarre. The role of linear objective functions in barrier methods. *Mathematical Programming*, 84:357–373, 1999.