# Optimization of complex robot applications under real physical limitations

Matthieu Guilbert* Pierre-Brice Wieber † Luc Joly ‡

August 30, 2007

**Abstract**

This paper deals with minimum time trajectory optimization along a specified path subject to thermal constraints. We point out here that robots are often integrated in complex robotic cells, and the interactions between the robot and its environment are often difficult or even impossible to model. The structure of the optimization problem allows us to decompose the optimization in two levels, the first one being based on models and results of the theory of the calculus of variations, the second one being based on measurements and derivative free algorithms. This decomposition allows us to optimize the velocity profiles efficiently without knowing in advance the interactions between the robot and is environment. We propose here two numerical algorithms for these two levels of the decomposition which show good convergence properties. The resulting optimal velocity profiles are 5 to 10% faster than classical ones, and have been executed on successfully on a real Stäubli Rx90 manipulator robot.

**Keywords:** Robotics, Trajectory, numerical optimization, calculus of variations, thermal model, derivative free optimization, augmented Lagrangian.

## 1 Introduction

The programming of industrial robots is generally based on the operator's experience, regardless of the system's precise dynamics and relevant optimization criteria. And due to the complexity of robots and manufacturing systems, even highly qualified operators can only reach a limited level of efficiency. A better exploitation of the performances of robots integrated in manufacturing systems can only be achieved therefore by using computer aided optimization methods. Now, previous results on trajectory optimization usually focus on generating trajectories with minimum time or minimum energy criteria subject to the actuators' limitations without taking into account all the unmodeled interactions between the robot and its environment usually composed of several machine tools, pallets, etc... Moreover, these results usually consider limitations such as maximum velocities, accelerations or torques [Bes92] [LLO91] [Hol84] [BDG85] [LCL83] which don't reflect all the real limitations of a robot such as overheating, wearing and breaking. We propose here to derive an algorithm for optimizing velocity profiles in order to obtain a minimum cycle time while taking into account thermal constraints on one side, and all the unmodeled interactions between the robot and its environment, on the other side.

*Stäubli Robotics Faverges
†INRIA Rhône Alpes
‡Stäubli Robotics Faverges

We will first derive a temperature model in Section 2, then we will show in Section 3 that the special structure of the proposed optimization problem allows decomposing it in two levels. We will develop in Section 4 an optimal profile generator which corresponds to the first level using some calculus of variations, and we will develop in Section 5 derivative free optimization method for dealing with the unmodeled interactions between the robot and its environment which corresponds to the second level. We will finally test these algorithms in Section 6 with numerical simulations and experiments on a real industrial robot.

# 2 Temperature prediction for robotic systems

Minimizing the duration of robotic applications usually induces strong demands on the mechanical and electrical parts of the robots. Wearing and overheating are some of the classical consequences of these demands, and we will focus here on the increase of temperature. Since a high temperature can cause damages, this increase of temperature must be controlled and since the rise of temperature is a slow phenomenon (it can take more than 5 hours to stabilize), sensors can't be used to measure in advance the future stabilized temperature, reason why we need a thermal model to predict it. Since most robotic applications are cyclic, it is possible to derive a model which predicts the stabilized temperature corresponding to a given cycle once this cycle is known precisely enough. To predict this temperature, the references [Fan95], [Kob88] and [Mat89] propose to take into account the loss by Joule effect in the motors. Only the reference [Den01] takes into account both the loss of the motors and the loss in the mechanical parts of the actuators. Heat transfers between gears, motors, and other parts of the robot can be described by conduction, convection and radiation phenomena [TP98], but in practice, these three transfer modes are simultaneous and not easy to separate: their study is therefore often empirical.

A thermal model will be derived in Section 2.1 which only takes into account the conduction phenomenon (this is the major heat transfer in our system), then the validity of the model will be tested in Section 2.2.

## 2.1 Thermal model of the system

In order to predict the temperature of the robot, we will predict in fact the temperature at different points considered to be representative of the system from a thermal point of view. Moreover the articulations in an industrial robot are often enclosed in casings: there exist therefore strong thermal coupling between actuators. Specifically, we will identify our model on a Stäubli Rx90 in which the actuators are enclosed by pairs. Six heat sources can be distinguished then, 3 by actuator:

$$\begin{pmatrix} \Delta T_1 \\ \Delta T_2 \end{pmatrix} = A \begin{pmatrix} I_1 \\ I_2 \end{pmatrix} + B \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} + \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \tag{1}$$

with $\Delta T_j$ the elevation of temperature of the representative point $j$, $A$ and $B$ two constant matrices which represent the thermal resistances in the different materials, $\gamma_1$ and $\gamma_2$ two constant vectors representing the constant loss of the coils in the brakes, $I_1$ and $I_2$ representing the loss by Joule effect of the coils depending on the joint torque (since the current is globally proportional to the torques), $V_1$ and $V_2$ representing the loss due to friction in the gears depending on the joint velocity (since the motor velocity is globally proportional to the joint velocity), with:

$$I_j = \frac{1}{t_f} \int_0^{t_f} \Gamma_j^2(t) dt, \text{ and } V_j = \frac{1}{t_f} \int_0^{t_f} \dot{q}_j^2(t) dt$$

where $\Gamma_j(t)$ is the joint torque and $\dot{q}_j(t)$ is the joint velocity of the $j^{th}$ axis of the robot.

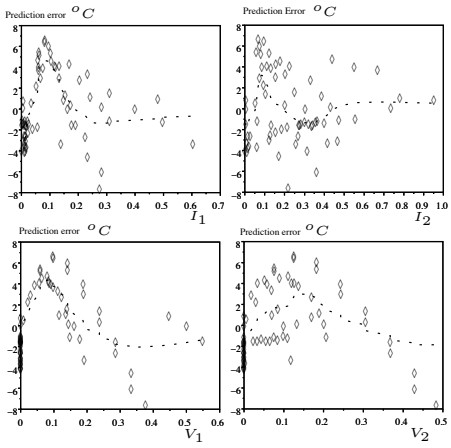


Figure 1: Prediction error according to $I_1$, $I_2$, $V_1$, $V_2$.

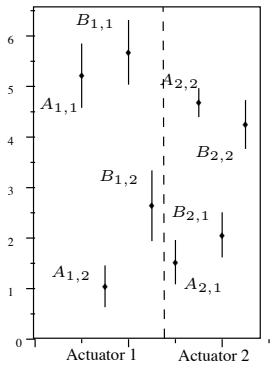


Figure 2: Confidence intervals of the elements of the matrix $A$ and $B$

The initial definition of this model has been based on physical considerations on Joule effects, friction, conduction, dissipation and other thermal effects, but a precise model of all these effects on a system as complex as a manipulator robot can be out of reach, and maybe not even useful for our purpose. This is why we restrict ourselves to the model (1) which can be considered to already reflect correctly the global behavior of the true system, as shown in the next section.

## 2.2 Identification and validation of the model

To identify the constants in this model for a given robot, 100 different trajectories have been executed on this robot with different current and velocity mean values ($I_j$ and $V_j$). For each trajectory, the stabilized temperature is measured after 6 hours of execution. The parameters can be identified then with a least squares procedure. The reliability of the model can be evaluated by studying the error prediction of this model and by calculating the confidence intervals of the identified parameters.

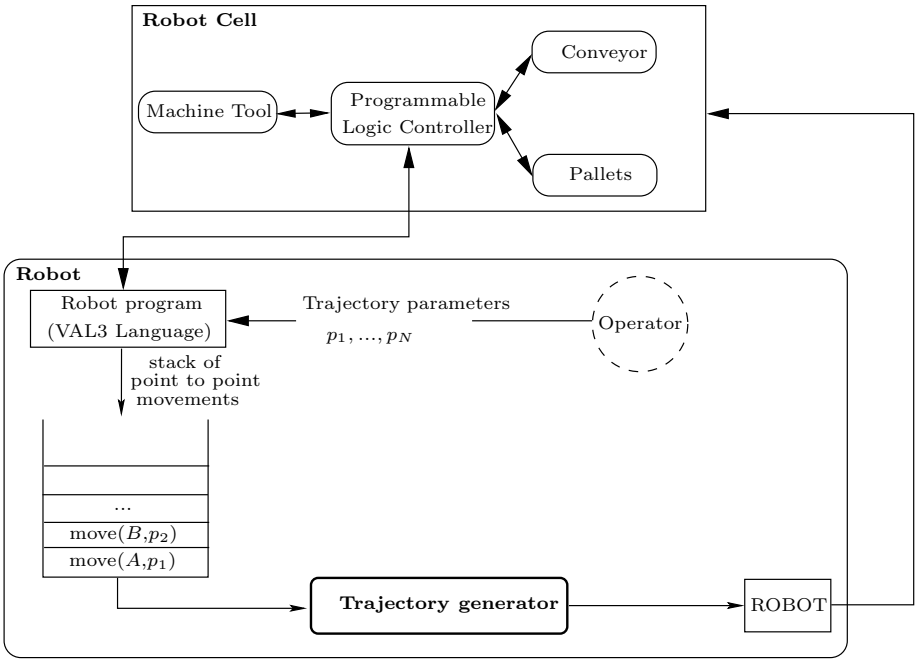


Figure 3: Classical structure of a complex robotic application

The general tendency of the prediction error can be seen in Figure 1 to show non linear behaviors for low currents and velocities. This obviously means that not all the physical phenomena have been modeled in equation (1) and that a non quadratic approach may be superior. However, despite the empirical design of this model, it gives predictions with a mean error of 5%, what can be considered as good enough here.

Confidence intervals [WW90] of the thermal resistances appearing in the matrices A and B have also been calculated. Statistically speaking, we are 95% sure that the parameters are in the intervals represented in Figure 2. Since the intervals don't cross the zero axis, all the identified parameters appear to have an influence on the predicted temperature, so all of them need to be present in equation (1).

# 3 Optimization of complex robotic applications

## 3.1 General structure of complex robotic applications

We are interested here in minimizing the cycle time of a complex robotic application without exceeding a maximum authorized temperature of the robot:

$$\min\ (t_{N+1} - t_1) \tag{2}$$

$$\frac{1}{t_{N+1} - t_1} \int_{t_1}^{t_N+1} (A\,\Gamma^2 + B\,\dot{q}^2)\,dt + \gamma + T_{amb} \leq T_{max}, \tag{3}$$

with $T_{amb}$ the ambient temperature, $T_{max}$ the maximum authorized temperature, $t_1$ the start time of the cycle and $t_{N+1}$ the end time of the cycle.

But in order to predict the temperature through the constraint (3) according to the thermal model (1), we are supposed to have a perfect knowledge of both the robot and the complete task it needs to realize. Unfortunately this task usually involves several machine tools, conveyors and pallets, all scheduled by a complex Programmable Logic Controller (PLC) which form together a complete robot cell that can be hard or even impossible to model. The interactions between the robot and the robot cell imply especially to be able to react to events which aren't always perfectly known in advance: small differences in the timings of the machine tools, small differences in the pick and place motions.

Reacting to such events not perfectly known in advance is usually done [BM03] by decomposing the trajectory in a series of point to point motions which are prepared only shortly ahead of time in order to allow quick reactions, and put in a stack as shown in Figure 3. Doing so, the trajectory generator needs therefore to work online with only a limited view of the task to be realized, what doesn't seem to be compatible at first sight with constraints such as the constraint (3) which needs to be taken into account over the whole cycle.

In order to be able to optimize somehow such complex and not perfectly known robotic applications, we will make the key assumption that they are "globally cyclic". More precisely, we will suppose that the differences between the cycles of these robotic tasks are small enough with respect to the optimal problem (2)-(3) so that some knowledge can be gathered cycle after cycle about the task, and used successfully in optimizing it.

## 3.2 Decomposability of the optimization problem

Let's make now an observation on the decomposability of the optimization problem (2)-(3) that is going to be of importance for solving it successfully in complex robotic applications such as the one described in Figure 3. If we decompose a cyclic movement of the robot in $N$ distinct motions $\mathcal{M}_i$ over time intervals $[t_i, t_{i+1}]$, we can consider $t_{i+1} - t_i = d(\mathcal{M}_i)$, $\int_{t_i}^{t_{i+1}} (A\Gamma^2 + B\dot{q}^2)dt = T(\mathcal{M}_i)$, so that the problem (2)-(3) can be written as:

$$\begin{cases} \min\limits_{(\mathcal{M}_1,\ldots,\mathcal{M}_N)} \sum\limits_{i=1}^{N} d(\mathcal{M}_i) \\ \sum\limits_{i=1}^{N} T(\mathcal{M}_i) - (T_{max} - \gamma - T_{amb})d(\mathcal{M}_i) \leq 0. \end{cases} \tag{4}$$

Following the resource decomposition method generally used for large scale problems [BGLS03], this optimization problem appears to be decomposable, what means that it is equivalent to:

$$\begin{cases} \min\limits_{(p_1,\ldots,p_N)} \sum\limits_{i=1}^{N} d_i^*(p_i) \\ \sum\limits_{i=1}^{N} p_i \leq 0, \end{cases} \tag{5}$$

with:

$$d_i^*(p_i) = \begin{cases} \min\limits_{\mathcal{M}_i} \ d(\mathcal{M}_i) \\ T(\mathcal{M}_i) - (T_{max} - \gamma - T_{amb})d(\mathcal{M}_i) \leq p_i. \end{cases} \tag{6}$$

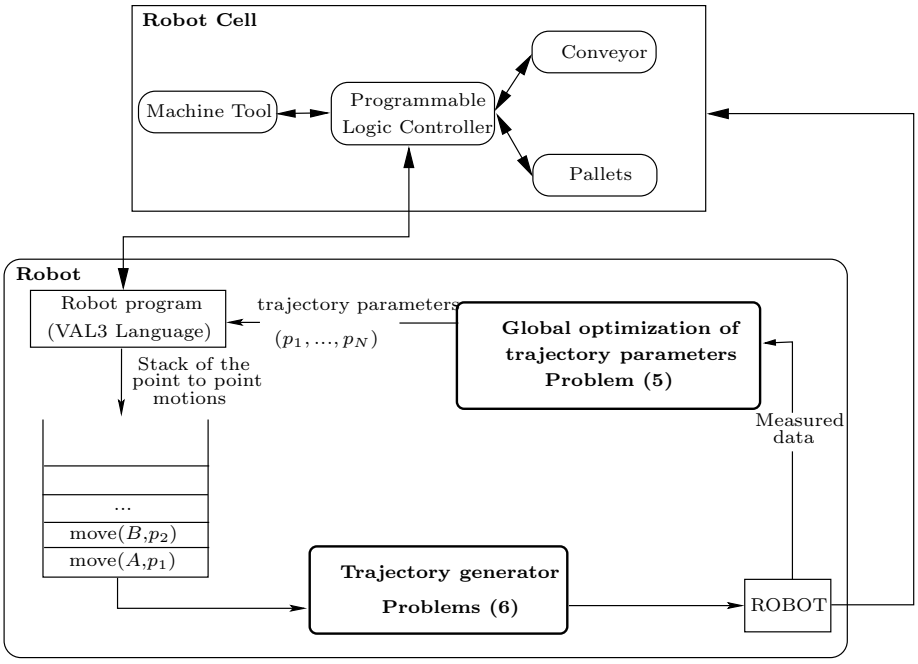


Figure 4: Two levels of optimization

Here, the optimization problems (6) correspond to optimizing each motion $\mathcal{M}_i$ independently from the others except for the resource $p_i$ which is allocated globally by the optimization problem (5). And these resources $p_i$ correspond to an energy increase allowed for each interval $[t_i, t_{i+1}]$.

## 3.3 Two levels of optimization

A nice property of the decomposition (5)-(6) of the optimization problem (2)-(3) is that it fits perfectly the structure of the robotic application shown in Figure 3. Indeed, we can observe in Figure 4 that the optimization problems (6) can be made to correspond to the trajectory generation with only a limited view of the task, while the global knowledge of this task is dealt with by the optimization problem (5). The key point in doing so is that different optimization methodologies can be used on these two distinct levels of optimization: the optimization problems (6) work on point to point motions which are sufficiently known in advance so that efficient model based optimization methods can be used, as will be shown in Section 4, whereas the interaction with the imperfectly known robot cell can be treated entirely at the level of problem (5), where specific methods for dealing with imprecise problems, such as derivative free optimization can be used, as will be shown in Section 5.

This way, dynamic models of the robot can be used to evaluate the duration $d(.)$ and the temperature increase $T(.)$ corresponding to a movement $\mathcal{M}_i$ when solving the optimization problems (6). But in order to take into account the unmodeled parts of the whole robotic cell when solving the resource allocation problem (5), these quantities will need to be evaluated with the help of measures gathered directly on the robot. The cycle time can be measured directly, but not the stabilized temperature, as discussed earlier in Section 2, so this temperature will still need

| | |
|---|---|
| **(i)** | Initialize the parameters $(p_1, ..., p_N)$ |
| **(ii)** | Execute a cycle of the robotic application |
| **(ii-a)** | Solve the problems (6) to generate each point to point motion |
| **(ii-b)** | Execute these motions and measure $t_c$, $\dot{q}$ and $\Gamma$ |
| **(iii)** | Find a new set of parameters $(p_1, ..., p_N)$ through the optimization problem (7) |
| **(iv)** | go to step (ii) until an optimal set of parameters is found |

Table 1: General guidelines for the two levels of optimization

to be estimated with the help of the model (1), but based on real measures of the torques and speeds of the actuators of the robot. Doing so, the problem (5) turns into:

$$\begin{cases} \min\limits_{(p_1,...,p_N)} t_c \\ \int_0^{t_c} (A\Gamma^2 + B\dot{q}^2)\, dt - (T_{max} - \gamma - T_{amb})\, t_c \leq 0, \end{cases} \tag{7}$$

where $t_c$, $\Gamma$ and $\dot{q}$ are the cycle time, the torque and the velocity, all measured directly on the robot. Such an optimization of complex robotic applications with measured data would follow therefore the guidelines of Table 1.

# 4 An optimal profile generator

The description of the general optimization algorithm in Section 3 has shown that the trajectory generator needs to solve a succession of local problems (6) which deal with point to point motions. Since we aren't interested here in optimizing the geometric path of the trajectory (for industrial reasons such as security), we will focus in this section on the optimization of point to point motions along a specified geometric path. We will propose first of all the calculation of an analytical solution in a simple case in Section 4.1, then a generic spline based algorithm in Section 4.2 to compute a numerical approximation of the solution in the general case. In this section, we will only consider a point to point motion beginning at time $t = 0$ and finishing at time $t = t_f$ without loss of generality.

## 4.1 Analytical solution in a simple case

Minimum time control problems in robotics are classically solved with the help of BANG-BANG or BANG-zero-BANG solutions, when bounds are expressed on the control variables: they present jumps of the control variables from one bound to another, what explains their name [BH75]. In our case, the bounds are not directly expressed on the control variables but on the temperature, therefore such classical profiles aren't correct answers to our problem. To find an analytical solution when the bounds are expressed on the temperature, we consider here a simple movement of a horizontal axis of our robot. By consequence, the system's dynamics presents no gravity effects, a constant inertia, no centrifugal and Coriolis forces, what leads to the simple dynamic model:

$$\Gamma = J\ddot{q} + F_v\dot{q} + F_s \tag{8}$$

where $\Gamma$ is the joint torque, $\ddot{q}$ the acceleration, $\dot{q}$ the velocity, $J$ the inertia of the whole system and $F_v$ and $F_s$ the viscous and Coulomb friction (a constant here since we will consider a trajectory where the sign of the velocity doesn't change). In terms of function to minimize and constraints, the problem we need to solve here is:

$$\min\ t_f = \int_0^{t_f} 1\, dt \tag{9}$$

subject to:

$$\frac{1}{t_f}\int_0^{t_f} a(J\ddot{q} + F_v\dot{q} + F_s)^2 + b\,\dot{q}^2\, dt \quad = \quad T_{max} - c \tag{10}$$

$$\int_0^{t_f} \dot{q}\, dt \quad = \quad q_f - q_0 \tag{11}$$

with $q_0$ and $q_f$ the initial and final position of the axis, $a$, $b$, $c$ the constants of the thermal model (always strictly positive) and $T_{max}$ the maximal authorized temperature. Note that the constraint (10) is an equality instead of an inequality since we consider that it will be an active constraint for this trajectory. We face therefore a minimization problem subject to isoperimetric constraints in which the end point isn't fixed [Pin93]. Lagrange multipliers $\lambda_1$ and $\lambda_2$ need to be introduced and we want to find the saddle points of:

$$\int_0^{t_f} \left(1 + \frac{\lambda_1}{t_f}(a(J\ddot{q} + F_v\dot{q} + F_s)^2 + b\dot{q}^2) + \lambda_2\dot{q}\right) dt = \int_0^{t_f} F(t,\dot{q},\ddot{q})dt \tag{12}$$

The necessary condition in this case is the Euler-Lagrange differential equation

$$\frac{\partial F(t,\dot{q},\ddot{q})}{\partial \dot{q}} - \frac{d}{dt}\left(\frac{\partial F(t,\dot{q},\ddot{q})}{\partial \ddot{q}}\right) = 0. \tag{13}$$

Since

$$\frac{\partial F}{\partial \dot{q}} = 2\lambda_1\frac{aF_v^2 + b}{t_f}\dot{q} + \frac{2\lambda_1 aJF_v}{t_f}\ddot{q} + \frac{2a\lambda_1 F_s F_v}{t_f} + \lambda_2, \tag{14}$$

$$\frac{\partial F}{\partial \ddot{q}} = \frac{2\lambda_1 aJ^2}{t_f}\ddot{q} + \frac{2\lambda_1 aJF_v}{t_f}\dot{q} + \frac{2a\lambda_1 F_s J}{t_f}, \tag{15}$$

and

$$\frac{d}{dt}\left(\frac{\partial F}{\partial \ddot{q}}\right) = \frac{2\lambda_1 aJ^2}{t_f}\dddot{q} + \frac{2\lambda_1 aJF_v}{t_f}\ddot{q}, \tag{16}$$

this Euler-Lagrange equation becomes

$$\frac{-2\lambda_1 aJ^2}{t_f}\dddot{q} + \frac{2\lambda_1(aF_v^2 + b)}{t_f}\dot{q} + \frac{2a\lambda_1 F_s F_v}{t_f} + \lambda_2 = 0, \tag{17}$$

or written differently,

$$\dddot{q} - r\dot{q} = C\left(\frac{\lambda_2}{\lambda_1}\right) \tag{18}$$

with $r = \dfrac{aF_v^2 + b}{aJ^2}$ and $C\left(\dfrac{\lambda_2}{\lambda_1}\right) = \dfrac{F_s F_v}{J^2} + \dfrac{\lambda_2}{2\lambda_1 aJ^2}$. Note that if $\lambda_1 = 0$, the Euler-Lagrange equation (17) gives $\lambda_2 = 0$ and the problem (12) degenerates: we can fairly consider therefore that $\lambda_1$ is different from zero. Integrating this differential equation leads then to solutions of the form:

$$\dot{q}(t) = \zeta\, \sinh(\sqrt{r}\, t) + \nu\, \cosh(\sqrt{r}\, t) - \frac{1}{r}C\left(\frac{\lambda_2}{\lambda_1}\right). \tag{19}$$

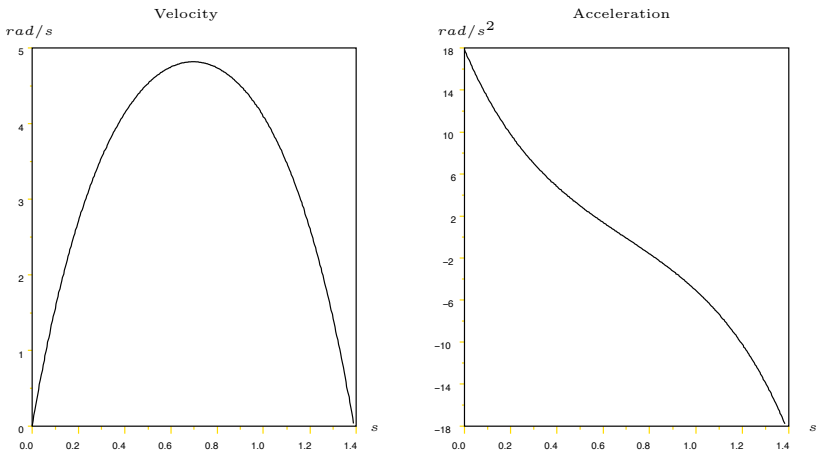


Figure 5: Optimal velocity and acceleration profiles

Using the equality constraint (11) together with boundary conditions such as:

$$\begin{cases} \dot{q}(0) = 0, \\ \dot{q}(t_f) = 0, \end{cases} \tag{20}$$

we can determine the constants $\zeta$, $\nu$ and the ratio $\frac{\lambda_2}{\lambda_1}$ (or directly the constant $C$) as functions of the final time $t_f$:

$$\begin{cases} \zeta(t_f) = (q_f - q_i)\dfrac{\sqrt{r}(\cosh(\sqrt{r}\,t_f) - 1)}{-2\cosh(\sqrt{r}\,t_f) + 2 + t_f\sqrt{r}\sinh(\sqrt{r}\,t_f)}, \\ \nu(t_f) = -(q_f - q_i)\dfrac{\sqrt{r}\sinh(\sqrt{r}\,t_f)}{-2\cosh(\sqrt{r}\,t_f) + 2 + t_f\sqrt{r}\sinh(\sqrt{r}\,t_f)}, \\ C(t_f) = r\nu, \end{cases} \tag{21}$$

while this final time $t_f$ can be computed by solving numerically the temperature constraint (10).

The Figure 5 shows such an optimal velocity profile on a Stäubli Rx90 for a movement of its first axis from $-2.26$ rad to $+2.26$ rad, for a maximal temperature of $100^oC$. After solving the equation (10), we find $t_f = 1.38s$. Since the jerk appears in the necessary condition (18), the acceleration is continuous and differentiable everywhere, what helps to avoid vibrations. Note that the velocity on the boundaries of the trajectory can be fixed at will through the boundary conditions (20), but the acceleration on these boundaries is unfortunately imposed by the shape of the solution.

## 4.2 Numerical approximation of the solution in the general case

In the general case, the simple dynamics (8) of the previous section turns into:

$$\Gamma = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}), \tag{22}$$

with $M(q)$ the inertia matrix, $C(q,\dot{q})$ the matrix of centrifugal and Coriolis effects, $G(q)$ the gravity effects and $F(\dot{q})$ the friction [KD99].

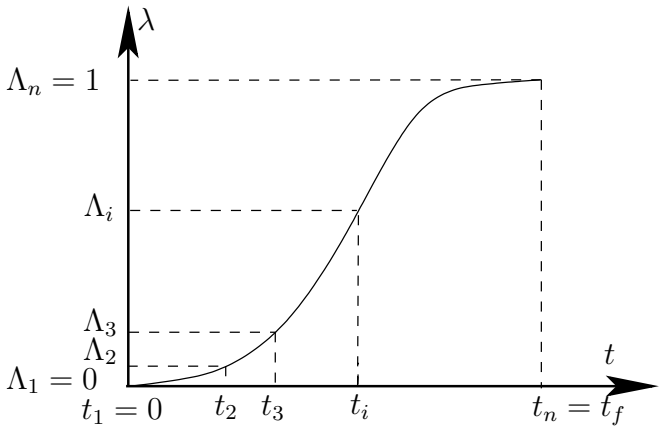


Figure 6: Discrete time law

We are only interested here in optimizing a velocity profile along a specified geometric path, so we can introduce the curvilinear abscissa $\lambda : [0, t_f] \rightarrow [0, 1]$ (the time law) and the geometric path $Q : [0, 1] \rightarrow \mathbb{R}^6$, both functions of class $C^2$ such that:

$$q(.) = Q(\lambda(.)) \tag{23}$$

$$\dot{q}(.) = \frac{dQ}{d\lambda}(\lambda(.))\dot{\lambda}(.) \tag{24}$$

$$\ddot{q}(.) = \frac{d^2Q}{d\lambda^2}(\lambda(.))\dot{\lambda}^2(.) + \frac{dQ}{d\lambda}(\lambda(.))\ddot{\lambda}(.) \tag{25}$$

Introducing these notations within the dynamics (22) allows reformulating it slightly more simply [BDG85], [Hol84] [Žla96]:

$$\Gamma = m(\lambda)\ddot{\lambda} + c(\lambda)\dot{\lambda}^2 + g(\lambda) + f(\dot{\lambda}) \tag{26}$$

where $m$, $c$, $g$ and $f$ are vectors which represent the inertia, centrifugal and Coriolis, gravity and friction effects. Still, this dynamics is much more complex than (8) and an analytical solution to the corresponding optimization problem will probably be out of reach. We need therefore to look for a numerical approximation of this solution. There exist various classical techniques for finding such an approximation, and we propose here to briefly describe the one proposed in [LLO91] which appears to be well suited to our problem.

Since the time law $\lambda(t)$ must be at least of class $C^2$, we will use cubic splines defined as shown in Figure 6, with $\lambda(t_i) = \Lambda_i$ for $1 \leq i \leq n$, $t_1 = 0$ and $\Lambda_1 = 0$, $t_n = t_f$ and $\Lambda_n = 1$. Since $t_n = t_f$ is variable whereas $\Lambda_n = 1$ is fixed, we will consider that all the $\{t_i\}_{(1 \leq i \leq n)}$ are variable whereas all the $\{\Lambda_i\}_{(1 \leq i \leq n)}$ are fixed, leading to a non-uniform spline. Following [LLO91], our strategy to define this cubic spline is to impose the continuity of the velocity and the acceleration at the nodes $t_i$, and to fix the velocity on the boundaries. We use the intermediate variables $\{\ddot{\Lambda}_i\}_{(1 \leq i \leq n)}$ to fix the acceleration at each knot. Each of the cubic polynomials $\lambda_i(t) = \lambda(t)$ for $t \in [t_i, t_{i+1}]$ constituting the spline can be written in terms of the $\ddot{\Lambda}_i$ and the $h_i$:

$$\lambda_i(t) = \frac{(t_{i+1} - t)^3}{6h_i}\ddot{\Lambda}_i + \frac{(t - t_i)^3}{6h_i}\ddot{\Lambda}_{i+1} + \left(\frac{\Lambda_{i+1}}{h_i} - \frac{h_i\ddot{\Lambda}_{i+1}}{6}\right)(t - t_i) + \left(\frac{\Lambda_i}{h_i} - \frac{h_i\ddot{\Lambda}_i}{6}\right)(t_{i+1} - t). \tag{27}$$

The continuity of the velocity is satisfied then by solving a linear system that leads to the computation of the $\{\ddot{\Lambda}_i\}_{(1\leq i\leq n)}$:

$$C(h)\ddot{\Lambda} = d(h) \tag{28}$$

with

$$C(h) = \begin{pmatrix} 2h_1 & h_1 & & & & 0 \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & & & & h_{n-1} & 2h_{n-1} \end{pmatrix},$$

$$d(h) = \begin{pmatrix} 6\left(\frac{\Lambda_2-\Lambda_1}{h_1} - v_1\right) \\ 6\left(\frac{\Lambda_3-\Lambda_2}{h_2} - \frac{\Lambda_2-\Lambda_1}{h_1}\right) \\ \vdots \\ 6\left(\frac{\Lambda_n-\Lambda_{n-1}}{h_{n-1}} - \frac{\Lambda_{n-1}-\Lambda_{n-2}}{h_{n-2}}\right) \\ 6\left(v_n - \frac{\Lambda_n-\Lambda_{n-1}}{h_n}\right) \end{pmatrix},$$

with $h_i = t_{i+1} - t_i$ the time intervals between knots,and $h = (h_1, h_2, ..., h_{n-1})^T$ the new set of parameters for the optimization procedure. After solving the linear system (28), the spline is totally determined by the $\{\Lambda_i\}_{1\leq i\leq n}$, the velocity on the boundaries $v_1$ and $v_n$ and the time intervals $\{h_i\}_{1\leq i\leq n-1}$. Note that these $\Lambda_i$ need not be uniformly distributed, and that a non-uniform distribution might even lead to a better numerical approximation. Note also that the matrix $C(h)$ is non singular here since it is diagonally dominant, and that there are efficient numerical methods to invert such tri-diagonal matrices.

With this parametrization, the cost function appears to be a simple linear function $t_f = \sum_{i=1}^{n-1} h_i$. It is interesting then to estimate the thermal constraint (3) with a trapezoidal approximation, precise enough in our case as we will see in the Section 6:

$$\frac{1}{2}\sum_{i=0}^{n-1}\left(A(\Gamma^2(t_i)+\Gamma^2(t_{i+1}))h_i + B\left(\dot{q}^2(t_i)+\dot{q}^2(t_{i+1})\right)h_i\right) + t_f(\gamma + T_{amb} - T_{max}) \leq 0. \tag{29}$$

Indeed, with this approximation, the dynamics (26) is always evaluated at constant predefined positions $\{\Lambda_i\}_{1\leq i\leq n}$, so that the vectors $m(\lambda)$, $c(\lambda)$, $g(\lambda)$ in (26) are constant throughout the optimization process, allowing a straightforward computation of the derivatives of this dynamics that are required by the optimization algorithms.

The last important point here is the initialization of the optimization process: to help its convergence, we must choose a first iterate as close as possible to the optimal solution and satisfying all the constraints. From a practical point of view, we generate a BANG-zero-BANG profile, and we use a dichotomy technique to improve the first iterate by testing the constraints: the duration of the movement is stretched if any constraint is violated, compressed otherwise. This whole procedure will be tested and validated in Section 6.

# 5 Global optimization of robot applications with hardware in the loop

The description of the general optimization algorithm in Section 3 has shown the need to solve the global problem (7) with a cost function and inequality constraints which need to be evaluated

from data directly measured on the robot, what appeared to be the only way to take into account the unmodeled part of the whole robotic cell.

A similar scheme can be found in Iterative Learning Control methods, when a robot repeatedly attempts to execute a prescribed task while an adaptation algorithm successively improves the control system's performance from one trial to the next by updating the control input based on the error signals from previous trials [Lon00] [Hor93]. But such methods can't be easily applied to problems with global criteria and constraints, such as the cycle time and the temperature constraints that we need to deal with here. More than that, the tasks that we consider here aren't exactly cyclic, what is generally a strict requirement for applying such methods successfully.

The difficulty in such algorithms is to deal with noisy data, with gradients of the criterion and constraints that don't exist or can't be obtained easily and efficiently: we must use therefore optimization methods without derivatives.

## 5.1 Unconstrained optimization without derivatives

Concerning optimization methods without derivatives, direct search methods as discussed in [Pow98] and [CST97] are to be looked for. The Nelder-Mead simplex method is one of the most frequently used algorithm in optimization without derivatives, but it doesn't converge in some cases and suffer from inefficiency when the dimension of the problem is too large. Other methods such as simulated annealing or genetic algorithms suffer from similar limitations [Pow98].

A real improvement in direct search methods has been obtained when Powell described a method for solving non-linear unconstrained minimization problems based on the use of conjugate directions [Pow64]: at most $n$ successive linear searches along mutually conjugate directions are necessary for finding the minimum of a positive definite quadratic form in $\mathbb{R}^n$. He proposed then (independently of [Win73]) to use the available values of the objective function for building a quadratic model of it. This model is assumed to be valid in a neighborhood of the current iterate, which is described as a trust region, whose radius is iteratively adjusted. The model is then minimized within this trust region, hopefully yielding a point with a lower value of the objective function.

There exist two efficient algorithms available today which implement this idea: Derivative Free Optimization (DFO) [CST97] from Conn, Scheinberg and Toint and NEWUOA from Powell [Pow04]. The main difference lies in the way the underlying quadratic model is updated every time a new value of the objective function is obtained. By a clever minimization of the Frobenius norm between the updates of the matrix corresponding to the quadratic term, the NEWUOA algorithm allows strongly reducing the total number of function evaluations required for finding the optimum. This is a very important detail since the evaluation of the cost function and of the constraints requires executing a whole application cycle with the robot, as explained in the algorithm of Table 1, what can be extremely costly and time consuming. NEWUOA appears therefore as the best choice today for minimizing our cost function without derivatives, but it doesn't deal with constraints, and our problem is subject to constraints.

## 5.2 Penalty methods in non-linear programming

The most classical way of taking care of constraints when working with an optimization algorithm not explicitly meant for this is to penalize the objective function when these constraints are violated. This amounts to not minimizing only the original objective function $t_c(p)$, but a combination such as

$$\Phi(p, \sigma) = t_c(p) + \sum_i \sigma_i e^{c_i(p)} \tag{30}$$

| | |
|---|---|
| **(i)** | Choose a fixed sequence $\{\sigma^{(k)}\}$, for example $\{1, 10, 10^2, 10^3, ...\}$ |
| **(ii)** | For each $\sigma^k$, find a local minimizer $p(\sigma^k)$ to $\min_p \Phi(p, \sigma^k)$ |
| **(iii)** | Terminate when $\max(c_i(p), 0)$ is sufficiently small |

Table 2: Iterative scheme for inexact penalty functions

| | |
|---|---|
| | Considering $\lambda_i = \theta_i \sigma_i$, |
| **(i)** | $\lambda \leftarrow \lambda^{(1)}$, $\sigma \leftarrow \sigma^{(1)}$, $k \leftarrow 0$, $\|\nabla \Psi^{(0)}\|_\infty \leftarrow \infty$ |
| **(ii)** | Find the minimizer $\mathbf{p}(\lambda, \sigma)$ of $\Phi(\mathbf{p}, \lambda, \sigma)$ and denote $\mathbf{c} = \mathbf{c}(\mathbf{p}(\lambda, \sigma))$ |
| **(iii)** | If $\|[-\max(c_i, \frac{\lambda_i}{\sigma_i})]_{i=1..m}\|_\infty > \frac{1}{4}\|\nabla \Psi^{(k)}\|_\infty$ then : $\forall i$, if $\|c_i\| > \frac{1}{4}\|c^{(k)}\|_\infty$ then $\sigma_i \leftarrow 10\sigma_i$ go to step (ii) |
| **(iv)** | $k \leftarrow k + 1$, $\lambda^{(\mathbf{k})} \leftarrow \lambda$, $\sigma^{(\mathbf{k})} \leftarrow \sigma$, $\mathbf{c}^{(\mathbf{k})} \leftarrow \mathbf{c}$ |
| **(v)** | $\forall i = 1..m, \lambda_i \leftarrow \lambda_i^{(k)} - \max(\sigma_i c_i^{(k)}, \lambda_i^k)$ and $\|\nabla \Psi^{(k)}\|_\infty \leftarrow \|[-\max(c_i, \frac{\lambda_i}{\sigma_i})]_{i=1..m}\|_\infty$ |

Table 3: Numerical scheme using the augmented Lagragian method

where the functions $c_i(p)$ correspond to the temperature constraints in the problem (7). It can be proved that with an iterative scheme such as in Table 2, the minima obtained in step (ii) converge to the minimum of the constrained problem when $\sigma \to \infty$ [Fle87]. But these penalized problems obviously become ill-conditioned when $\sigma$ increases and their minima never correspond exactly to the minimum of the constrained problem.

Another option then is the augmented Lagrangian method [Fle87], which can still be seen as a penalty method, where minima of the function

$$\Phi(p, \theta, \sigma) = t_c(p) + \frac{1}{2} \sum_i \sigma_i (\max(c_i(p) - \theta_i, 0))^2. \tag{31}$$

are searched in a slightly more complex iterative scheme, shown in Table 3. This iterative scheme can be shown to find the exact solution to the original constrained problem with finite values of the parameters $\sigma_i$ and $\theta_i$, avoiding any ill-conditioning [Fle87] and answering therefore to the main problems raised by the previous approach.

A third option could be to consider a penalization such as

$$\Phi(p, \sigma) = t_c(p) + \sigma \sum_i \max(c_i(p), 0) \tag{32}$$

the minimum of which is exactly the same as the one of the constrained problem for a large enough but finite value of $\sigma$ [BGLS03] [Fle87]. But this penalty function is not differentiable at the minimum, what algorithms such as NEWUOA are not able to deal with properly, leading to serious convergence problems. This third approach must be discarded then, and only the exponential penalty function (30) and the augmented Lagrangian (31) will be considered in the next section.


Figure 7: Flat configuration of a Stäubli Rx90

# 6 Numerical and experimental validation

It is necessary now to validate the three algorithms developed in the previous sections: the optimal profile generator of Section 4, the global optimizer of Section 5 and the complete algorithm described in Figure 4 of Section 3. We first validate the optimal profile generator and the global optimizer separately in Sections 6.2 and 6.3, all these experiences lead to a comparison between all the optimized profiles in Section 6.4. All the tests will be based on three robot applications which will be described first of all in Section 6.1.

## 6.1 Description of the robot tasks to optimize

In order to verify the convergence of the optimal profile generator of Section 4, we will apply it to a simple case, a movement of the first axis of a Stäubli Rx90 in flat configuration (Figure 7) from $-2.26$ rad to $+2.26$ rad with a pause of $0.5\,s$ at the end position. The analytical solution to this simple robot task has already been described in Figure 5: we will be able therefore to verify the precision of the numerical scheme by direct comparison with this analytical solution. The robustness of the convergence of this optimal profile generator will be tested then by applying it to a real industrial application, the pick and place application shown in Figure 8.

In order to test the global optimizer of Section 5 independently from the optimal profile generator of Section 4, we will apply it first with a BANG-zero-BANG profile generator. In that case, the trajectory parameters tuned by this global optimizer (Figure 4) won't be the temperature resources $p_i$ introduced in the decomposition (5)-(6) of the original optimization problem (4), but the classical maximum acceleration, velocity and deceleration of the BANG-zero-BANG profile generator for each motion $\mathcal{M}_i$. This will be applied to the pick and place application of Figure 8 and a load/unload application described in Figure 9. This load/unload application implies 4.5 times more parameters to optimize than the pick and place application, what will allow testing the applicability of the global optimizer on large real-life problems.

Applying independently the optimizer of Section 5 and the optimal profile generator of Section 4 on the same pick and place application of Figure 8 will allow us to compare optimized BANG-zero-BANG profiles to truly optimal profiles, allowing to quantify the contribution of the latter with respect to the former.

For testing the complete algorithm of Figure 4 with two levels of optimization, only simulations have been realized since we didn't have access to a real robot at that moment for experiments. It is tested first of all on the same simple application as before, the movement of the first axis from $-2.26\,rad$ to $2.26\,rad$ with a pause of $0.5\,s$, but under real working conditions, that is with a total cycle time and the corresponding motor torques and speeds not known in advance. The

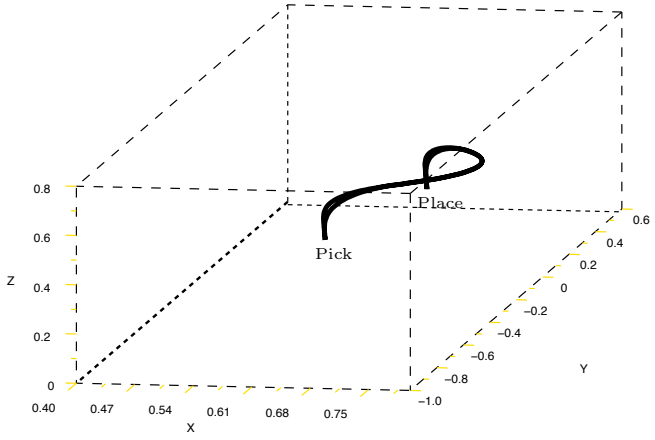


Figure 8: Geometric trajectory of a manipulator robot during a pick and place application

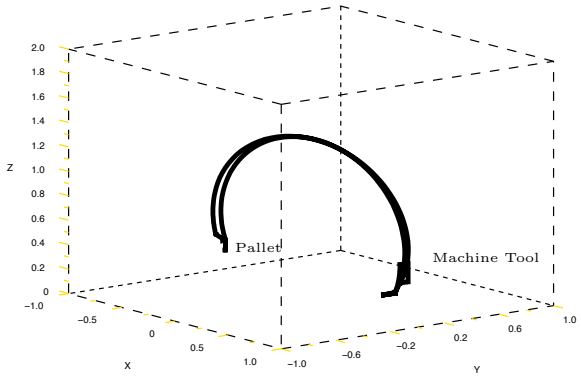


Figure 9: Geometric trajectory of a manipulator robot during a load/unload application

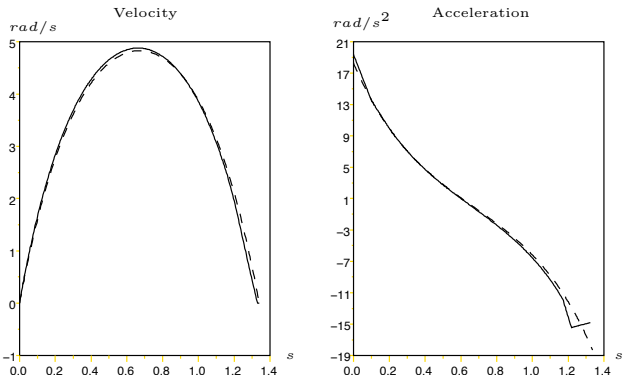


Figure 10: Analytical (dashed curve) and numerical (plain curve) optimal profiles

pause of 0.5 $s$ and its impact on the thermal constraint isn't modeled here but discovered through measures realized on a simulated robotic cell.

The pick and place application of Figure 8 is considered then, split into 9 point to point motions in order to validate the resource decomposition method introduced in Section 3.

## 6.2 Optimal profile generator

When applied to the simple task of Figure 5, the numerical algorithm described in Section 4.2 converges to the solution showed in Figure 10, with an optimal time $t_f = 1.35\,s$ (using the Feasible Sequential Quadratic Programming (FSQP) algorithm [LZT97] to solve the underlying non-linear optimization problem). We can observe that it is very close to the analytical solution found in Section 4.1. The very small difference between these two solutions can be identified to be solely due to the discretization of the time law. For the same reason, the approximate computation in (29) of the constraint (3) appears to slightly underestimate the limiting temperature constraint in this specific case, allowing a faster solution here than the truly optimal solution described in Section 4.1, with only 1.35 $s$ of cycle time instead of 1.38 $s$ in Section 4.1.

More generally, this algorithm has been observed to converge properly as soon as the constraints are satisfied from the beginning of the optimization process, as soon as the first iterate is a feasible point. This condition appeared to be of great importance to obtain this convergence: the initialization described at the end of Section 4.2 appears therefore to be a key point for the robustness of the whole numerical algorithm.

This trajectory has been executed then on a real Stäubli Rx90 robot without filtering, in spite of the discontinuities of the acceleration at the boundaries. The stabilized temperature measured after 6 hours reaches the prescribed limit with only 3% of error. Both the dynamic and the temperature models appear therefore to allow very precise predictions. Note however that oscillations appear in Figure 11 which have not been predicted. These oscillations are due to the discontinuities of the acceleration on the boundaries that excite the vibration modes of the robot: it appears that fixing the acceleration at the boundaries can solve very simply this problem.

## 6.3 Global optimization with BANG-zero-BANG profiles

Then we test the global optimizer of Figure 4 with a BANG-zero-BANG profile generator. Three different experiments are realized in order to:

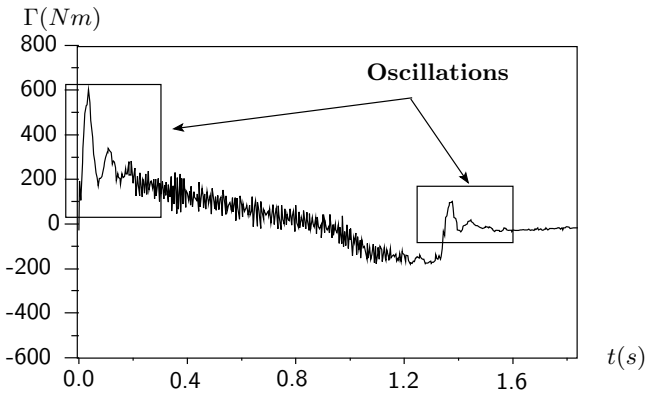


Figure 11: Measured torque

| Exponential penalty function | Augmented Lagrangian penalty function |
|---|---|
| $\sigma^1 = [50, ..., 50]^T$ | $\sigma^{(1)} = [100, ..., 100]$<br>$\lambda^{(1)} = [0, ..., 0]$ |

Table 4: Initial values of the penalty coefficients.

- compare the exponential penalty function (30) with the augmented Lagrangian (31),
- test the robustness to task changes,
- test the influence of a large number of trajectory parameters,

The comparison of the exponential penalty function (30) with the augmented Lagrangian (31) is realized on the optimization of the pick and place application of Figure 8, with weighting coefficients initialized as in Table 4.

Figure 12 shows an identical evolution in both cases in the beginning (in the boxed area): this is due to the initialization of the quadratic approximation of the cost function which is always realized in the same systematic way by the NEWUOA algorithm. We can observe then in both cases a decrease of the cycle time while the constraints rise up to their limit, with convergence in less than half an hour. We can observe that the use of an exponential penalty function implies a milder management of the temperature constraints, but both methods lead to an equivalent cycle time: it seems therefore difficult to make a clear choice between them. Note also that the constraints can be violated temporarily during the convergence process of both methods, as can be seen in Figures 12 and 13: this shouldn't be problematic as long as the only constraint being considered is a stabilized temperature, but this can be an important detail in other cases.

The experiment for testing the robustness of this optimization method to task changes consists in executing the same pick and place application as before but carrying a load of 6 Kg. The torques and the velocities of the actuators and therefore the stabilized temperature and the cycle time are altered: without giving any specific information to the algorithm, a convergence to a different slower solution can be observed in Figure 13. The algorithm automatically takes into account the changes in the robot task thanks to the use of data directly recorded by sensors on the robot in order to find an appropriate solution.

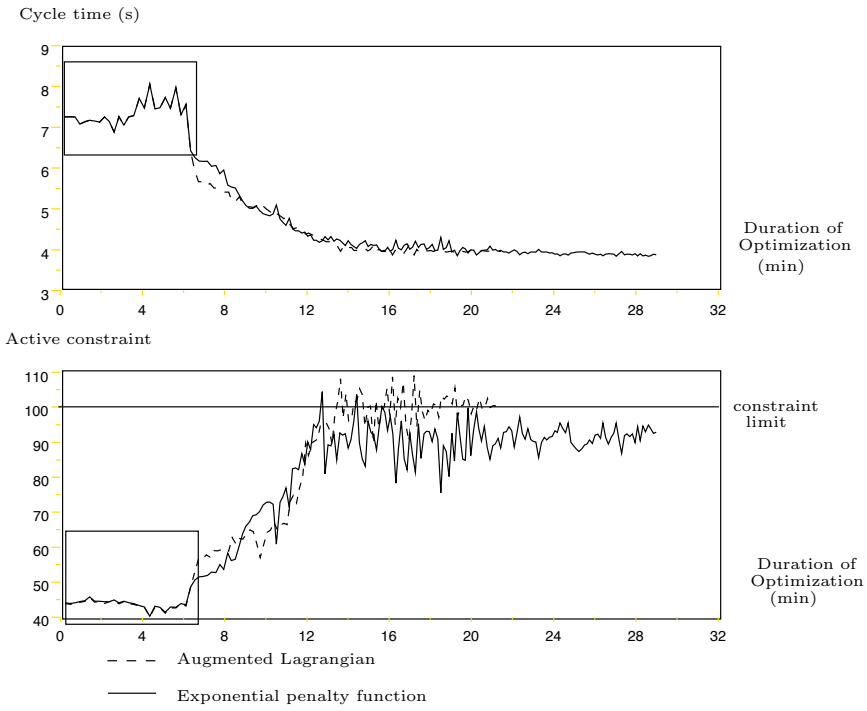


Figure 12: Convergence of the algorithm for the pick and place application without load.

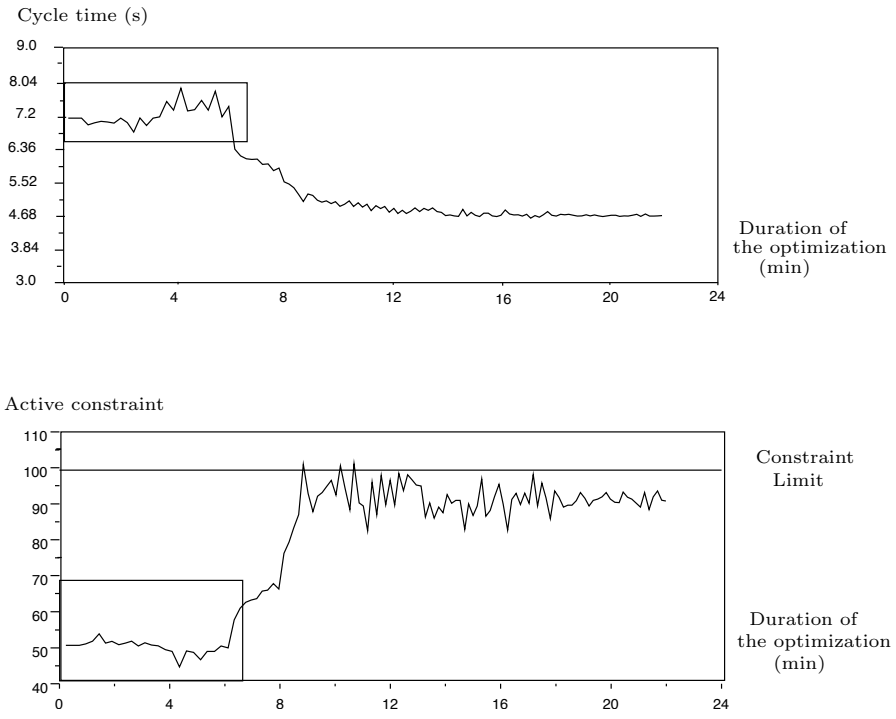


Figure 13: Convergence of the algorithm for the pick and place application with a load of 6Kg, using the exponential penalty function.

| Application | Number of trajectory parameters | Number of cycles | Duration of the optimization |
|---|---|---|---|
| Pick & Place | 12 | 600 | 30 min |
| Pick & Place with load | 12 | 400 | 30 min |
| Load/unload | 54 | 1800 | 5 h |

Table 5: Optimization time before reaching convergence.

| Application | Nominal cycle time | After Optimization | Gain |
|---|---|---|---|
| Pick & Place | 6.52s | 3.89s | **40%** |
| Pick & Place with load | 6.52s | 4.66s | **28%** |
| Load/unload | 8.12s | 7.44s | **8.4%** |

Table 6: Gain of cycle time using the global optimizer.

The experiment for testing the influence of a large number of parameters consists in the load/unload application of Figure 9 with 54 parameters instead of only 12 for the pick and place application of Figure 8. A minimum is reached after 5 hours of optimization instead of 30 minutes earlier, as shown in Table 5. Table 6 shows that the improvement of the performance of the robot with respect to the nominal constructor settings is less than for the previous application, but still of 8%. Note that 5 hours for optimizing a robotic application isn't long when this application is going to be executed repeatedly 8% faster for months or years. The proposed algorithm appears therefore to be well adapted to the optimization of a complex industrial applications.

## 6.4 Comparison between the different optimized profiles

We can quantify then the increase of performance when using the truly optimal velocity profiles with respect to optimized BANG-zero-BANG profiles on the simple task of Figure 5 and the pick and place application of Figure 8. Table 7 shows that the optimized BANG-zero-BANG profiles are already 40 to 50% faster than the nominal profiles suggested by the constructor, but the truly optimal velocity profiles are still 3 to 6% faster, what appears still as a significant increase in productivity on an industrial set-up.

Table 7 also shows that the profiles optimized by the complete algorithm in real industrial

| Application | Nominal | optimized BANG-zero-BANG profile | optimal velocity profile | Complete algorithm |
|---|---|---|---|---|
| Simple task | 7.91s | 3.76s | 3.68s | 3.70 |
| Pick & Place | 6.52s | 3.89s | 3.72s | 3.74 |

Table 7: Comparison of the cycle time resulting from different methods of optimization and the nominal profile.

conditions are only 0.5% slower than the truly optimal velocity profiles and 1.5 to 4% faster than the optimized BANG-zero-BANG profiles: it allows therefore a significant increase of productivity with respect to classical trajectory generators.

# 7 Conclusion

Numerous works on trajectory optimization in robotics have explored different techniques to find optimal trajectories [Bes92] [LLO91] [Hol84] [BDG85] [LCL83]. They usually only focus on algorithmic and numerical aspects, but they don't use precise models of the real physical limitations such as the thermic one considered here, and they don't take into account the integration of the robots in an industrial robotic cell which is usually very imperfectly modelized. These two aspects can't be neglected if we want to reach the maximum performances of a robot in real industrial working conditions, i.e. when the robot is integrated in a complex robotic cell.

We have pointed out here in the first section that the structure of a robotic application isn't directly compatible with this optimization problem we have to deal with, but it has a great property: it is decomposable. This property allows to split the optimization in two levels. We have derived then two algorithms for these two levels using different optimization techniques:

- an optimal profile generator which uses an optimization based on precise models of the robot and only needs a limited view of the robot task, solving a problem of calculus of variations using a direct method,
- a global optimization algorithm with hardware in the loop which allocates energetic ressources to each point to point motion taking into account all the imperfectly modelized interactions between the robot and the cell, based on optimization techniques without derivatives and on penalty methods to take into account the constraints.

The experimental results obtained on a real Stäubli Rx90B manipulator robot with these algorithms are excellent! Most importantly, they are able to adapt the behavior of the robot to changes in the task without any intervention from the operator. On top of that, the resulting optimal velocity profiles appear to be 5 to 10% faster than classical BANG-zero-BANG profiles, inducing a dramatic increase of productivity of the whole robotic cell. This work is protected by patent applications.

# References


[BDG85] James Bobrow, S. Dubowsky, and James Gibson. Time optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 1985.

[Bes92] Yasmina Bestaoui. On line reference trajectory definition with joint torque and velocity constraints. *The International Journal of Robotics Research*, 1992.

[BGLS03] Joseph-Frédéric Bonnans, Jean-Charles Gilbert, Claude Lemaréchal, and Claudia Sagastizabal. *Numerical Optimization : Theoritical and Practical Aspects*. Springer, Collection, 2003.

[BH75] Arthur Bryson and Yu-Chi Ho. *Applied Optimal Control, Optimization, Estimation and Control*. Taylor and Francis, 1975.

[BM03] Geoffrey Biggs and Bruce MacDonald. A survey of robot programming systems. *Proceedings of the Australasian Conference on Robotics and Automation in Brisbane, Australia*, 2003.

[CST97] A.R. Conn, K. Scheinberg, and Ph.L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *ISMP97, Lausanne*, 1997.

[Den01] Denso Corp. Motor controller. *Japan Patent JP2001292586*, 2001.

[Fan95] Fanuc Ltd. Displaying method for duty of industrial robot. *Japan Patent JP7087787*, 1995.

[Fle87] Roger Fletcher. *Practical Methods of Optimization, Second Edition*. John Wiley and Sons, 1987.

[Hol84] John Hollerbach. Dynamic scaling of manipulator trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 1984.

[Hor93] Roberto Horowitz. Learning control of robot manipulators. *ASME Journal of Dynamic Systems Measurement and Control 50th Anniversary Issue*, 1993.

[KD99] Wisama Khalil and Etienne Dombre. *Modélisation, identification et commande des robots*. Hermes Science Publications, 1999.

[Kob88] Kobe Steel Ltd. Method for foreseeing working limit of teaching playback type robot. *Japan Patent JP63126009*, 1988.

[LCL83] Chun-Sin Lin, Po-Rong Chang, and J.Y.S. Luh. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, 1983.

[LLO91] Alessandro De Luca, Luca Lanari, and Giuseppe Oriolo. A sensitivity approach to optimal spline robot trajectories. *Automatica*, 1991.

[Lon00] Richard W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control, Vol. 73, No 10, p.930-954*, 2000.

[LZT97] Craig Lawrence, Jian Zhou, and André Tits. User's guide for cfsqp version 2.5 : A c code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical report, Electrical Engineering Department and Institute for Systems Research, University of Maryland, 1997.

[Mat89] Matsuhita Electric Ind. co Ltd. Controller for industrial robot. *Japan Patent JP1020990*, 1989.

[Pin93] Enid Pinch. *Optimal Control and the calculus of Variations*. Oxford Science Publications, 1993.

[Pow64] Mickael Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 1964.

[Pow98] Mickael Powell. Direct search algorithms for optimization calculations. *Acta Numerica, Vol. 7, Cambridge University Press*, 1998.

[Pow04] Mickael Powell. The NEWUOA software for unconstrained optimization without derivatives. *40th Workshop on Large Scale Nonlinear Optimization (Erice, Italy)*, 2004.

[TP98] Jean Taine and Jean-Pierre Petit. *Transferts thermiques, Mécanique des fluides anisothermes*. Dunod, 1998.

[Win73] D. Winfield. Function minimization by interpolation in a data table. *Journal of the Institute of Mathematics and its applications*, 1973.

[WW90] Thomas Wonnacott and Ronald Wonnacott. *Statistique : Economie, Gestion, Sciences, Médecine*. Economica, 1990.

[Žla96] L. Žlajpah. On time optimal path control of manipulators with bounded joint velocities and torques. *Proceedings IEEE Int. Conf. on Robotics and Automation*, 1996.