Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Improving Trajectory Constraints Processing in some Optimal Control Algorithms ⋆

J.B.Coulaud * **G.Campion**

* *CESAME, Euler, 4-6 Av. G.Lemaître, B1348 Louvain-La-Neuve, Belgium, jean-baptiste.coulaud@uclouvain.be*

Abstract: This paper shows how the treatment of trajectory constraints in some recent software packages like the Nonlinear Trajectory Generation (NTG) can be improved by taking into account some structural properties of the initial optimal control problem that are lost in the transformation of the problem into an optimization one. The proposed method leads to a smaller complexity and a better accuracy.

1. INTRODUCTION

With the increasing performances of computers, it is natural to expect that optimal and predictive control can be used in real time with systems that are faster and faster. Nevertheless, in its original expression, solving accurately a nonlinear constrained optimal control problem remains very consuming in computation ressources: indirect shooting methods are often impossible to use in a first step, whereas direct methods based on the Hamilton-Jacobi formulation of the problem require less but often too much time and are less accurate (see Trélat [2005]).

Several software packages have been designed in the past few years to bring the efficiency of constrained optimal control to nonlinear systems with fast dynamics, like robots. One of the most efficient is NTG (for Nonlinear Trajectory Generation, see Milam *et al.* [2000] and Petit *et al.* [2001]); it uses a spline description of the trajectories and reduces the optimal control problem to an optimization one, taking advantage of dynamics inversion based on differential flatness (on this topic see Fliess [1995]). However in this transformation, a part of the initial information about the functions describing the trajectory constraints is lost: this can affect the performances of the algorithm. This paper aims at describing and analysing an alternative treatment of trajectory constraints, which is suited to an optimal control approach.

Section 2 briefly describes the basic principles of NTG and stresses on the pros of this method but also show a weakness in the constraints treatment. Section 3 proposes an alternative to take advantage of the smoothness of trajectory constraints, and gives two mathematical arguments that justify this approach. Section 4 describes how the complexity is reduced with this method even with increased accuracy. Section 5 gives some simulation experiments to confirm and precise the theoretical results. Section 6 analyses briefly some difficulties that can arise with the proposed algorithm and gives some ideas of solution. Finally, Section 7 summarizes these results and gives directions for complementary work.

2. BRIEF OVERVIEW OF NTG

2.1 Description of the approach

For a nonlinear system

$$\dot{x} = h(x,u), \quad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,$$

an optimal control problem is considered, with a nonlinear cost function $J(x,u)$ to minimize under constraints of the form:

$$\begin{array}{rcl} l_i \leq & \psi_i(x(t_0),u(t_0)) & \leq u_i \ \text{(initial)} \\ l_t \leq & \psi_t(x(t),u(t),t) & \leq u_t \ \text{(path)} \\ l_f \leq & \psi_f(x(t_f),u(t_f)) & \leq u_f \ \text{(final)} \end{array} \quad (1)$$

If the system is differentially flat, there exist a vector y of m outputs, an integer q and a function ϕ s.t.:

$$(x(t),u(t)) = \phi(y,\dot{y},...,y^{(q)}). \quad (2)$$

Then, it is possible to map the cost function J and the constraints (1) to the output space. Consequently, if the optimal control problem is parametrized in function of the outputs, the number of unknowns significantly decreases and it is not necessary to solve the ODE of the system. This last fact reduces dramatically the complexity of the problem.

As this new formulation of the optimal control induces the use of the derivatives of the unknowns y, it is necessary to describe them with smooth functions. In NTG, the outputs y are parametrized in terms of B-splines curves, that are piecewise polynomials (see Boor [1978]). After this parametrization, the unknowns of the problem are the coordinates Y in the spline basis, and every function of the problem only depends on Y and t. A complete description of NTG can be found in the Thesis of Milam (2003), but the idea of the procedure is the following.

2.2 From optimal control to optimization

Let us consider an optimal control problem on the horizon $[0,T]$. As the paper focuses on trajectory constraints, only this type of constraints with upper bounds is considered in the following. Hence the set of constraints (1) is described

⋆ This paper presents research results of the Belgian Programme on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office. The scientific responsability rests with its authors.

by: $C(x,u,t) \leq 0, \quad \forall t \in [0,T]$, with a smooth function C. In the NTG procedure, the cost $J(x,u)$ and the constraint $C(x,u,t)$ are reformulated

- in a first step, in $J'(y,\dot{y},...,y^{(q)})$ and $C'(y,\dot{y},...,y^{(q)},t)$ by introducing equation (2) (where y are the flat outputs),
- in a second step, in $\tilde{J}(Y)$ and $\tilde{C}(Y,t)$ where Y are the parameters of the spline description of the outputs.
- in a third step, a finite set of collocation points
$$0 = t_0 < t_1 < ... < t_k = T \tag{3}$$
is defined and the constraint C is lastly reformulated in a finite set of constraints
$$\tilde{C}_i(Y) = \tilde{C}(Y,t_i) \leq 0, \quad \forall i \in \{0,...,k\}. \tag{4}$$

The procedure has lead to an optimization problem:
$$\begin{array}{l} \min\limits_{Y} \ \tilde{J}(Y) \\ \quad \tilde{C}_i(Y) \leq 0, \quad \forall i \in \{0,...,k\} \end{array} \tag{5}$$
which only depends on the set of variables Y.

2.3 Few words about the nonlinear solver

NTG transmits problem (5) to the nonlinear solver NPSOL, which is based on a SQP method. Basically, SQP is divided in two levels of iterations:

- major iterations consist in a linesearch in a given search direction p,
- minor iterations solve a QP problem to find the search direction p involved in the major iterations.

The QP problem solved by NPSOL is the following:
$$\begin{array}{l} \min\limits_{p} \ \frac{1}{2} p^T H^*(Y) p + \nabla_Y \tilde{J}(Y)^T p \\ \quad \tilde{C}_i(Y) + \nabla_Y \tilde{C}_i(Y)^T p \leq 0, \quad \forall i \in \{0,...,k\} \end{array} \tag{6}$$
where Y is the current iterate at the start of a major iteration, H^* is a positive-definite quasi-Newton approximation to the Hessian,and $\nabla_Y \tilde{J}$ and $\nabla_Y \tilde{C}_i$ represent the gradients of the functions $\tilde{J}$ and $\tilde{C}_i$.

2.4 Main advantages

The spline parametrization of the trajectories is a very appropriate description for an optimal control approach:

- first, most of real trajectories are piecewise smooth, which is obviously a property of splines,
- second, as a consequence of the previous point, for the same accuracy, one will need less parameters to decribe a trajectory with splines than with a sampling method, because some properties of the trajectory are induced by definition in a spline parametrization, whereas they need lots of points to appear in a sampling description,
- third, in the case of differentially flat (even partially) systems, it allows to avoid completely (or partially) the resolution of ODEs, without any loss of accuracy, and again with quite few parameters.

Another advantage of the NTG approach is practical: transforming a nonlinear optimal control problem into an optimization one allows the use of an existing software (like NPSOL) to solve the problem instead of implementing the whole procedure.

2.5 Some lost informations about constraints

However, the second advantage of NTG previously mentionned, induces a loss of informations about trajectory constraints. Two types of informations are lost in formulation (5):

- first, as the time is sampled, it is not sure in general that the constraint C will be satisfied between two sample times.
- second, what is transmitted to the nonlinear otpimization software is the set of constraints $\tilde{C}_i$. This means that the algorithm handles every constraint independently whereas they are computed from the same analytical function $\tilde{C}$. As $\tilde{C}$ is very often smooth this loss of information can result in a waste of time as it will be shown in the following sections

3. ANOTHER ALGORITHMIC APPROACH

3.1 Description of the approach

The same framework is kept: trajectories are parametrized by splines, and we also take advantage of differential flatness properties as in NTG. But instead of considering, like in NTG, $k+1$ independent constraints we take benefit from the fact that all those constraints derive from a unique analytical function and will only take into account the maximum of that function, or several local maxima. From a complexity point of view, this can reduce significantly the number of computations for one minor iteration. Obviously, it must be shown that the efficiency of each iteration is not deteriorated. This is done in Section 3.2 by some smoothness considerations of the initial constraint function $\tilde{C}$, that are experimentally confirmed in Section 5.

The idea of the approach is to keep the problem under the following form:
$$\begin{array}{l} \min\limits_{Y} \ \tilde{J}(Y) \\ \quad \tilde{C}(Y,t) \leq 0 \end{array} \iff \begin{array}{l} \min\limits_{Y} \ \tilde{J}(Y) \\ \quad \max\limits_{t} \tilde{C}(Y,t) \leq 0 \end{array} \tag{7}$$
and adapt this formulation at each iteration. This last problem involves only one constraint, but it can be difficult to handle it. To do so, we propose to keep the main concepts of SQP with major and minor iterations with a change in the minor iteration.

Let Y the current iterate. For this iterate let
$$t^* = t_j = \max_{i \in \{0,...,k\}} \tilde{C}(Y,t_i) = \max_{i \in \{0,...,k\}} \tilde{C}_i(Y).$$
Then the minor iteration consists in finding a search direction p by solving the following QP problem:
$$\begin{array}{l} \min\limits_{p} \ \frac{1}{2} p^T H^*(Y) p + \nabla_Y \tilde{J}(Y)^T p \\ \quad \tilde{C}(Y,t^*) + \nabla_Y \tilde{C}(Y,t^*)^T p \leq 0 \end{array} \tag{8}$$
As there is only 1 constraint instead of $k+1$, the complexity of this QP-problem is much less than for problem (6). The key point is here, in the reduction of the number of constraints taken into account at each iteration.

3.2 Two justifications about this approach

First argument
One could argue that, even if the proposed method is a

possible way to solve the problem, it can be less efficient than computing the gradient of the constraint at each collocation point. Let t_j a collocation point. A Taylor expansion immediatly shows that the gradient $\nabla_Y \tilde{C}(Y, t_j)$ is representative of the average gradient in the neighborhood of t_j. Consider for instance $2l+1$ collocation points uniformly distributed (with time interval $\delta\tau$), on a symmetric interval around a given t_j. It is easy to see, that the average of the gradient $\nabla_Y \tilde{C}(Y, t)$ evaluated at the $2l+1$ collocation points is approximated at first order in $\delta\tau$ by the gradient $\nabla_Y \tilde{C}(Y, t_j)$:

$$\frac{1}{2l+1} \sum_{j=-l}^{l} \nabla_Y \tilde{C}(Y, t_{i+j}) = \nabla_Y \tilde{C}(Y, t_i) + O(l^2 \delta\tau^2)) \quad (9)$$

This means that the directions of the gradient at t_j and of the average value differ only at the second order in $\delta\tau$. The evaluation of the gradient at t_j reflects with a good accuracy the values of the gradients in the neighborhood. This holds also near a local maximum, at t^*. As the time instants for which the constraint is the closest to its bound are necessarily in the neighborhood of the maximum, this property shows that taking $\tilde{C}(Y, t^*)$ into account handles also the "constraints" $\tilde{C}_i(Y)$ that are the most likely to be violated.

Second argument
Let Y the current iterate and $Y_+ = Y + \delta Y$ the iterate at the end of the current iteration.

If one decided to decrease the value of the maximum $\tilde{C}(Y, t^*)$ at the current iteration, the best way to do it with a first order Taylor expansion, is to modify the parameters with a correction in the opposite direction of the gradient of the constraint at $t^* = t_j$. This means that the correction is of the form:

$$\delta Y = -\alpha \nabla_Y \tilde{C}(Y, t^*) \quad (10)$$

where α is a positive scalar that gives the length of the step for the current iteration.

From an optimization point of view, decreasing one constraint does not necessarily interfere with another, because the influence of one constraint on another is not supposed to be predictible. For a trajectory constraint it is different: decreasing $\tilde{C}(Y, t^*)$ in this way decreases $\tilde{C}$ on a neighborhood of t^*. It is obvious on the following Taylor expansion:

$$\begin{aligned} \tilde{C}(Y_+, t_i + \delta t) &= \tilde{C}(Y, t) + \nabla_Y \tilde{C}(Y, t_i)\delta Y \\ &\quad + \frac{\partial \tilde{C}}{\partial t}(Y, t_i)\delta t + O(\delta Y^2 + \delta t^2) \\ &= \tilde{C}(Y, t_i) + \nabla_Y \tilde{C}(Y, t_i)\delta Y \\ &\quad + O(\delta Y^2 + \delta t^2) \\ &= \tilde{C}(Y, t_i) - \alpha \left(\nabla_Y \tilde{C}(Y, t_i) \right)^2 \\ &\quad + O(\alpha^2 + \delta t^2) \end{aligned} \quad (11)$$

This equality takes into account the fact that $\tilde{C}(Y, t_i)$ is a local maximum, which implies that $\frac{\partial \tilde{C}}{\partial t}(Y, t^*)$ vanishes. Provided that $\nabla_Y \tilde{C}(Y, t^*) \neq 0$, which is true most of the time, the consequence of this result is that, at first order in δt, the value of $\tilde{C}$ decreases on a neighborhood of t_i for an adequate choice of α. Then, the local maximum of $\tilde{C}$ can be slightly moved, but whatever, it is less than its value at the previous iteration.

We do not say here that $-\alpha \nabla_Y \tilde{C}(Y, t^*)$ is the correction computed in practice, but it is one more clue to show that handling the constraint near its maximum is reasonable in the QP problem, because the constraints are linearly approximated using their gradients.

4. COMPLEXITY COMPARISONS

This section gives two comparisons of complexity. A first comparison between the NTG approach and the approach of the paper in Section 4.1. As the main point of the paper is not reduced to NTG, we have add a second comparison using the same properties for a gradient descent algorithm in Section 4.2.

In order to achieve a quantitative comparison between the strategy proposed in the paper and the one that considers independent constraints, it is necessary to make some assumptions:

- the complexity of evalutating $\tilde{C}$ is denoted by K,
- the complexity of evalutating the derivative of $\tilde{C}$ w.r.t. one variable is assumed to be equal to K also,
- the number of independent splines parameters used to describe y, *i.e.* the size of Y, is denoted by n,
- the number of collocation points, which is also the number of discrete constraints $\tilde{C}_i$, is denoted by p.

4.1 First Comparison: complexity of a minor iteration

Here we compare the complexity of a minor iteration in NTG using NPSOL and a minor iteration proposed in the paper. The main difference is in the number of constraints: NPSOL must deal with p constraints whereas problem (8) has only 1 constraint. An adaptation of the complexity analysis made in Boyd [2004] about convex QP algorithms, considering that n and p are of the same orders of magnitude, leads to a theoretical bound proportional to $\sqrt{p}(n+p)^3$ in the case of NTG and $(n+1)^3$ for problem (8). As mentionned in Boyd [2004], the factor $\sqrt{p}$ is not very relevant in practice and can be eliminated. Even with this better bound, the ratio between the two bounds is:

$$\left(\frac{n+p}{n+1} \right)^3$$

If $p \approx n$ then the ratio is approximately 8.

This comparison shows very well the advantage of our method for one iteration, but it does not take into account the efficiency of every iteration, which can interfere with the complexity of the whole procedure. That is why we propose a second comparison on a simpler algorithm, but with a more complete analysis.

4.2 Second comparison: complexity of a gradient descent

In this section, a basic gradient descent with constant step is applied to both strategies (with p constraints or only 1 constraint). The algorithm is described in Fig.1.

The difference between the two strategies is only in the choice of δ_Y:

Data: constraint function $\tilde{C}(\cdot)$
Result: Y s.t. $\tilde{C}(Y,t) \leq B$
Initialization: first guess $Y = Y_0$;
while $\exists t, \tilde{C}(Y,t) > B$ **do**
  find t^*, s.t. $\tilde{C}(Y,t^*) = \max_t \tilde{C}(Y,t)$;
  determine δY ;
  $Y \leftarrow Y - \alpha \delta Y$;
end

Fig. 1. Algorithm decreasing a constraint under a bound B.

- in the first one, denoted A_1, $\delta Y = \frac{\nabla_Y \tilde{C}(Y,t^*)}{\|\nabla_Y \tilde{C}(Y,t^*)\|}$,
- in the second, denoted A_2, $\delta Y = \frac{\sum_{i\in I} \nabla_Y \tilde{C}_i(Y)}{\|\sum_{i\in I} \nabla_Y \tilde{C}_i(Y)\|}$,

where I is the set of constraints that are not satisfied.

Let q the number of points for which the constraint C is not satisfied in the current iteration.

- For algorithm A_1, the smoothness of the constraint is taken into account, and the number of operations for one iteration is: $N_1 = Kp + Kn$. The term Kp corresponds to the evaluation of $\tilde{C}$ at the p collocation points, and the term Kn corresponds to the evaluation of the gradient $\nabla_Y \tilde{C}$ (which has n components) at t^*.
- For algorithm A_2, the constraint is discretized in p independent constraints, and the number of operations for one iteration is: $N_2 = Kp + Knq$. The first term is the same as in N_1 and the second term nKq corresponds to the evaluation of $\nabla_Y \tilde{C}$ at q collocation points.

Let us assume that the first algorithm converges within r_1 iterations, and the second within r_2 iterations. Let $\bar{q}$ the mean value of q over all r_2 iterations. Then the ratio of the two numbers of computations is:

$$R = \frac{r_1(Kp + Kn)}{r_2(Kp + Knq)} = \frac{r_1(p+n)}{r_2(p+n\bar{q})} \tag{12}$$

5. NUMERICAL EXPERIMENTS

5.1 Framework of the tests

All the simulations have been performed with MATLAB 7.4. In every simulation, a single output $y(t) \in \mathbb{R}$ is considered, and a constraint of the form $C(y(t), \dot{y}(t)) \in \mathbb{R}$ is treated. To obtain statistics over the properties which are examined, C is a random polynomial function of two variables of degree [1] d. Such a polynomial has $\frac{(d+1)(d+2)}{2}$ coefficients, which are chosen smaller than 1 in absolute value. Instead of using B-splines, the polynomials used in the test are Bézier curves, that are known to have very similar properties.

5.2 About the "first argument"

This paragraph refers to Section 3.2.1. A large number of functions $C(y, \dot{y})$ are taken at random and for a given C, several Y are taken at random. Then the gradient $\nabla_Y \tilde{C}$ is computed at a local maximum of the function and on a neighborhood. Then the direction of the vectors involved in the two members of equation (9) are compared, using the Euclidian distance with the computed unit vectors. The algorithm used is reported in Fig.2.

Data: the degree d, the order s and a lag l
Result: $\bar{\delta}$ average value of the distance δ between

$$\frac{\sum_{j=-l}^{l} \nabla_Y \tilde{C}(Y,t_{i+j})}{\|\sum_{j=-l}^{l} \nabla_Y \tilde{C}(Y,t_{i+j})\|} \text{ and } \frac{\nabla_Y \tilde{C}(Y,t_i)}{\|\nabla_Y \tilde{C}(Y,t_i)\|}$$

near maxima
Initialization: $D = \emptyset$;
for $k \leftarrow 1:1000$ **do**
  take $C(y,\dot{y})$ at random ;
  for $j \leftarrow 1:100$ **do**
    take Y at random;
    find a local max of $\tilde{C}(Y,t)$ for $t \in (0,1)$;
    compute δ;
    $D \leftarrow D \cup \{\delta\}$;
  end
end
$\bar{\delta} \leftarrow \text{mean}(D)$;

Fig. 2. Algorithm to test statistically the argument of Section 3.2.1.

As δ is a distance between two unit vectors, it is equivalent to an angle $\alpha \in [0,\pi] = [0°, 180°]$ between those two vectors. This other value, may be more meaningful than δ, is also reported in the Table 1, in function of the degree d of C, the order of the spline s and the lag l. The number of collocation points taken here is $p = 100$. The ratio $\frac{2l+1}{p}$, which represents the density - w.r.t. the horizon of the control - of the range where $\nabla \tilde{C}$ is evaluated, is significant, and thus is also mentionned.

d	s	l	$\frac{2l+1}{p}$	$\bar{\alpha}(°)$	$\bar{\delta}$
5	4	4	0.09	2.49	0.042
5	4	7	0.15	3.88	0.065
5	4	10	0.21	5.92	0.098
5	6	4	0.09	11.0	0.17
5	6	7	0.15	13.6	0.22
5	6	10	0.21	15.5	0.25
5	8	4	0.09	18.2	0.28
5	8	7	0.15	20.1	0.31
5	8	10	0.21	21.3	0.34
10	4	4	0.09	3.03	0.050
10	4	7	0.15	4.85	0.079
10	4	10	0.21	6.57	0.11
10	6	4	0.09	14.5	0.22
10	6	7	0.15	15.7	0.24
10	6	10	0.21	15.8	0.25
10	8	4	0.09	23.0	0.33
10	8	7	0.15	23.5	0.36
10	8	10	0.21	23.1	0.37

Table 1. Statistics about "the 1st argument"

Several remarks can be made about the results reported in Table 1. As expected:

- $\frac{\sum_{j=-l}^{l} \nabla_Y \tilde{C}(Y,t_{i+j})}{\|\sum_{j=-l}^{l} \nabla_Y \tilde{C}(Y,t_{i+j})\|}$ is not very far from $\frac{\nabla_Y \tilde{C}(Y,t_i)}{\|\nabla_Y \tilde{C}(Y,t_i)\|}$;
- the angle between the two vectors increases with the lag l;

[1] The degree of a polynomial of two variables x and y is the maximum sum $i+j$ of terms $x^i y^j$.

Moreover, this angle increases also with the order of the spline n, and a bit with the order of the polynomial constraint C: this is because polynomials with high order have much more oscillations, which increases the curature near extrema and then reduces the impact of the argument.

It is also important to note that in practice, conditions are probably often better than the ones of the algorithm presented here, because a random constraint with a random guess of Y can be much oscillating than the "average" situation of an algorithm after the first iterations.

5.3 About the "second argument"

This paragraph refers to Section 3.2.2. A large number of functions $C(y, \dot{y})$ are taken at random and, for a given C, a sequence of Y (the coordinates of y in the Bézier functions basis) is computed such that the maximum decreases at each iteration. This is done using equation (11). The relevance of the invoked argument is tested by counting the number of collocation points located in the neighborhood of the maximum where the constraint function C decreases. The algorithm is described in Fig. 3. The results of the

Data: the order of the splines s, the degree d
Result: $\overline{c}$ average value of the number of collocation points where C decreases
Initialization: $N = \emptyset$;
for $k \leftarrow 1 : 1000$ **do**
  take $C(y, \dot{y})$ at random ;
  while $\max_{t \in [0,T]} C(\tilde{Y}, t) > B$ **do**
    $Y_+ \leftarrow Y - \alpha \nabla_Y C(Y, t^*)$;
    count the number c of collocation points t near t^* s.t. $C(Y_+, t) < C(Y, t^*)$;
    $N = N \cup \{c\}$;
    $Y = Y_+$;
  end
end
$\overline{c} \leftarrow \text{mean}(N)$;

Fig. 3. Algorithm to test statistically the argument of Section 3.2.2.

simulations are reported in Table 2.

d	s	$\frac{\overline{c}}{p}$
5	4	0.28
5	6	0.21
5	8	0.17
10	4	0.21
10	6	0.16
10	8	0.12

Table 2. Statistics about "the 2nd argument"

The conclusion of this experiment is very straightforward: it can be observed that the proportion $\frac{\overline{c}}{p}$ - of collocation points influenced by the gradient of C at its maximum - decreases with s and d, but remains non negligible in any case.

5.4 Comparison of the two algorithms

This experiment refers to Section 4.2. Its aim is to evaluate statistically the ratio R. Algorithm presented in Fig.1 is

n	p	$\frac{\overline{r}_1}{\overline{r}_2}$	$\overline{q}$	R
4	25	1.1	7.4	0.51
4	50	1.1	15.0	0.46
4	100	1.1	29.4	0.44
6	25	1.2	6.8	0.47
6	50	1.2	13.2	0.43
6	100	1.1	25.9	0.39
8	25	1.2	6.6	0.44
8	50	1.2	12.6	0.38
8	100	1.2	25.3	0.34
11	25	1.3	6.5	0.40
11	50	1.3	12.6	0.33
11	100	1.3	24.9	0.29
16	25	1.4	6.5	0.38
16	50	1.5	12.8	0.29
16	100	1.4	24.9	0.25

Table 3. Statistics to compare the complexity of algorithms A_1 and A_2.

applied with algorithms A_1 and A_2 a lot of times as shown in Fig.4. In this experiment, $d = 5$, because there is no important influence of the degree of the polynomial C.

Data: order n, number of points p
Result: ratio R and $\overline{r}_1$, $\overline{r}_2$, $\overline{q}$, average values of r_1, r_2, q
Initialization: $R_1 = R_2 = Q = \emptyset$;
for $k \leftarrow 1 : 100$ **do**
  take $C(y, \dot{y})$ at random ;
  for $j \leftarrow 1 : 10$ **do**
    take a first guess Y_0 at random;
    run A_1;
    compute r_1;
    $R_1 \leftarrow R_1 \cup \{r_1\}$;
    run A_2;
    compute r_2 and q;
    $R_2 \leftarrow R_2 \cup \{r_2\}$;
    $Q \leftarrow Q \cup \{q\}$;
  end
end
$\overline{r}_1 \leftarrow \text{mean}(R_1)$;
$\overline{r}_2 \leftarrow \text{mean}(R_2)$;
$\overline{q} \leftarrow \text{mean}(Q)$;
compute R from R_1, R_2, and Q;

Fig. 4. Algorithm to test statistically the complexity of A_1 and A_2.

The advantage of the proposed strategy is obvious in Table 5.4: the maximum of the ratio R is 0.5 but it decreases significantly with n and p till 0.25 for $n = 16$. This means that this part of an optimization can be in average 4 times faster. Actually, it can be much more, because 16 points of control for a spline is not a maximum, and if there are several outputs, the number of variables is also multiplied.

Fig. 5 shows an example comparing the evolution of a function until it passes under its upper bound (equal to one) represented by the blue horizontal line, for the two algorithms. The evolution of the function is from red to blue, and the green line represents the final situation of the function. It can be notice that algorithm A_1 gives results very close to algorithm A_2.

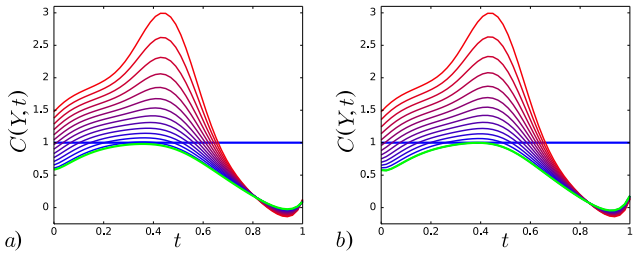

Fig. 5. The evolution of a constrained function for algorithms A_1 (left) and A_2 (right).

6. SOME DIFFICULTIES AND IDEAS OF SOLUTION

6.1 Competition between maxima

A first problem which can appear is a competition between several local maxima. This can induce numerous switchings of the localization of the global maximum during the algorithm. A possible improvement is to detect several or all the local maxima that do not satisfy a constraint and take them into account. This would obviously increase the complexity of each iteration of the algorithm, but would remain more accurate and less expensive than the "classical" optimization approach, because in average, this should decrease the ratio $\frac{r_1}{r_2}$.

A second inconvenience related to the previous, is due to the nature of an optimal control problem: very often, the optimal solution under constraints will presents ranges where one constraint is constant. Then the algorithm will be in the previous situation. A way to deal with that is to detect when such a range appears, and use dynamic breackpoints to match that range and use an adapted parametrization of the outputs on that range (like a low order spline).

6.2 Following the maxima

In order to improve the precision of the solution, one would like to increase the number of collocation points, but in the previous proposal of algorithm, it is necessary to evaluate the constraint at each point, which slows down the algorithm. To improve the efficiency of the algorithm, it is possible not to do an accurate localization at each iteration but only for few of them. Then, when a maximum is localized, at t^* for Y, it is possible to have an approximate localization $t^*_+ = t^* + \delta t$ of the maximum corresponding to $Y_+ = Y + \delta Y$. As $\tilde{C}(Y, t^*)$ is a maximum, $\frac{\partial \tilde{C}}{\partial t}(Y, t^*) = 0$. What is expected is that $\frac{\partial \tilde{C}}{\partial t}(Y_+, t^*_+) = 0$. At first order, the Taylor expansion of $\frac{\partial \tilde{C}}{\partial t}$ gives:

$$\frac{\partial \tilde{C}}{\partial t}(Y_+, t^*_+) = 0 + \frac{\partial^2 \tilde{C}}{\partial t^2}\delta t + \nabla_Y \tilde{C}\delta Y + O(\delta t^2 + \delta Y^2)$$

As this quantity should vanish to ensure being at a maximum, it leads to:

$$\delta t \approx -\nabla_Y \tilde{C}(Y, t^*)\delta Y \;/\; \frac{\partial^2 \tilde{C}(Y, t^*)}{\partial t^2}.$$

In this perspective, it can become advantageous to detect all local maxima and follow them approximately by this way during several iterations before checking accurately their positions again.

7. CONCLUDING REMARKS AND FUTURE WORK

In this paper it has been shown how it is possible to reduce the complexity of some algorithms suited for real time optimal control and thus to improve their performances. Nevertheless, for instance in the case of NTG, it induces probably a complete reimplementation of the optimization routines, because the classical optimization formulation is lost and the proposal of this paper cannot be treated by an optimization software like NPSOL.

Furthermore, some side effects induced by the method should be taken into account in order not to lose its efficiency, as mentionned in Section 6. But, again, a good treatment of these specificities should lead to an improvement of the accuracy.

An important concluding remark is that the result presented in this paper should be much more significant for systems governed by PDE like presented in Petit [2002]: indeed, in that case, the number of collocation points can increase dramatically with the dimension of the problem, because it is a discretisation of a higher dimensional space than a simple range of $\mathbb{R}$. As shown by the statistics and the theoretical results of this paper, increasing the number of collocation points p - and thus $\overline{q}$ - decreases the ratio R defined in equation (12).

REFERENCES

E.Trélat. Contrôle optimal : théorie & applications. *Vuibert, Collection "Mathématiques Concrètes"*, 2005.

M.B.Milam, K.Mushambi, and R.M.Murray. A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems. *IEEE Conference on Decision and Control*, 2000.

N.Petit, M.B.Milam, R.M.Murray. Inversion Based Constrained Trajectory Optimization. *5th IFAC Symposium on Nonlinear Control Systems*, 2001.

N.Petit, M.B.Milam, R.M.Murray. A new computational method for optimal control of a class of constrained systems governed by partial differential equations. *in Proc. of the 15th IFAC World Congress*, 2002.

M.Fliess, J.Levine, P.Martin, and P.Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61(6):1327–1360, 1995.

S.Boyd, L.Vandenberghe. Convex Optimization. *Cambridge University Press*, 2004.

C. de Boor. A Practical Guide to Splines. *Springer-Verlag*, 1978.