Energy Minimization in Time-Constrained Robotic Tasks via Sequential Quadratic Programming

Marco Faroni, Domenico Gorni, Antonio Visioli

Energy Minimization in Time-Constrained Robotic Tasks via Sequential Quadratic Programming

Marco Faroni
Dipartimento di Ingegneria Meccanica e Industriale
University of Brescia
Brescia, Italy
Email: m.faroni003@unibs.it

Domenico Gorni
Dipartimento di Ingegneria Meccanica e Industriale
University of Brescia
Brescia, Italy
Email: domenico.gorni@unibs.it

Antonio Visioli
Dipartimento di Ingegneria Meccanica e Industriale
University of Brescia
Brescia, Italy
Email: antonio.visioli@unibs.it

***Abstract*— Reduction of the energy consumption in robotized processes is a key issue in nowadays manufacturing. In this paper, we propose a simple approach to energy minimization of robotic tasks with assigned cycle time based on sequential quadratic programming. The method aims at re-shaping a given timing law in the sense of energy saving, without modifying the desired path and the given cycle time. Thanks to the iterative linearization of the nonlinear time-constraint, the resulting minimization problem is solved by only using common quadratic programming solvers, making the method suitable for a direct implementation in robot industrial controllers. At first, the method is devised by only considering the kinematics of the manipulator. The dynamic model is then straightforwardly included, without significantly increasing the complexity of the method. Validation in simulation environment is provided in order to show the effectiveness of the methodology.**

***Index Terms*—Industrial robotics, energy minimization, trajectory optimization, quadratic programming.**

I. Introduction

Nowadays energy efficiency as well as the conservation of energy and natural resources are key aspects in the economic sector. The target of reduction of CO_2 emission is strictly correlated to the reduction of energy waste. In fact, companies belonging to many different sectors have been stimulated by the European Energy Management Standard (EN 6001) to reduce and monitor their energy consumption. Furthermore, during the last years, the price of energy is continuously increasing [1], [2]. Companies therefore react moving from the reduction of production time to the optimal trade-off between energy consumption and time. Recent statistics show that one of the major consumers of energy is actually the manufacturing industry.

During the last decades, the number of robots implemented in manufacturing has exponentially increased, moving companies to optimize their energy consumption in order to reduce the total amount of consumed energy [3]. For example, measurements show that an average 200-kg-payload robot in body shop yearly consumes around 8 $[MWh]$ and, obviously, the most part of this energy is spent during the movement [4], [5]. In a typical automotive car body, all the sheet metal parts are joined together by up to 4000 weld spots. Furthermore, other joining methods are gluing, arc welding and stud welding. In this kind of plants hundreds of robots that work unceasingly are involved [6], [7]. These data point out that, in order to satisfy economic efficiency criteria, industries are more and more interested in robot with lower energy consumption for comparable process time and accuracy.

The problem of computing optimal energy consumption paths and trajectories has been subject of research since the late 1960s [8]. In the last years, many researches have been carried out trying to solve it, however there is still the lack of a simple methodology that can be easily implemented. First of all, the energy minimization problem can be addressed from two different points of view based on the robotic application. The first approach concerns about finding optimal geometrical paths which can globally decrease the total energy consumption. For instance, if the robot has to execute a task like pick and place, the only geometrical constrains are the starting and the ending positions (and a few intermediate waypoints at most). Therefore, the geometrical path can be devised in such a way that the energy consumption is minimized. The second approach is followed in case of predefined geometrical paths. In this case, a decrease in the energy consumption can be obtained by acting on the timing law applied to the manipulator actuators by means of change in jerk, acceleration and velocity reference values.

This paper deals with this second class of strategies. In this framework, most of the proposed approaches set up a nonlinear optimization problem, which minimizes a cost function proportional to the energy consumption and takes into account the manipulator limits (e.g., configuration, velocity, acceleration and torque limits) in the constraints [4], [9], [10], [11]. However, these methodologies usually require heavy computational burden and their implementation may often result to be cumbersome, as also pointed out in [12]. Heuristic approaches are also often adopted by practitioners [13], [14]. In this case, the timing law is devised by using smooth functions and by using an iterative approach to minimize velocity or acceleration peaks given the total task time and checking that the resulting motion satisfies the robot limits. These approaches are easily implementable, computationally light,

and do not require optimization solvers. However, compliance with the robot limits can be checked only *a posteriori*, and the resulting timing law is suboptimal with respect to the original problem. There is still the need of easily-implementable and effective techniques which can attract practitioners and boost the industrial application of optimization-based methods.

This was also pointed out in [12], where a nonlinear optimization approach was developed to minimize a simplified energy index (namely, the squared sum of the joint accelerations). Such approach still requires the use of nonlinear solvers and does not take into account the dynamics of the manipulator. Nevertheless, it gives very good results, with a reduction of the energy consumption in the order of 30%.

In this paper, we therefore follow a similar approach, aiming at tackling the need for nonlinear solvers in the resolution of the optimization problem and the lack of inclusion of the dynamics of the robot in the problem. First of all, the energy minimization problem is set up in a different way with respect to [12]: the energy consumption index is written as a quadratic cost function and all the robot limits are written as linear constraints in the optimization variable, which is given by the curvilinear abscissa along the nominal path. The only nonlinear constraint is therefore given by the imposition of the total execution time of the task. The optimization problem is then solved by means of a Sequential Quadratic Programming (SQP) approach [15], which consists in iteratively linearizing the nonlinear constraint around the solution obtained at the previous iteration. The proposed method therefore presents the great advantage of requiring just the implementation of Quadratic Programming (QP) techniques, which could also be easily implemented on industrial robot controllers. Moreover, we address the inclusion of the dynamic model of the manipulator and we show that, by following the approach presented in [16], this does not significantly increase the computational complexity of the problem.

The paper is organized as follows. In Section II, the basic energy-optimal time-constrained path tracking problem is defined. In Section III, a linearization approach, based on Taylor expansion, is applied to the presented problem in such a way that the original problem can be solved as a sequence of quadratic programs. Section IV extends the kinematic-based approach presented up to that point by including the dynamic model of the manipulator. Finally, in Section V, simulation results are presented to show the effectiveness of the methodology. Conclusions are given in Section VI.

II. Problem formulation

Consider a given trajectory in the joint space and a set of kinematic constraints for a robot manipulator. The strategy proposed in this paper aims at modifying such trajectory by minimizing the energy consumption, preserving the path tracking and the given execution time of the task. Therefore, considering a manipulator composed by n joints, the trajectory $\mathbf{q}(t)$ can be defined by means of the path-velocity decomposition approach as the composition of the following functions:

$$\begin{aligned} s &: [0,t_f] \to [0,s_f], \qquad t \mapsto s(t), \\ \mathbf{q} &: [0,s_f] \to \mathbb{R}^n, \qquad s \mapsto \mathbf{q}(s), \end{aligned} \tag{1}$$

where s represents the Euclidean distance from the beginning of the path, and $\mathbf{q}$ is a vector that contains all the joint coordinates respect to time $\mathbf{q}(t) = [q_1(t), \ldots, q_n(t)]^T$. Furthermore t_f is the total traveling time and $s_f = s(t_f)$ is the total length of the path. As this paper deals with path tracking problems, we reasonably assume that $\dot{s}(t) > 0 \; \forall \, t \in]0, t_f[$ and $\dot{s}(t) \geq 0$ in the extreme points $t = 0$ and $t = t_f$.

For a given path the joints velocities and accelerations can be rewritten as a function of the defined scalar path coordinate trajectory, as follows:

$$\begin{aligned} \dot{\mathbf{q}}(t) &= \mathbf{q}'(s(t))\dot{s}(t), \\ \ddot{\mathbf{q}}(t) &= \mathbf{q}'(s(t))\ddot{s}(t) + \mathbf{q}''(s(t))\dot{s}(t)^2, \end{aligned} \tag{2}$$

where $\dot{s} = ds/dt$, $\ddot{s} = d^2s/dt^2$, $\mathbf{q}'(s) = d\mathbf{q}/ds$, and $\mathbf{q}''(s) = d^2\mathbf{q}/ds^2$.

In case the dynamic model of the manipulator is not available, it is possible to have an approximate measure of the energy consumption as a weighted sum of squared angular accelerations for all the joints [17]. Therefore the problem of minimizing the energy consumption can be written as:

$$\begin{aligned} \underset{s}{\text{minimize}} \quad & \int_0^{t_f} \ddot{\mathbf{q}}(t)^T \ddot{\mathbf{q}}(t) dt \\ \text{subject to} \quad & s(0) = 0, \\ & s(t_f) = s_f \\ & \dot{s}(0) = \dot{s}_0 \\ & \dot{s}(t_f) = \dot{s}_f \\ & \dot{s}(t) \geq 0 \\ & \underline{\dot{\mathbf{q}}} \leq \dot{\mathbf{q}}(t) \leq \overline{\dot{\mathbf{q}}} \\ & \underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}}(t) \leq \overline{\ddot{\mathbf{q}}} \end{aligned} \tag{3}$$

where $\dot{s}_0$ and $\dot{s}_f$ are the initial and final velocities (typically equal to zero), while $\underline{\dot{\mathbf{q}}}$, $\overline{\dot{\mathbf{q}}}$, $\underline{\ddot{\mathbf{q}}}$, and $\overline{\ddot{\mathbf{q}}}$ are, respectively, the lower and upper joints velocity constraints and the lower and upper joints acceleration constraints.

It is clear that the proposed performance index is nonlinear with respect to the path coordinate s, however it is possible to reformulate the optimal control problem (3) as an optimal control problem with linear system dynamics subject to nonlinear state dependent constraints. Indeed, a smart variable change has been proposed in [16]. Even though this requires a further set of equality constraints, the nonlinearity is moved from the performance index to a constraint. Firstly, a change in the integration variable, from t to s is required, so that the total time of the trajectory is rewritten as:

$$t_f = \int_0^{t_f} 1 dt = \int_{s(0)}^{s(t_f)} \frac{1}{\dot{s}} ds \tag{4}$$

Secondly, the following change of variables is applied:

$$\begin{aligned} a(s) &= \ddot{s}, \\ b(s) &= \dot{s}^2. \end{aligned} \tag{5}$$

The relation between the new optimization variables $a(s)$ and $b(s)$ is given by the following additional equality constraint:

$$\dot{b}(s) = b'(s)\dot{s} = 2a(s)\dot{s}, \tag{6}$$

which implies

$$b'(s) = 2a(s). \tag{7}$$

It is now possible to rewrite Problem (3) in such a way that the new optimization variables are $a(s)$ and $b(s)$ instead of s, that is:

$$\begin{aligned} \underset{a,b}{\text{minimize}} \quad & \int_0^{s_f} (\mathbf{q}'(s)a(s) + \mathbf{q}''(s)b(s))^T (\mathbf{q}'(s)a(s) + \mathbf{q}''(s)b(s))ds \\ \text{subject to} \quad & b(0) = \dot{s}_0^2, \\ & b(s_f) = \dot{s}_f^2, \\ & b'(s) = 2a(s), \\ & b(s) \geq 0, \\ & (\mathbf{q}'(s))^2 b(s) \leq (\dot{\overline{\mathbf{q}}})^2, \\ & \ddot{\underline{\mathbf{q}}} \leq \mathbf{q}(s)'a(s) + \mathbf{q}(s)''b(s) \leq \ddot{\overline{\mathbf{q}}}. \end{aligned} \tag{8}$$

The resulting optimization problem is therefore quadratic and can be easily solved by using conventional QP solvers. Note that the cost function in (8) is not exactly equivalent to the cost function in (3). In fact, the new cost function minimizes the integral of squared acceleration along the curvilinear abscissa s, while the original problem (3) minimizes the integral of the squared acceleration along time. However, in this way, the new cost function results to be quadratic, with significant benefits from the computational point of view.

Furthermore, Problem (8) is not complete yet, as none of the stated constraints accounts for the total traveling time. Thus, a further constraint has to be added, that is,

$$t_f = \int_{s(0)}^{s(t_f)} \frac{1}{\sqrt{b(s)}} ds. \tag{9}$$

This equality constraint is nonlinear. The next section addresses the linearization of this constraint in order to solve the optimization problem by means of a sequence of quadratic programs.

III. Linearization of the Problem and Terminal Conditions

The longitudinal path length $s \in [0, s_f]$ is discretized by means of a gridpoint $\{s_k\}, k = 0, \ldots, K$, and $a(s)$ and $b(s)$ are considered to be constant along the discretized intervals. Therefore, define $\mathbf{a} = [a_0, \ldots, a_K]$ and $\mathbf{b} = [b_0, \ldots, b_K]$ where $a_k = a\left(\frac{s_k - s_{k-1}}{2}\right)$ and $b_k = b\left(\frac{s_k - s_{k-1}}{2}\right)$.

The constraint expressing the total traveling time in (9) can be rewritten as:

$$t_f = \int_{s(0)}^{s(t_f)} \frac{1}{\sqrt{b(s)}} ds = \sum_{k=1}^{K} \frac{1}{\sqrt{b_k}} \Delta s_k, \tag{10}$$

where $\Delta s_k = s_k - s_{k-1}$. Therefore, a single sample of time Δt_k, at each generic instant k, can be rewritten as a function of Δs_k and b_k, as follows:

$$\Delta t_k(b_k) = \frac{1}{\sqrt{b_k}} \Delta s_k. \tag{11}$$

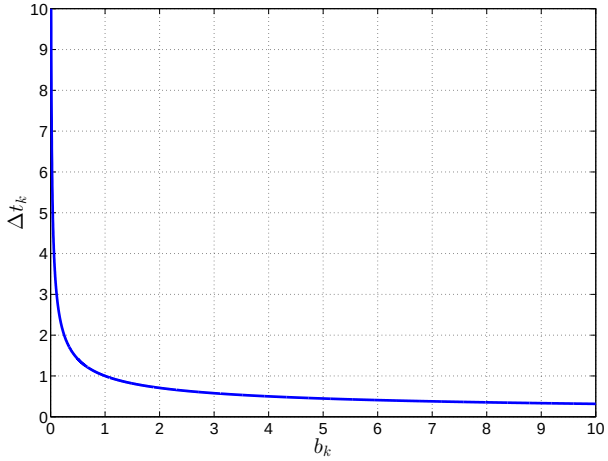

Fig. 1: Pseudo-hyperbole representing the nonlinear relation between Δt_k and $b_k(s)$.

Figure 1 shows the function that represents the nonlinear relation between Δt_k and $b_k(s)$. To overcome this nonlinearity, a linearization approach based on Taylor series expansion is applied. First of all, it is important to stress that this approach is based on the knowledge of the nominal timing law that has to be optimized. Therefore, the initial values of a_k and b_k on the whole path $[0, s_f]$ are assumed to be known and they are denoted by $\mathbf{a^0}$ and $\mathbf{b^0}$.

By applying a first-order Taylor series approximation to (11), we obtain:

$$\Delta \hat{t}_k(b_k) = \frac{\Delta s_k}{\sqrt{b_k^0}} - \frac{1}{2} \frac{\Delta s_k}{b_k^0 \sqrt{b_k^0}} (b_k - b_k^0) \tag{12}$$

where b_k^0 is the kth element of the set of known initial values of the optimization variable b_k and $\Delta \hat{t}_k(b_k)$ is the approximated value of $\Delta t_k(b_k)$ in the sense of Taylor expansion. Such a linearization is, obviously, an approximation of the pseudo-hyperbole around each initial value of the optimization variable b_k. However, it has to be pointed out that in many cases b_0 and b_K are equal to zero, therefore (12) results impossible to be applied. To overcome this issue, the first and the last time intervals are approximated by making use of the acceleration variables, that is:

$$\Delta t_k(a_k) = \sqrt{\frac{2\Delta s_k}{a_k}}, \quad \text{for} \quad k = 0 \wedge k = K. \tag{13}$$

Equation (13) is then linearized in the same way as (11) by means of its first-order Taylor expansion.

It is clear that each b_k is now constrained to move on a line instead of on the pseudo-hyperbole. Thus, to obtain a reliable solution, it is important to limit the admissible variation around the linearization point. This can be implemented by imposing a maximum admissible deviation between the pseudo-hyperbole and the line. This result in an additional constraint such that:

$$\underline{b_k} \leq b_k \leq \overline{b_k} \quad \forall\, k = 1, \ldots, K \tag{14}$$

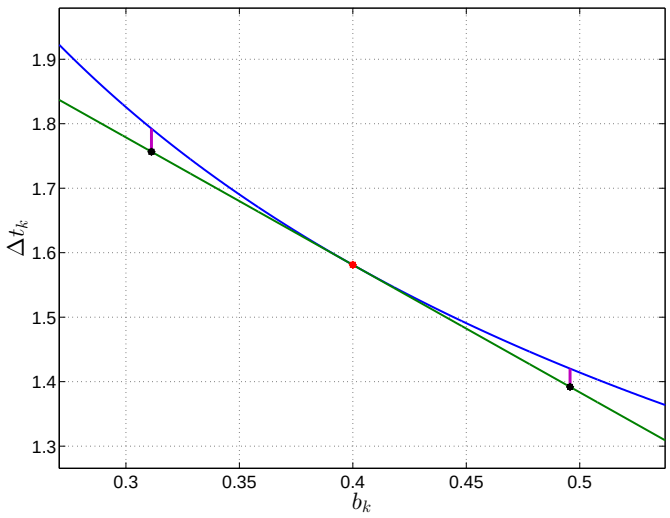

Fig. 2: Linearization of the pseudo-hyperbole. The red dot represent the starting point given by a generic b_k^0; the blue line is the pseudo-hyperbole that has to be approximated; the green line is the Taylor linearizing function, while the black dots represent the maximum admissible variation $\underline{b_k^0}$ and $\overline{b_k^0}$.

where $\underline{b_k}$ and $\overline{b_k}$ are calculated by imposing that the relative deviation between the pseudo-hyperbole and the line is smaller than a maximum value δ, i.e.:

$$\begin{aligned} \left|\frac{\Delta t_k(\underline{b_k}) - \Delta\hat{t}_k(\underline{b_k})}{\Delta t_k(\underline{b_k})}\right| &\leq \delta \\ \left|\frac{\Delta t_k(\overline{b_k}) - \Delta\hat{t}_k(\overline{b_k})}{\Delta t_k(\overline{b_k})}\right| &\leq \delta \end{aligned} \tag{15}$$

The considered linearized optimal problem therefore results:

$$\begin{aligned} \underset{\mathbf{a},\mathbf{b}}{\text{minimize}} \quad & (\mathbf{q}'(s_k)a_k + \mathbf{q}''(s_k)b_k)^T(\mathbf{q}'(s_k)a_k + \mathbf{q}''(s_k)b_k) \\ \text{subject to} \quad & b_0 = \dot{s}_0^2, \\ & b_K = \dot{s}_f^2, \\ & \mathbf{b}' = 2\mathbf{a} \\ & \mathbf{b} \geq 0 \\ & (\mathbf{q}'(s_k))^2 b_k \leq (\overline{\ddot{\mathbf{q}}}(s_k))^2 \\ & \underline{\ddot{\mathbf{q}}} \leq \mathbf{q}(s_k)'a_k + \mathbf{q}(s_k)''b_k \leq \overline{\ddot{\mathbf{q}}} \\ & \left|\frac{\Delta t_k(\underline{b_k}) - \Delta\hat{t}_k(\underline{b_k})}{\Delta t_k(\underline{b_k})}\right| \leq \delta \\ & \left|\frac{\Delta t_k(\overline{b_k}) - \Delta\hat{t}_k(\overline{b_k})}{\Delta t_k(\overline{b_k})}\right| \leq \delta \\ & \Delta\hat{t}_k(b_k) = \frac{\Delta s_k}{\sqrt{b_k^0}} - \frac{1}{2}\frac{\Delta s_k}{b_k^0\sqrt{b_k^0}}(b_k - b_k^0) \;\; k = 2,\ldots,K\text{–}1 \\ & \Delta\hat{t}_k(a_k) = \sqrt{\frac{2\Delta s_k}{a_k^0}} - \frac{1}{2}\frac{\sqrt{2\Delta s_k}}{a_k^0\sqrt{a_k^0}}(a_k - a_k^0) \;\; k = \{1, K\} \end{aligned} \tag{16}$$

Problem (16) is based on the linearization of (11) and (13). The error introduced by the linearization is smaller when the approximated solution is close to the initial conditions. For this reason, we adopt a sequential programming approach, by solving (16) iteratively, and using the last solutions as new initial timing law for the next optimization, until the difference between successive solutions is smaller than a given threshold ε. This procedure is summarized in Algorithm 1.

Algorithm 1 Computation of the optimal motion law

Input: $\mathbf{a}^0, \mathbf{b}^0$
Output: $\mathbf{a}, \mathbf{b}$
repeat
  $\Delta\hat{t}_1 \leftarrow \sqrt{\frac{2\Delta s_1}{a_1^0}} - \frac{1}{2}\frac{\sqrt{2\Delta s_1}}{a_1^0\sqrt{a_1^0}}(a_1 - a_1^0)$
  $\Delta\hat{t}_K \leftarrow \sqrt{\frac{2\Delta s_K}{a_K^0}} - \frac{1}{2}\frac{\sqrt{2\Delta s_K}}{a_K^0\sqrt{a_K^0}}(a_K - a_K^0)$
  for $i = 2$ to $K-1$ **do**
    $\Delta\hat{t}_k \leftarrow \frac{\Delta s_k}{\sqrt{b_k^0}} - \frac{1}{2}\frac{\Delta s_k}{b_k^0\sqrt{b_k^0}}(b_k - b_k^0)$
  end for
  $\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} \leftarrow$ argmin Problem (16)
  $max\_value \leftarrow \max\left|\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} - \begin{bmatrix}\mathbf{a^0}\\ \mathbf{b^0}\end{bmatrix}\right|$
  $\mathbf{a^0} \leftarrow \mathbf{a}$
  $\mathbf{b^0} \leftarrow \mathbf{b}$
until $max\_value \leq \varepsilon$

Depending on the value of δ and ε, it is possible to obtain a faster or slower convergence of the recursive algorithm and of course, a bigger or a lower proximity to the global optimal minimum of the nonlinear problem.

IV. Reformulation of the problem given the dynamic model

Consider the dynamic model of the manipulator in the form:

$$\tau(t) = \mathbf{M}(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + \mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + \mathbf{f}(\mathbf{q}(t))\text{sgn}(\dot{\mathbf{q}}(t)) + \mathbf{g}(\mathbf{q}(t)) \tag{17}$$

where $\mathbf{M}(\mathbf{q}(t))$ is a positive definite inertia matrix and $\mathbf{C}(\mathbf{q}(t),\dot{\mathbf{q}}(\mathbf{t}))$ is a matrix accounting for Coriolis effects. Then, $\mathbf{f}(\mathbf{q}(t))$ is a vector that describes the Coulomb friction forces while $\mathbf{g}(\mathbf{q}(t))$ denotes the vector accounting for gravity.

By applying (2) into (17) we obtain:

$$\tau(s(t)) = \mathbf{M_s}(s(t))\ddot{s}(t) + \mathbf{C_s}(s)\dot{s}(t)^2 + \mathbf{g_s}(s) \tag{18}$$

where

$$\begin{aligned} \mathbf{M_s}(s(t)) &= \mathbf{M}(\mathbf{q}(s(t)))\mathbf{q}'(s(t)) \\ \mathbf{C_s}(s(t)) &= \mathbf{M}(\mathbf{q}(s(t)))\mathbf{q}''(s(t)) + \mathbf{C}(\mathbf{q}(s(t)), \mathbf{q}'(s(t)))\mathbf{q}'(s(t)) \\ \mathbf{g_s}(s(t)) &= \mathbf{f}(\mathbf{q}(s(t)))\text{sgn}(\mathbf{q}'(s(t))) + \mathbf{G}(\mathbf{q}(s(t))) \end{aligned} \tag{19}$$

By using the change of variable described in (5), the torque $\tau(s)$ results to be linear in the optimization variables $a(s)$ and $b(s)$. This means that a quadratic cost function can be easily set up to minimize the energy consumption in terms of

integral of the squared torques. Similarly to (8), the following optimization problem results:

$$\begin{aligned}
\underset{a,\,b}{\text{minimize}} \quad & \int_0^{s_f} (\tau(s(t)))^T (\tau(s(t)))ds \\
\text{subject to} \quad & b(0) = \dot{s}_0^2, \\
& b(s_f) = \dot{s}_f^2 \\
& b'(s) = 2a(s) \\
& b(s) \geq 0 \\
& (\mathbf{q}'(s))^2 b(s) \leq (\dot{\overline{\mathbf{q}}})^2 \\
& \ddot{\underline{\mathbf{q}}} \leq \mathbf{q}(s)' a(s) + \mathbf{q}(s)'' b(s) \leq \ddot{\overline{\mathbf{q}}} \\
& \underline{\tau} \leq \mathbf{M_s}(s(t))a(s) + \mathbf{C_s}(s(t))b(s) + \mathbf{g_s}(s(t)) \leq \overline{\tau}
\end{aligned} \tag{20}$$

where $\underline{\tau}(s)$ and $\overline{\tau}(s)$ are the torque lower and upper bounds. The solution of such problem is analogous to the above mentioned kinematic case as described in Section III.

V. Results

In order to verify the performance of the proposed methodology, the algorithm has been used to plan the trajectory on a 6-DOF manipulator. In particular, we consider a six-axis Efort ER3A industrial manipulator that has to perform a trajectory defined in the joint space. The target is, as mentioned before, to complete a task preserving the time and the path tracking by consuming as less energy as possible.

The methodology described in this work is implemented off-line, in the sense that the algorithm is completely executed before the start of the motion. Differently from many other nonlinear algorithms developed to solve the same problem, this methodology requires only a standard quadratic programming solver. This kind of solvers are simple and fast enough to be easily included directly in the robotic controller [18], [19]. This experimental case study aims at demonstrating some of the features presented in the previous sections. No dynamic model is used in order to show the effectiveness of the methodologies in usual industrial case, where the dynamic model is often unknown or unreliable. For the sake of simplicity the trajectory is defined in the joint space. In fact, even though the tasks in industrial robotics are mainly defined in the operational space, from a practical point of view the motion is controlled acting directly on each single joint.

First of all, in order to improve the performance of the algorithm, an additional assumption has to be done. During the motion planned by a standard industrial controller, the most used motion laws (for instance, the seven-segments or the cycloidal ones) present the maximum velocity in the central part of the movement. The maximum accelerations and decelerations are usually mainly concentrated at the beginning and at the end of the trajectory. Due to this fact, it is recommended, in order to obtain a better convergence of the algorithm, to have a thicker path sampling in these parts of the trajectory. In case of dense sampling, the number of optimization variables increases significantly, causing an increment of the required computational time. We therefore

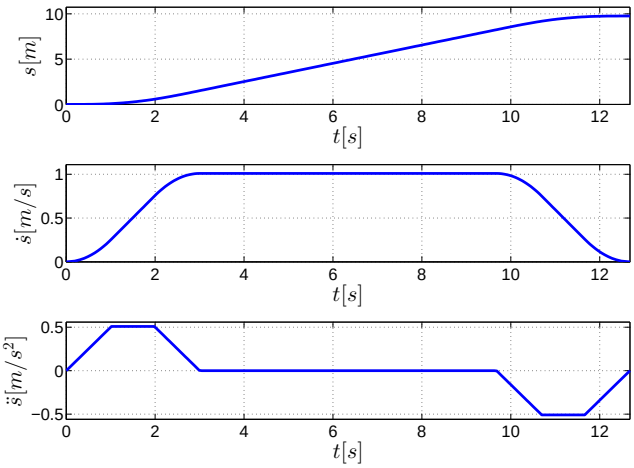

Fig. 3: Position s, velocity $\dot{s}$ and acceleration $\ddot{s}$ of the applied law of motion before the optimization is run.

choose an irregular sampling of the path length $\{s_k\}$ by making use the well-known Chebyshev nodes, which gives:

$$s_k = \frac{1}{2}s_f + \frac{1}{2}s_f \cos\left(\frac{2k-1}{2k}\pi\right), \quad k = 1, \ldots, K. \tag{21}$$

Here we consider a trajectory defined in the path coordinates $s(t)$ and its time derivatives. The timing law applied to the manipulator is a seven-segments one (see Figure 3), which has been generated based on the following values:

- $s_f = 10$ $[rad]$;
- $s_0 = 0$ $[rad]$;
- $\dot{s}_0 = 0$ $[rad/s]$;
- $\dot{s}_f = 0$ $[rad/s]$;
- $\dot{s}_{max} = 1$ $[rad/s]$;
- $\ddot{s}_{max} = 0.5$ $[rad/s^2]$;
- $\dddot{s}_{max} = 0.5$ $[rad/s^3]$;

The total execution time of the trajectory therefore results to be $t_f = 12.5$ $[s]$

The path length has been discretized by using a number of gridpoints $K = 200$. Furthermore, the tolerance values δ and ε have been chosen respectively equal to 2% and 1.5%. The choice of these two values is very important as the convergence of the algorithm and the proximity of the linearized solution with respect to the global optimum one depends mainly on these ones. First of all, it is important to stress that the output tolerance is based on the maximum displacement value between the set of solution at the iteration i and those at the iteration $i-1$. Therefore, in order not to obtain a fake convergence it is recommended to choose the δ value bigger than the ε one. The optimization variables have to be free to move more than the output tolerance in order to be sure that the resulting set of optimal values has actually converged to the global optimal solution. Obviously, in order to have a better precision, the values of these two parameters have to be small. Furthermore, increasing too much these two values will cause a bad convergence due to the lost of the good approximation

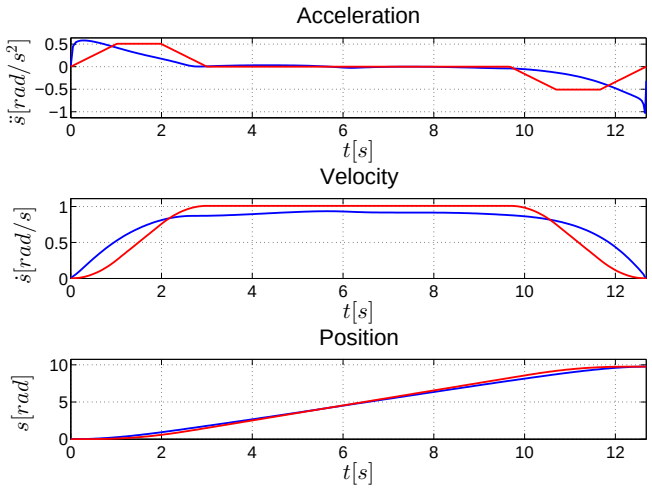

Fig. 4: Red: timing law obtained by the proposed method. Blue: original timing law.

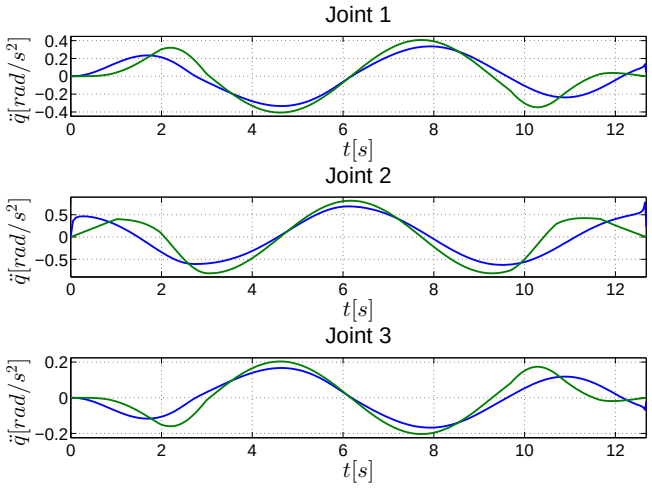

Fig. 5: Blue: joint accelerations calculated by the proposed method. Green: original joint accelerations. The three plots are related to joint 1, 2, and 3 starting from the top to the bottom.

hypothesis. In practice, the values are moving on a straight line that is actually approximating, with a good precision, the pseudo-hyperbole only when it is close to the tangent point (see Figure 2).
The nominal trajectory is defined as:

$$\mathbf{q} = \begin{bmatrix} 0.4(1-\cos(s(t))) \\ 0.8\sin(s(t)) \\ 0.2\cos(s(t)) \\ 1.1(1-\sin(s(t))) \\ 0.3(1-\cos(s(t))) \\ 1.4\sin(s(t)) \end{bmatrix} \tag{22}$$

where $s(t)$ is given by the seven-segment timing law described in Figure 3. The new timing law obtained by means of the proposed method is shown in Figure 4.

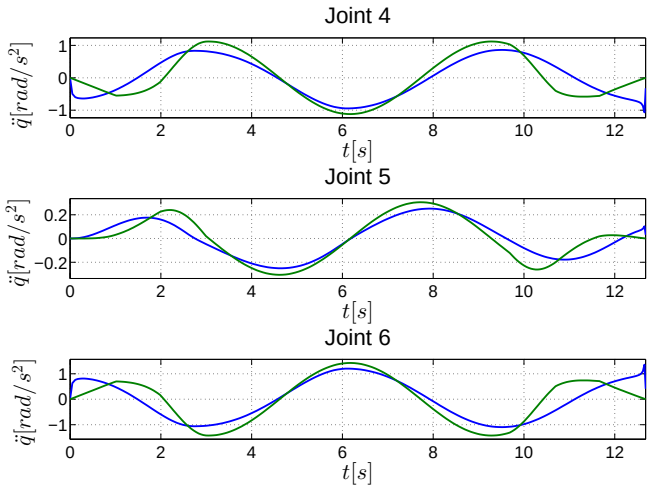

Fig. 6: Blue: acceleration calculated by the proposed method. Green: original acceleration profile. The three plots are related to joint 4, 5, and 6 starting from the top to the bottom.

TABLE I: Comparison of the energy consumption (computed as in (23)) for the original and the optimized timing law.

Joint	Original timing law	Optimized timing law	%saving
1	0.72	0.51	29
2	3.18	2.40	24.5
3	0.181	0.13	29
4	6.01	4.54	24.4
5	0.41	0.29	29
6	9.73	7.35	24.5
TOT	20.23	15.21	24.8

Furthermore, in Figures 5 and 6, the resulting joint accelerations of the 6-DOF manipulator are presented.

Table I shows the energy consumption of each joint in case of the original and the optimized timing law. The energy consumption is calculated as:

$$\sum_{k=1}^{K} \ddot{\mathbf{q}}(k)^T \ddot{\mathbf{q}}(k) \Delta \hat{t}_k. \tag{23}$$

It is possible to notice that the optimized trajectory allows a reduction of the energy consumption of about 25%, which is indeed a significant improvement.

As regards the computational time required by the algorithm, the convergence time for the proposed example resulted to be equal to 29 seconds, taking 19 iteration to satisfy the exit tolerance (the simulations have been executed in Matlab, on a standard laptop mounting an i7-4500U processor).

VI. Conclusions

In this paper we have presented a very simple and linear methodology that can be used to solve an energy minimization in time constrained robotic tasks. This kind of problem is very popular in industries where the energy minimization problem is, at the present day, an important topic. This algorithm has many advantages, for example it can be used both with or without the knowledge of the dynamic model of the manipulator.

Furthermore, it preserves the path tracking and it is constrained to re-shape the motion law without changing the total time of the task. The linearization and, therefore, the possibility to use simple linear quadratic programming solvers makes it suitable for a direct implementation in the robot motion controller. However, the linearization also introduces the disadvantage of tuning the values of two tolerance parameters, which can affect the convergence performance.

Energy is actually saved by modifying the original timing law by minimizing the sum of the squared acceleration values subject to many different equality and inequality constraints. A simple case study has shown that the application of the proposed methodology allows a significant reduction of the energy consumption.

REFERENCES

[1] F. Unander, "Decomposition of manufacturign energy use in iea countries, how do recent development compare with historical long-term trandes," *Applied Energy*, vol. 84, 2007.

[2] Paryanto, M. Brossog, M. Bornschlegl, and J. Franke, "Reducing the energy consumption of industrial robots in manifacturing systems," *The International Journal of Advanced Manifacturing Technology*, pp. 1–14, 2015.

[3] I. F. R., "https://ifr.org/news/world-robotics-report-2016," 2016.

[4] D. Meike and L. Ribickis, "Energy efficient use of robotics in the automobile industry," in *Proceedings of the 15th International Conference on Advanced Robotics*, Tallin (Estonia), 2011, pp. 507–511.

[5] O. Maimon, E. Profeta, and S.Singer, "Energy analysis of robot task motions," *Journal of Intelligent and Robotic System*, vol. 4, pp. 175–198, 1991.

[6] S. Björkenstam, D. Gleeson, and R. Bohlin, "Energy efficient and collision free motion of industrial robots using optimal control," in *Proceedings of the 9th IEEE International Conference on Automation Science and Engineering*, Madison (USA), 2013.

[7] D. Meike, M. Pellicciari, and G. Berselli, "Energy efficient use of multirobot production line in the automotive industry: Detailed system modeling and optimization," *IEEE Transactions On Automation Science and Engineering*, vol. 11, pp. 798–809, 2014.

[8] G. Field and Y. Stepanenko, "Iterative dynamic programming: An approach to minimum energy trajectory planning for robotic manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis (USA), 1996, pp. 2755–2760.

[9] O. Wigström, B. Lennartson, A. Vegano, and C. Breitholdtz, "High-level scheduling od energy optimal trajectories," *IEEE Transactions on Automation Science and Engineering*, vol. 10, pp. 57–64, 2013.

[10] K. G. Shin and N. D. Mckay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 6, pp. 491–500, 1986.

[11] Z. Shiller, "Time-energy optimal control of articulated system with geometric path constraints," *Robotics and Automation*, vol. 4, pp. 2680–2685, 1994.

[12] S. Riazi, K. Bengtsson, O. Wingström, E. Vidarsson, and B. Lennartson, "Energy optimization of multi-robot systems," in *Proceedings of the 2015 International Conference on Automation Science and Engineering*, Gothenburg (Sweden), 2015, pp. 1345–1350.

[13] K. K. Ayten, P. Iravani, and M. N. Sahinkaya, "Optimum trajectory planning for industrial robots through inverse dynamics," in *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics*, Setubal (Portugal), 2011, pp. 105–110.

[14] B. Martin and J. Bobrow, "Minimum effort motions for open chain manipulators with task-dependent end-effector constraints," *Robotics and Automation*, vol. 3, pp. 2044–2049, 1997.

[15] Z. Zhu, "An efficient sequential quadratic programming algorithm for nonlinear programming," *Journal of Computational and Applied Mathematics*, vol. 175, pp. 447–464, 2005.

[16] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, pp. 2318–2327, 2009.

[17] S. Riazi, O. Wigström, K. Bengtsson, and B. Lennartson, "Energy and peak power optimization of time-bounded robot trajectories," *IEEE Transactions on Automation Science and Engineering*, vol. 14, pp. 646–657, 2017.

[18] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[19] C. Hildreth, "A quadratic programming procedure," *Michigan Agricultural Experiment Station*, vol. 2001, pp. 79–95, 1957.