SAPIENZA
UNIVERSITÀ DI ROMA

PhD Thesis in Automatica, Bioingegneria e Ricerca Operativa

XXXI Ciclo

Optimal Redundancy Control for Robot Manipulators

Khaled Al Khudir

Advisor Prof. Alessandro De Luca

October 2018

Dipartimento di Ingegneria Informatica, Automatica e Gestionale (DIAG)
Sapienza Università di Roma

To my daughter, Yafa.

Abstract

Optimal control for kinematically redundant robots is addressed for two different optimization problems. In the first optimization problem, we consider the minimization of the transfer time along a given Cartesian path for a redundant robot. This problem can be solved in two steps, by separating the generation of a joint path associated to the Cartesian path from the exact minimization of motion time under kinematic/dynamic bounds along the obtained parametrized joint path. In this thesis, multiple sub-optimal solutions can be found, depending on how redundancy is locally resolved in the joint space within the first step. A solution method that works at the acceleration level is proposed, by using weighted pseudoinversion, optimizing an inertia-related criterion, and including null-space damping. The obtained results demonstrate consistently good behaviors and definitely faster motion times in comparison with related methods proposed in the literature. The motion time obtained with the proposed method is close to the global time-optimal solution along the same Cartesian path. Furthermore, a reasonable tracking control performance is obtained on the experimental executed motions. In the second optimization problem, we consider the known phenomenon of torque oscillations and motion instabilities that occur in redundant robots during the execution of sufficiently long Cartesian trajectories when the joint torque is instantaneously minimized. In the framework of on-line local redundancy resolution methods, we propose basic variations of the minimum torque scheme to address this issue. Either the joint torque norm is minimized over two successive discrete-time samples using a short preview window, or we minimize the norm of the difference with respect to a desired momentum-damping joint torque, or the two schemes are combined together. The resulting local control methods are all formulated as well-posed linear-quadratic problems, and their closed-form solutions generate also low joint velocities while addressing the primary torque optimization objectives. Stable and consistent behaviors are obtained along short or long Cartesian position trajectories. For the two addressed optimization problems in this thesis, the results are obtained using three different robot systems, namely a 3R planar arm, a 6R Universal Robots UR10, and a 7R KUKA LWR robot.

Keywords: *Robotics, Optimization and Optimal Control, Motion Control, Redundant Robots, Dynamics.*

Acknowledgements

Firstly, I would like to express my sincere gratitude to my teacher and advisor Prof. Alessandro De Luca for the sustained support of my Ph.D. study and related research, motivation, and immense knowledge.

In addition, I would like to thank all Robotics Lab members who shared with me this journey. Special thank to Dr. Claudio Gaz for his continuous help and all useful discussions.

My sincere thanks also go to Ms. Ester Latini, the international students' advisor at DIAG, for her substantial help regarding my work and stay in Rome.

Also, I would like to thank all my friends for their motivation and nice refresh breaks during the day.

A big thanks to my parents for their endless encouragement, support, and patience for being far from them.

Last but not least, my great gratitude and thanks to Maram Khatib for being my colleague, friend, and wife. Thanks!

Finally, I am grateful for the financial support of my work, provided by the International Office of SAPIENZA University, represented by Mrs. Graziella Gaglione, and by the Robotics Lab of DIAG.

Contents

Introduction

Recent years have seen a significant increase in production of manipulators that have more joints than the conventional industrial robots which usually supported with a maximum six degrees of freedom. More joints give robots the ability to achieve their tasks taking into account different performance criteria and make it able to adapt with any change in the workspace while conserving their main task. However, these potential benefits should be traded off against a greater structural complexity of construction, such as mechanical (more links and actuators, sensors, costs) and more complicated algorithms for inverse kinematics and motion control.

A manipulator is kinematically redundant if the number n of its degrees of freedom (viz. joints) is larger than to those strictly needed to execute a given task of dimension m, or $n > m$. Redundancy can be exploited either in the Cartesian space to avoid collision with obstacles and to increase manipulability in specified directions, or in the joint space to avoid kinematic singularities, stay within the admissible joint ranges, minimize energy consumption, optimize execution time and any other quantitatively measurable criteria (Siciliano et al. [2008], Chiaverini et al. [2008]).

Following a prescribed geometric path with an end-effector tool is one of the most common tasks that robot manipulators perform in industrial applications. The path only determines the task geometry in the Cartesian space. If the velocity motion profile along the path is unspecified, it is often desirable to traverse the path in the least possible time while not violating actuator limits. While if a desired Cartesian velocity is also assigned, reducing the torque needed to execute the robot task is preferred.

In the first case, several algorithms have been proposed for the time-optimal path following problem under dynamic constraints, starting with the seminal works (Bobrow et al. [1985], Shin and McKay [1985]), refined later in (Slotine and Yang [1989], Shiller [1994]). The original idea was to work in the phase plane defined by the path parameter and its first time derivative. In Verscheure et al. [2009], the problem has been formulated in terms of convex optimal control, taking advantage of general numerical algorithms. More recently, an efficient and stable algorithmic tool called TOPP (Time-Optimal Path Parameterization), has been implemented in Pham [2014], solving the problem under dynamic as

well as kinematic constraints. All these results apply both to Cartesian and joint paths, but in the first case, it is implicitly assumed that the robot has as many joints as strictly needed for moving along the desired Cartesian path (non-redundancy).

A general and efficient global solution to the time-optimal control problem along Cartesian paths in kinematically redundant robots has not been found yet. In Galicki [2000], the problem was addressed by Pontryagin's maximum principle, looking for regular trajectories in an extended state space. An approach based on decomposition into non-redundant and redundant joints was introduced in Ma and Watanabe [2004]. These techniques are able to generate optimal solutions mainly for the case of degree of redundancy $n-m=1$. A non-convex numerical method with higher order inverse kinematics and null-space augmentation is proposed in Reiter et al. [2018].

A different way to tackle the problem is to separate it into two steps: first, a specific joint path is generated from the assigned Cartesian path, typically by local (or semi-global) inverse differential methods; then, motion time is exactly minimized under the given kinematic/dynamic bounds along the unique joint path thus found. This approach was pioneered in Chiacchio [1990], and later in Basile and Chiacchio [2003], providing satisfactory results. In their first step, robot redundancy was locally exploited using first-order differential inverse kinematic solutions that, e.g., increase robot manipulability along the tangent direction to the Cartesian path. Indeed, there are infinite ways to generate a path in the joint space within the first step of this procedure. The challenge is to obtain paths along which the optimal selection of the timing law (say, by the TOPP algorithm) achieves the fastest possible motion transfer. For this, a number of additional dynamic issues, such as those considered in Khatib [1987] and De Luca and Ferrajoli [2008], should be conveniently incorporated in the differential inversion of the Cartesian path.

In this thesis, we propose to generate a sequence of joint configurations by means of a second-order differential inverse kinematics scheme, using weighted pseudoinversion, optimizing locally an inertia-related criterion, and including judiciously a damping action in the null space of the task Al Khudir and De Luca [2018]. The obtained configurations are then interpolated with a parameterized path in the joint space, and an exact minimum time solution is computed using the TOPP algorithm. In case the initial robot configuration is not assigned a priori, we include also a kinematic optimization scheme to find the best initial joint configuration corresponding to the starting point of the Cartesian path. The performance of the new method was tested in a large number of Cartesian paths using different robot systems which returned definitely faster motion times in comparison with related methods proposed in the literature. The motion time obtained with the proposed method is close to the global time-optimal solution along the same Cartesian path.

For the second optimization problem considered in this thesis, when the Cartesian path is assigned together with the desired velocity profile, minimizing the torque needed to execute the given robot task is a common control objective where dynamics plays a major role. In the presence of redundancy, the authors in Hollerbach and Suh [1987] presented the first method for torque optimization, deriving the closed-form expression of the solution that instantaneously minimizes the norm of the joint torque, while executing a desired Cartesian trajectory. However, for longer movements, this local optimization method often

exhibits an unstable behavior in the form of motion oscillations, growing joint velocities, and sudden whipping torque effects with a practical loss of control. The obvious countermeasure to this would be to pursue a global torque optimization over the whole motion trajectory (Suh and Hollerbach [1987], Kazerounian and Wang [1988]), but this implies the off-line numerical solution of a two-point boundary value problem and the availability in advance of the entire desired task.

Wishing instead to keep an on-line redundancy resolution method, which is also suited for conversion into a sensor-based feedback control scheme, local optimization alternatives have been further explored. The minimization of the infinity-norm of the joint torque was proposed in Shim and Yoon [1997], but the torque explosion problem would still appear and extra inequality constraints had to be added to the algorithm. A joint decomposition formulation was presented in Ma [1996b] for torque optimization, with the aim of forcing a return to zero of the joint velocities at the end of the motion task. The relevance of keeping the joint velocity under control in order to address the instability problem was pointed out also in Ma and Nenchev [1996]. Special attention was given to the robot self-motion dynamics, providing also a first explanation for the good torque performance experienced in simulations when using a purely kinematic joint velocity minimization scheme Chen et al. [1994]. Based on this, balancing between the minimum velocity and the minimum torque solutions was proposed in Ma [1996a]. The instabilities that can occur when redundancy is solved at the second-order level (in terms of joint accelerations or torques) were analyzed in O'Neil and Chen [2000], relating them to the smallest singular value of the task Jacobian and to particular null-space accelerations that cause a quadratic growth of joint velocities. In O'Neil [2002], a complete characterization of the instability phenomenon was given, studying the self-motion manifold in the case of robots with one degree of redundancy. Interestingly, it was shown that simple damping of joint velocities in the null space of the task may not prevent large motion oscillations in the long run.

Unfortunately, the above mentioned local methods do not provide conclusive answers: instability problems are still encountered for longer movements, or they would fail for certain tasks and initial conditions, or countermeasures were proven to work only in relatively simple cases (i.e., when $n - m = 1$). Last but not least, most papers on model-based optimization of joint torques in redundant robots have presented only simulation results on simple manipulators. Among the few exceptions, there are Peters et al. [2008] and lately (Herzog et al. [2016], Mingo Hoffman et al. [2018]), where, however, the issue of instability in local torque minimization on long trajectories is not explicitly addressed.

In this thesis, we introduce two basic variations to the minimum torque norm (*MTN*) scheme of Hollerbach and Suh [1987] that address the issue of unstable joint motion in redundant robots Al Khudir et al. [2019]. In the first scheme, we propose the minimization of the joint torque norm over two successive discrete-time samples using a short preview window (possibly, in the next sampling instant). Suitable dynamic approximations are introduced in the estimation of the future robot state, so as to keep a linear-quadratic formulation for the problem, in a way similar to Duchaine et al. [2007]. Such a dynamic optimization scheme, that we denote model-based preview (*MBP*), can be seen as a compromise between local and global redundancy resolution methods, trying to inherit the best of both worlds —real-time simplicity and stable behavior.

Because of the presence of a predictive window in the future, *MBP* is similar to a model predictive control (*MPC*) approach. However, most linear Poignet and Gautier [2000] and nonlinear Tassa et al. [2012] *MPC* methods in robotics have not considered explicitly robot redundancy or on-line task constraints. When real-time execution is a must, different local approximations are being performed to reduce the computational effort (e.g., using a linear inverted pendulum predictive model in humanoid gait control Scianca et al. [2016]). The main difference with respect to *MPC* is that *MBP* does not need to compute/predict multiple system samples in a future window. Rather, in its basic version, our method uses just a single preview state, which may be placed close in time or further away from the current one, where the task-based equality constraint is also imposed. Computational efficiency is achieved even for a large number of degrees of freedom, thanks to the closed-form solution of a linear-quadratic (LQ) problem that is always guaranteed to be positive definite. On the other hand, contrary to the common case in *MPC*, bounds on the future state are not considered. While feasible in principle, such an extension is in fact not strictly needed because of the nature of the problem addressed, i.e., stable optimization of the motion torques.

In the second proposed control scheme, we choose the command torque that realizes the Cartesian task and is closest in norm to the desired torque, which is proportional to the current generalized momentum of the robot. This induces natural dynamic damping of the joint velocities, preventing their oscillations or growth, and we label the solution as minimum torque norm with damping (*MTND*). The two proposed modifications can also be combined, leading to a model-based preview scheme with damping (*MBPD*). Again, these schemes are formulated as well-posed LQ problems, providing efficiently the solution in closed form. The performance of the new controllers was tested in a large number of short and long motions, always yielding a stable behavior under torque optimization.

The present thesis is structured as follows. In Chapter 1 different techniques for local and global optimal redundancy resolution are reviewed. Chapter 2 includes different formulations of exploiting robot redundancy at the first- and second-order differential level for the time-optimal planning problem. Comparative results on different robots are considered including experimental evaluation. Chapter 3 presents the derivation of the proposed torque optimization methods with model-based preview, as well as the damping extensions. Simulation and experimental results are reported. The ideas and methods presented in Chapters 2 and 3 have been published as author's original work in Al Khudir and De Luca [2018] and Al Khudir et al. [2019].

1 Optimal Redundancy Resolution

1.1 Introduction

Different optimization techniques can be used to exploit robot redundancy. In general, they are separated into local and global methods. Local approaches can return closed form solutions which make it suitable for on-line applications. Although global methods can return a unique optimal solution, it still suffer from complex formulation and numerical process needed to solve boundary conditions.

Optimal redundancy resolution using local and global methods are reviewed in Sec. 1.2. Solving redundancy at the acceleration level is introduced in Sec. 1.3.

1.2 First-order methods

Suppose that the robot should perform a task in the Cartesian space described by the variables $\boldsymbol{x} \in \mathbb{R}^m$, with $m < n$, related to the robot joint space $\boldsymbol{q} \in \mathbb{R}^n$ by the task kinematics

$$\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{q}). \tag{1.1}$$

The differential relation at the first-order level is given by

$$\dot{\boldsymbol{x}}(t) = \frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{1.2}$$

where $\boldsymbol{J}$ is the $m \times n$ analytic task Jacobian. The redundancy of the robot with respect to the task can be resolved using different optimization approaches. Usually, these approaches optimize the desired criteria either instantaneously at the current time step (locally), or along the whole desired Cartesian path (globally) Chiaverini et al. [2008].

Local optimization

Different classes of methods for local optimal redundancy resolution are proposed Siciliano et al. [2008]. Some of them are Jacobian-based methods where a solution among infinite ones is chosen to optimize a suitable norm (usually weighted) using a standard linear-quadratic optimization problem. For example, the instantaneous minimization of the weighted system kinetic energy can be defined according to the Whitney [1969] as

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}} \; & G_l(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\, \dot{\boldsymbol{q}}^T \boldsymbol{W}(\boldsymbol{q}) \dot{\boldsymbol{q}} \\ & \text{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q}) \dot{\boldsymbol{q}}, \end{aligned} \tag{1.3}$$

where $\boldsymbol{W}(\boldsymbol{q})$ is a positive definite weighting matrix. The solution to (1.3) is obtained with the method of Lagrange multipliers Luenberger and Ye [2010]. Define the Lagrangian L as

$$L = G_l + \boldsymbol{\lambda}^T \left(\boldsymbol{J} \dot{\boldsymbol{q}} - \dot{\boldsymbol{x}} \right), \tag{1.4}$$

and impose the necessary (here also sufficient) conditions for a constrained minimum of G_l:

$$\nabla_{\dot{\boldsymbol{q}}} L = \left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}} \right)^T = \boldsymbol{W} \dot{\boldsymbol{q}} + \boldsymbol{J}^T \boldsymbol{\lambda} = \boldsymbol{0}, \tag{1.5}$$

$$\nabla_{\boldsymbol{\lambda}} L = \left(\frac{\partial L}{\partial \boldsymbol{\lambda}} \right)^T = \boldsymbol{J} \dot{\boldsymbol{q}} - \dot{\boldsymbol{x}} = \boldsymbol{0}, \tag{1.6}$$

$$\nabla^2_{\dot{\boldsymbol{q}}} L = \boldsymbol{W} > 0. \tag{1.7}$$

Solving for $\dot{\boldsymbol{q}}$ from (1.5)

$$\dot{\boldsymbol{q}} = -\boldsymbol{W}^{-1} \boldsymbol{J}^T \boldsymbol{\lambda}, \tag{1.8}$$

and replacing in (1.6) yields the multiplier

$$\boldsymbol{\lambda} = -\left(\boldsymbol{J} \boldsymbol{W}^{-1} \boldsymbol{J}^T \right)^{-1} \dot{\boldsymbol{x}}. \tag{1.9}$$

Replacing (1.9) in (1.8)

$$\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1} \boldsymbol{J}^T \left(\boldsymbol{J} \boldsymbol{W}^{-1} \boldsymbol{J}^T \right)^{-1} \dot{\boldsymbol{x}}, \tag{1.10}$$

which can be rewritten as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}_{\boldsymbol{W}} \dot{\boldsymbol{x}}, \tag{1.11}$$

with the weighted pseudoinverse

$$\boldsymbol{J}^{\#}_{\boldsymbol{W}} = \boldsymbol{W}^{-1} \boldsymbol{J}^T \left(\boldsymbol{J} \boldsymbol{W}^{-1} \boldsymbol{J}^T \right)^{-1}. \tag{1.12}$$

If the weighting matrix $\boldsymbol{W}$ in (1.3) is identity (i.e. $\boldsymbol{W} = \boldsymbol{I}$), then the solution (1.11) is the generalized inverse method which minimizes locally the joint velocity norm.

Another way to optimize the robot kinematic redundancy locally is by adding a term to the (1.11) solution so as not to affect execution of the task trajectory, i.e., belonging to the null-space of the task Jacobian as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{W}}^{\#}\dot{\boldsymbol{x}} + (\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{W}}^{\#}\boldsymbol{J})\dot{\boldsymbol{q}}_0, \tag{1.13}$$

where $(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{W}}^{\#}\boldsymbol{J})$ is the orthogonal projection matrix in the null space of $\boldsymbol{J}$ Chiaverini et al. [2008]. Note that the solution (1.13) is exactly the same one resulted from solving the problem

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}} \; & G_l(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\,(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0)^T \boldsymbol{W}(\boldsymbol{q})(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0) \\ & \text{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}. \end{aligned} \tag{1.14}$$

By specifying the vector $\dot{\boldsymbol{q}}_0$, the robot redundancy can be exploited to optimize a less priority desired criteria without any influence for the higher robot priority task specified by (1.11). This is usually done by using the gradient of a configuration-dependent objective function

$$\dot{\boldsymbol{q}}_0 = \nabla_{\boldsymbol{q}} k(\boldsymbol{q}). \tag{1.15}$$

The typical objective functions $k(\boldsymbol{q})$ are the manipulability, for maximizing the distance from singularities, and the joint range to minimize the distance from the middle points of joint limits Siciliano et al. [2008].

For redundant robots with several degrees of freedom, the previous solutions are computationally intensive since the pseudoinverse is needed. Instead, when a Jacobian has full rank, a more efficient approach based on joint space decomposition can be used (De Luca and Oriolo [1990], Ma and Watanabe [2004]). The robot joints are divided into two sets. The non-redundant of the same dimension as the task space and the remaining redundant joints. The non-redundant joint positions are found using analytic inverse kinematics for the end-effector, while the $n - m$ redundant joints can be selected independently for optimizing the desired objective function. In this case, the projection of gradient (1.15) will be reduced to be w.r.t the non-redundant joints only as

$$\dot{\boldsymbol{q}} = \begin{pmatrix} \dot{\boldsymbol{q}}_{nr} \\ \dot{\boldsymbol{q}}_r \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_{nr}^{-1} \\ \boldsymbol{0} \end{pmatrix} \dot{\boldsymbol{x}} + \begin{pmatrix} -\boldsymbol{J}_{nr}^{-1}\boldsymbol{J}_r \\ \boldsymbol{I} \end{pmatrix} \begin{pmatrix} -(\boldsymbol{J}_{nr}^{-1}\boldsymbol{J}_r)^T & \boldsymbol{I} \end{pmatrix} \nabla_{\boldsymbol{q}} k(\boldsymbol{q}). \tag{1.16}$$

The so called reduced gradient method in (1.16) is analytically simpler and numerically faster than the full projected gradient (1.13), but still requires a strategy to specify which joints to be considered redundant.

Global optimization

The performance index of the instantaneous minimization in (1.3) and (1.13) can be redefined for global optimization as

$$\begin{aligned} \int_{t_0}^{t_f} G_g(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\; dt &= \frac{1}{2}\int_{t_0}^{t_f} \dot{\boldsymbol{q}}^T \boldsymbol{W}(\boldsymbol{q})\dot{\boldsymbol{q}} + k(\boldsymbol{q})\; dt \\ & \text{s.t. } \boldsymbol{x}(t) - \boldsymbol{f}(\boldsymbol{q}(t)) = \boldsymbol{0}, \end{aligned} \tag{1.17}$$

and solved by the calculus of variation (Martin et al. [1989], Kim et al. [1994]). In order to specify a unique optimal solution, both the necessary and boundary

conditions should be considered. In this case the Lagrangian function is defined as

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\lambda}, t) = G_g(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) + \boldsymbol{\lambda}^T(\boldsymbol{x}(t) - \boldsymbol{f}(\boldsymbol{q}(t))), \tag{1.18}$$

and the necessary conditions for optimality are given by the Euler-Lagrange equations

$$\frac{\partial L}{\partial \boldsymbol{q}} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} = 0. \tag{1.19}$$

At the end, the inverse kinematic resolution is given at the acceleration level as

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{W}}^{\#}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{W}}^{\#}\boldsymbol{J})\ddot{\boldsymbol{q}}_0, \tag{1.20}$$

where

$$\ddot{\boldsymbol{q}}_0 = \boldsymbol{W}^{-1}\Big(- \dot{\boldsymbol{W}} + \frac{1}{2}\frac{\partial(\dot{\boldsymbol{q}}^T\boldsymbol{W})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial k}{\partial \boldsymbol{q}}\Big). \tag{1.21}$$

For the simple case $\boldsymbol{W} = \boldsymbol{I}$ and $k(\boldsymbol{q}) = 0$ in the problem (1.17), the solution (1.20) is given by

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}), \tag{1.22}$$

which is the generalized inverse method at the acceleration level that minimizes globally the joint velocity norm. In this case, if the initial and final configurations are not prespecified, there are only kinematic constraints and the natural boundary conditions to be satisfied are split and expressed as

$$\dot{\boldsymbol{q}}|_{t_0,t_f} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}}|_{t_0,t_f}, \tag{1.23}$$

which requires a numerical solution of a two-point boundary value problem (TPBVP). If the joint configuration and/or joint velocity are constrained at the beginning and/or at the terminal points, the robot motion is controlled by (1.22) at that points and the optimal solution is considered weak Kazerounian and Wang [1988].

Another way to optimize the robot redundancy globally is to resort to the Pontryagin's maximum principle Nakamura and Hanafusa [1987]. Considering $\boldsymbol{W} = \boldsymbol{I}$, the first differential inverse kinematic (1.13) can be regarded as the following dynamic system

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} + (\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})\dot{\boldsymbol{q}}_0 \equiv g(\boldsymbol{q}, \boldsymbol{u}, t), \tag{1.24}$$

considering $\boldsymbol{q}$ as a state vector and $\dot{\boldsymbol{q}}_0$ as an input vector $\boldsymbol{u}$. According to the Pontryagin's principle, to minimize the cost function (1.17), the Hamiltonian can be defined as

$$\begin{aligned} H(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\lambda}, t) &= G_g(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) + \boldsymbol{\lambda}^T g(\boldsymbol{q}, \boldsymbol{u}, t) \\ &= \tfrac{1}{2}(g + \boldsymbol{\lambda})^T(g + \boldsymbol{\lambda}) - \tfrac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{\lambda} + k(\boldsymbol{q}), \end{aligned} \tag{1.25}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ is a costate vector Kim et al. [1994]. The input $\boldsymbol{u}$ which minimizes the Hamiltonian (1.25) can be written as

$$\boldsymbol{u} = -(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})\boldsymbol{\lambda}. \tag{1.26}$$

The optimal trajectory $\boldsymbol{q}(t)$ is yielded by solving the following $2n$ differential equations

$$\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} = g(\boldsymbol{q}, \boldsymbol{u}, t), \tag{1.27}$$

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{q}} = -\big(\frac{\partial g}{\partial \boldsymbol{q}}\big)^T(g + \boldsymbol{\lambda}) - \frac{\partial k}{\partial \boldsymbol{q}}. \tag{1.28}$$

For the terminal points, also the TPBVP appears which can be solved using the initial value adjusting method Nakamura and Hanafusa [1987].

Local vs. global optimizationn

The main advantage of the local redundancy resolution is the simple formulation which requires a relatively small amount of computation. This makes it a suitable option for real-time control applications. On the other hand, since the local optimization methods instantaneously minimize a desired criterion, they do not guarantee a global minimum and can lead to unsatisfactory performance over long tasks Hollerbach and Suh [1987]. Furthermore, the solution (1.11) leads to non-repeatable motion in the joint space and cannot guarantee global singularity avoidance Chiaverini et al. [2008].

The complex formulation, and the computationally intensive numerical procedure required for the actual minimization of (1.17) make it impractical for real-time kinematic control. However, for off-line industrial applications, this may be acceptable.

1.3 Second-order methods

Working at the acceleration level admits considering the dynamic performance during the manipulator motion. Moreover, the computed joint accelerations together with the corresponding positions and velocities can be used as reference signals of a task space dynamic controller. On the other hand, a second-order redundancy resolution scheme in general demands more computational load Chiaverini et al. [2008].

The previously introduced optimization techniques can be used also at the acceleration level. By differentiating (1.2), the second-order relation can be obtained as

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{1.29}$$

where the vector $\boldsymbol{h}$ has a quadratic dependence on $\dot{\boldsymbol{q}}$. As in (1.13), the least-squares solution for the weighted joint accelerations norm is given by

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{W}}^{\#}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{W}}^{\#}\boldsymbol{J})\ddot{\boldsymbol{q}}_0. \tag{1.30}$$

Note that, the equation (1.30) coming from the local optimization at the second-order is identical to the (1.20) coming from the global optimization approach. In this case, the arbitrary vector $\ddot{\boldsymbol{q}}_0$ can be specified for stabilizing and smoothing joint trajectories as

$$\ddot{\boldsymbol{q}}_0 = -\boldsymbol{D}\dot{\boldsymbol{q}}, \tag{1.31}$$

where $\boldsymbol{D}$ is a $n \times n$ diagonal, positive semi-definite matrix Flacco and De Luca [2015].

Let the dynamic model of a rigid robot with n degrees of freedom described as Siciliano et al. [2008],

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{1.32}$$

with symmetric, positive definite inertia matrix $\boldsymbol{M}$, Coriolis and centrifugal terms $\boldsymbol{c}$, gravity term $\boldsymbol{g}$, and commanded joint torque $\boldsymbol{\tau}$. Vector $\boldsymbol{c}$ has a quadratic dependence on $\dot{\boldsymbol{q}}$. In particular, its components $c_i(\boldsymbol{q}, \dot{\boldsymbol{q}})$, for $i = 1, \ldots, n$, take the following expression

$$c_i = \dot{\boldsymbol{q}}^T \boldsymbol{C}_i(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad \boldsymbol{C}_i = \frac{1}{2}\left(\left(\frac{\partial \boldsymbol{m}_i}{\partial \boldsymbol{q}}\right) + \left(\frac{\partial \boldsymbol{m}_i}{\partial \boldsymbol{q}}\right)^T - \frac{\partial \boldsymbol{M}}{\partial q_i}\right), \tag{1.33}$$

where $\boldsymbol{m}_i(\boldsymbol{q})$ is the ith column of $\boldsymbol{M}$ and the elements C_{ijk} of matrix $\boldsymbol{C}_i$ are the so-called Christoffel symbols. Another convenient factorization of $\boldsymbol{c}$ is given by

$$\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}, \tag{1.34}$$

where matrix $\boldsymbol{S}$ can be chosen in different ways, but we will prefer one that induces the skew-symmetric property to $\dot{\boldsymbol{M}} - 2\boldsymbol{S}$. It can be shown that this is equivalent to the property

$$\dot{\boldsymbol{M}}(\boldsymbol{q}) = \boldsymbol{S}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{S}^T(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{1.35}$$

From (1.33), one such matrix is given by

$$\boldsymbol{S}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \operatorname{col}\left\{\dot{\boldsymbol{q}}^T \boldsymbol{C}_i(\boldsymbol{q})\right\}.$$

In the following, we will also use the compact notation

$$\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) \tag{1.36}$$

to denote all terms in (1.32) not depending on joint acceleration. The redundancy of the robot can be resolved by minimizing locally some (possibly, weighted) norm of the joint torques. A standard linear-quadratic optimization problem can be defined as

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}}} G_l(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) &= \frac{1}{2}\|\boldsymbol{\tau}\|^2 \\ \text{s.t. } \boldsymbol{\tau} &= \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{n} \\ \ddot{\boldsymbol{x}} &= \boldsymbol{J}\ddot{\boldsymbol{q}} + \boldsymbol{h}. \end{aligned} \tag{1.37}$$

Similar to (1.3), the solution to the (1.37) problem is obtained with the method of Lagrange multipliers and can be expressed in the (1.30) form using the weighting matrix $\boldsymbol{W} = \boldsymbol{M}^{-2}$ and the joint acceleration $\ddot{\boldsymbol{q}}_0 = \boldsymbol{M}^{-1}\boldsymbol{n}$ in the null space Hollerbach and Suh [1987]. For longer movements, the previous local optimization solution exhibits an unstable behavior in the form of motion oscillations, growing joint velocities, and sudden whipping torque effects with a practical loss of control. Global optimization techniques can be used to pursue a global torque optimization over the whole motion trajectory (Suh and Hollerbach [1987], Nakamura and Hanafusa [1987]), but again this will imply the off-line numerical solution of a TPBVP and the availability in advance of the entire desired task. Another way to solve the problem, using the so-called model based preview approach by Al Khudir et al. [2019], will be introduced in Chapter 3.

2 Time-Optimal Control

2.1 Introduction

In this chapter, we address the time-optimal trajectory planning along a Cartesian path for a kinematically redundant robot with a two-step procedure. We propose to generate a sequence of joint configurations by means of a second-order differential inverse kinematics scheme, using weighted pseudoinversion, optimizing locally an inertia-related criterion, and including judiciously a damping action in the null space of the task. The obtained configurations are then interpolated with a parameterized path in the joint space, and an exact minimum time solution is computed using the TOPP algorithm. In case the initial robot configuration is not assigned a priori, we include also a kinematic optimization scheme to find the best initial joint configuration corresponding to the starting point of the Cartesian path.

The chapter is organized as follows. The formulation of the time-optimal planning problem on a parametrized joint path and its basic solution algorithm are reviewed in Sec. 2.2. Section 2.3 presents the core of the method, moving ideas that exploit robot redundancy from the first- to the second-order differential level. We provide also a procedure in order to evaluate the distance between solutions obtained with a two-step approach and the global time-optimal motion computed along the same Cartesian path, using an alternative nonlinear programming method. Section 2.4 reports comparative results on different robot systems. Numerical simulations are done using a 3R planar arm and the Universal Robots UR10. Experimental results are reported for a KUKA LWR IV robot executing positional tasks.

The problem described in this Chapter has been addressed in the work by Al Khudir and De Luca [2018].

2.2 Time-optimal planning on a geometric path

We briefly review the basic formulation of the time-optimal planning on a parametrized joint path as in (Bobrow et al. [1985], Pham [2014]).

Consider the robot dynamic model described by (1.32) and (1.34). Assume that robot motion in the joint space is constrained to a given path that is continuously parametrized by a scalar s as a (non-decreasing) function of time t, or

$$\boldsymbol{q} = \boldsymbol{q}(s), \quad s \in [0, s_f], \qquad s = s(t), \quad t \in [0, t_f]. \tag{2.1}$$

Differentiating (2.1) once and twice with respect to time yields

$$\dot{\boldsymbol{q}} = \boldsymbol{q}' \dot{s}, \qquad \ddot{\boldsymbol{q}} = \boldsymbol{q}' \ddot{s} + \boldsymbol{q}'' \dot{s}^2, \tag{2.2}$$

where a dot $\dot{(\,)}$ and a prime $(\,)'$ denote differentiation with respect to time t and to parameter s. The robot is subject to bounds on the joint torques

$$\boldsymbol{\tau}^{\min} \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}^{\max}, \qquad \forall t \in [0, t_f], \tag{2.3}$$

where $\boldsymbol{\tau}^{\max}$ and $\boldsymbol{\tau}^{\min}$ (usually, equal to $-\boldsymbol{\tau}^{\max}$) are constant vectors, and inequalities are to be intended component-wise. Substituting (2.1) and (2.2) into (1.32) and (2.3), and rearranging the terms, leads to

$$\boldsymbol{\tau}^{\min} \leq \boldsymbol{a}(s)\ddot{s} + \boldsymbol{b}(s)\dot{s}^2 + \boldsymbol{g}(s) \leq \boldsymbol{\tau}^{\max} \tag{2.4}$$

where $\boldsymbol{a} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'$, $\boldsymbol{b} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'' + \boldsymbol{S}(\boldsymbol{q}, \boldsymbol{q}')\boldsymbol{q}'$, and $\boldsymbol{g}$ is the gravity torque vector (all arguments are evaluated using (2.1) and (2.2)). As a result, a trajectory $\boldsymbol{q}(s(t))$ will be feasible if and only if the following bounds on the pseudo-acceleration $\ddot{s}$ are satisfied along the whole path

$$\alpha(s(t), \dot{s}(t)) \leq \ddot{s}(t) \leq \beta(s(t), \dot{s}(t)), \quad \forall s \in [0, s_f]. \tag{2.5}$$

For each $(s, \dot{s})$, the upper and lower acceleration bounds in (2.5) are defined as

$$\alpha(s, \dot{s}) = \max_i \alpha_i(s, \dot{s}) \quad \text{and} \quad \beta(s, \dot{s}) = \min_i \beta_i(s, \dot{s}). \tag{2.6}$$

The expressions of α_i and β_i depend on the sign of $a_i(s)$. In particular, for $i = 1, \ldots, n$:

- if $a_i(s) > 0$, then $\begin{cases} \alpha_i = \frac{\tau_i^{\min} - g_i(s) - b_i(s)\dot{s}^2}{a_i(s)}, \\ \beta_i = \frac{\tau_i^{\max} - g_i(s) - b_i(s)\dot{s}^2}{a_i(s)}; \end{cases}$
- if $a_i(s) < 0$, then $\begin{cases} \alpha_i = \frac{-\tau_i^{\max} + g_i(s) + b_i(s)\dot{s}^2}{|a_i(s)|}, \\ \beta_i = \frac{-\tau_i^{\min} + g_i(s) + b_i(s)\dot{s}^2}{|a_i(s)|}; \end{cases}$
- if $a_i(s) = 0$, then s is a zero-inertia point.

The last case is a dynamic singularity that should be handled separately (Slotine and Yang [1989], Shiller [1994]).

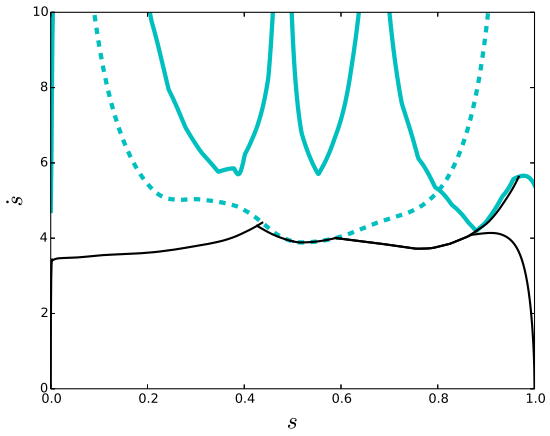


Figure 2.1: Time-optimal trajectory in the phase plane. The cyan lines are the maximum velocity curves MVC_t (solid) and MVC_v (dotted), while the black line is the time-optimal profile.

From (2.5), a maximum velocity curve $\text{MVC}_t(s)$ is imposed in the $(s, \dot{s})$ plane, defined by

$$\text{MVC}_t(s) = \begin{cases} \min\{\dot{s} \geq 0 : \alpha(s, \dot{s}) = \beta(s, \dot{s})\}, & \text{if } \alpha(s, 0) \leq \beta(s, 0), \\ 0, & \text{if } \alpha(s, 0) > \beta(s, 0). \end{cases}$$

Kinematic constraints (e.g., joint velocity limits) can also be considered Zlajpah [1996]. Assuming symmetric velocity bounds, we have for the velocity of joint i, $i = 1, \ldots, n$:

$$-\dot{q}_i^{\max} \leq \dot{q}_i \leq \dot{q}_i^{\max} \quad \Rightarrow \quad \dot{s}_i^{\max}(s) = \frac{\dot{q}_i^{\max}}{|q_i'(s)|}. \tag{2.7}$$

The overall bound on $\dot{s}$ will be

$$\dot{s}_{\max}(s) = \min_i \dot{s}_i^{\max}(s), \quad \forall s \in [0, s_f]\,. \tag{2.8}$$

Equation (2.8) induces another maximum velocity curve, denoted $\text{MVC}_v(s)$. As a result, every feasible timing law $s = s(t)$ must remain below the curve in the phase plane $(s, \dot{s})$ defined by $\text{MVC} = \min\{\text{MVC}_t, \text{MVC}_v\}$ as shown in Fig. 2.1.

Based on Pontryagin Maximum Principle, the optimal trajectory in the $(s, \dot{s})$ plane that minimizes the rest-to-rest motion time is given by a control law of the bang-bang type. The pseudo-acceleration $\ddot{s}$ follows alternatively α or β, while the profile of $\dot{s}$ should always stay below the maximum velocity curve MVC. From (2.5) and (2.8), it can be shown that at least one joint is saturated at any time either to its torque bound or to its velocity bound. The optimal timing law $s^*(t)$ and the associated minimum time t_f^* can be found using TOPP, a complete and robust algorithm presented in Pham [2014]. From this, having also $(\dot{s}^*(t), \ddot{s}^*(t))$, the time profiles $\dot{\boldsymbol{q}}(t)$ and $\ddot{\boldsymbol{q}}(t)$ are evaluated using (2.2) and the needed joint torque is computed algebraically using (1.32).

2.3 Exploiting robot redundancy

First-order schemes

Let a parametrized path in the m-dimensional Cartesian (or task) space be assigned as

$$\boldsymbol{x} = \boldsymbol{x}(s), \qquad s \in [0, s_f], \tag{2.9}$$

If a $\boldsymbol{q}_0 \in \mathbb{R}^n$ is assigned as initial configuration ($\boldsymbol{q}(0) = \boldsymbol{q}_0$), it should satisfy $\boldsymbol{f}(\boldsymbol{q}_0) = \boldsymbol{x}(0) = \boldsymbol{x}_0$. Dropping dependencies, the first-order differential kinematics in the parameter space s is

$$\boldsymbol{x}' = \boldsymbol{J}\boldsymbol{q}'. \tag{2.10}$$

The different inverse kinematic schemes introduced in Chapter 1 can be used also in the s space. The simplest first-order differential inverse kinematics is given by

$$\boldsymbol{q}' = \boldsymbol{J}^{\#}\boldsymbol{x}'. \tag{2.11}$$

A parameterized joint path $\boldsymbol{q} = \boldsymbol{q}(s)$ can be generated by numerical integration of (2.11), starting from $\boldsymbol{q}_0$ and evaluated over discrete samples of the parameter s, followed by interpolation of the obtained data points in the joint space with a class of smooth functions (e.g., cubic splines). In the first order schemes, the pseudo joint acceleration $\boldsymbol{q}''$ (needed to evaluate the robot dynamics in (2.4) and the final joint acceleration in (2.2)) is obtained by a numerical derivation of (2.11). At this stage, the second step can be processed by applying TOPP along the generated parametrized joint path under the robot kinematic and dynamic constraints.

Projected gradient at the velocity level (PGV)

The scheme (1.13) can be defined in the s space as

$$\boldsymbol{q}' = \boldsymbol{J}^{\#}\boldsymbol{x}' - \delta(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})\nabla_{\boldsymbol{q}}k, \tag{2.12}$$

where $\delta \geq 0$ is a suitable scalar gain. Assume that $\dot{\boldsymbol{q}} = \boldsymbol{0}$, so that velocity-dependent terms vanish in (1.29), and that gravitational terms are neglected in (1.32). In Chiacchio [1990], the author has considered the dynamic manipulability ellipsoid in the Cartesian space introduced by Yoshikawa [1985]

$$\ddot{\boldsymbol{x}}^T(\boldsymbol{M}\boldsymbol{J}^{\#})^T\boldsymbol{M}\boldsymbol{J}^{\#}\ddot{\boldsymbol{x}} \leq 1, \tag{2.13}$$

where $\ddot{\boldsymbol{x}}$ is the set of all Cartesian accelerations that realizable by a joint torque $||\boldsymbol{\tau}|| \leq 1$. In order to improve the acceleration/deceleration capabilities of the robot end-effector along the specified Cartesian path, the following objective function can be used in (2.12)

$$k(\boldsymbol{q}) = \boldsymbol{t}^T(\boldsymbol{M}\boldsymbol{J}^{\#})^T\boldsymbol{M}\boldsymbol{J}^{\#}\boldsymbol{t}, \tag{2.14}$$

with $\boldsymbol{t} \in \mathbb{R}^m$ being a unit vector along the tangent direction to the path. This will help in locally generating a constrained joint path along which the robot will expose a reduced inertial load in the task space (minimum distance to the surface of the dynamic manipulability ellipsoid).

Weighted pseudoinverse at the velocity level (WPV)

As an alternative to adopting a null-space projection scheme, the weighted pseudoinverses at the velocity level similar to (1.11) scheme can be used as

$$\boldsymbol{q}' = \boldsymbol{J}_{\boldsymbol{Q}}^{\#}\boldsymbol{x}'. \tag{2.15}$$

Considering symmetric bounds on the joint torques, the use of a symmetric weight matrix

$$\boldsymbol{Q} = (\boldsymbol{L}^{-1}\boldsymbol{M})^T\boldsymbol{L}^{-1}\boldsymbol{M} > 0, \tag{2.16}$$

with a diagonal scaling matrix $\boldsymbol{L} = \text{diag}\{\tau_1^{\max}, \ldots, \tau_n^{\max}\}$, was proposed in Basile and Chiacchio [2003]. Following the pseudo-velocity (2.15) will limit the motion of those joints that have larger inertia-to-maximum torque ratios. A scalar parameter $\gamma \geq 0$ can be added to the weighting matrix $\boldsymbol{Q}$ as

$$\boldsymbol{Q}_\gamma = \exp(\gamma \ln \boldsymbol{Q}), \tag{2.17}$$

where exp(.) and ln(.) compute the matrix exponential and the principal matrix logarithm, respectively. For $\gamma = 0$, the simple psoudinverse solution (2.11) is used, while for $\gamma = 1$ the weighted pseudoinverse (2.15) is obtained.

Second-order schemes

Instead of using first-order differential inverse kinematics solutions as in (2.12) and (2.15), in the first step of our minimum-time planning problem we propose to exploit redundancy at the second-order (pseudo-acceleration) level in the similar way introduced in (1.30) in the time domain. Differentiating (2.10) w.r.t. the parameter s gives

$$\boldsymbol{x}'' = \boldsymbol{J}\boldsymbol{q}'' + \boldsymbol{J}'\boldsymbol{q}', \qquad \boldsymbol{J}' = \frac{d\boldsymbol{J}}{ds}. \tag{2.18}$$

Using the weighting matrix in (2.17), the second-order differential inverse kinematics can be written as a weighted pseudoinversion

$$\boldsymbol{q}'' = \boldsymbol{J}_{\boldsymbol{Q}_\gamma}^{\#}(\boldsymbol{x}'' - \boldsymbol{J}'\boldsymbol{q}'), \tag{2.19}$$

where $\boldsymbol{J}_{\boldsymbol{Q}_\gamma}^{\#} = \boldsymbol{Q}_\gamma^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{Q}_\gamma^{-1}\boldsymbol{J}^T)^{-1}$. In addition, we can use also a null-space term in (2.19) and obtain the general expression

$$\boldsymbol{q}'' = \boldsymbol{J}_{\boldsymbol{Q}_\gamma}^{\#}\left(\boldsymbol{x}'' - \boldsymbol{J}'\boldsymbol{q}'\right) + \left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{Q}_\gamma}^{\#}\boldsymbol{J}\right)\boldsymbol{q}_0'', \tag{2.20}$$

where $\boldsymbol{q}_0'' \in \mathbb{R}^n$ is a preferred pseudo-acceleration vector in the joint space. We will label this solution as *ACC*.

To determine the preferred pseudo-acceleration $\boldsymbol{q}_0''$, similar techniques as those introduced in Chiacchio [1990] and Chiacchio and Concilio [1998] will be used. For the general case of different bounds on the joint torque components, it is useful to use a normalization with respect to the nominal joint torque limits. Assume that $\dot{\boldsymbol{q}} = \boldsymbol{0}$, so that velocity-dependent terms vanish, and that gravitational terms are neglected in (1.32). Using $\boldsymbol{Q}_\gamma$ as weighting matrix in the

pseudoinverse, the normalized joint torques $\tilde{\boldsymbol{\tau}}$, w.r.t nominal torque bounds, in the time domain can be written as

$$\tilde{\boldsymbol{\tau}} = \boldsymbol{L}^{-1}\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{L}^{-1}\boldsymbol{M}\boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\ddot{\boldsymbol{x}}. \tag{2.21}$$

Consequently, the associated dynamic manipulability ellipsoid in the Cartesian space will be

$$\ddot{\boldsymbol{x}}^T {\boldsymbol{J}^{\#}}^T_{\boldsymbol{Q}_\gamma} \boldsymbol{Q}\, \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\ddot{\boldsymbol{x}} \leq 1. \tag{2.22}$$

To improve the acceleration/deceleration capabilities of the robot end effector along the Cartesian path, it is useful to minimize the quantity

$$k_1(\boldsymbol{q}) = \boldsymbol{t}^T {\boldsymbol{J}^{\#}}^T_{\boldsymbol{Q}_\gamma} \boldsymbol{Q}\, \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\boldsymbol{t}, \tag{2.23}$$

with $\boldsymbol{t} \in \mathbb{R}^m$ defined as in (2.14). The preferred vector $\boldsymbol{q}_0''$ in (2.20) is then chosen as

$$\boldsymbol{q}_0'' = -\delta_1 \nabla_{\boldsymbol{q}} k_1 - \boldsymbol{D}\boldsymbol{q}', \tag{2.24}$$

where $\delta_1 \geq 0$ is a scalar gain and $\boldsymbol{D}$ is a $n \times n$ diagonal, positive semi-definite matrix. The second term in (2.24) is a damping term on the pseudo-velocity, which guarantees that bounded displacements are generated in the joint space. This property is similar to the null-space damping (1.31) in the time domain.

In the present framework, the choice of both δ_1 and $\boldsymbol{D}$ turns out to be critical in determining the total length of the generated joint path, and thus indirectly also the achievable minimum time associated to the path. Intuitively, a too small damping matrix $\boldsymbol{D}$ (or no damping at all) will lead to a potential drift or wandering of the joint path associated to the original Cartesian path. Conversely, if the damping action is too strong, joint reconfigurations intended to optimize the objective function (2.23) will be penalized. Similarly, the choice of δ_1 should balance the length of the path generated in the joint space vs. the efficacy in the auxiliary optimization of $k_1(\boldsymbol{q})$. For handling this trade off, the following bounds are imposed to δ_1:

$$0 \leq \delta_1 \leq \min\left\{ \delta_2 \frac{\|\boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}(\boldsymbol{x}'' - \boldsymbol{J}'\boldsymbol{q}')\|}{\|(\boldsymbol{I} - \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\boldsymbol{J})\nabla_{\boldsymbol{q}} k_1\|}, \delta_{1,\max} \right\}, \tag{2.25}$$

with $\delta_2 \in [0, 1]$.

In order to avoid numerical drifts during calculations, a stabilizing PD term on the (spacial) task error can be added to (2.20). Using the definition of the path-tracking error $\boldsymbol{e}$ and its derivatives $\boldsymbol{e}'$ and $\boldsymbol{e}''$ w.r.t. the parameter s, we have

$$\begin{aligned} \boldsymbol{e}(s) &= \boldsymbol{x}(s) - \boldsymbol{f}(\boldsymbol{q}(s)), \\ \boldsymbol{e}'(s) &= \boldsymbol{x}'(s) - \boldsymbol{J}(\boldsymbol{q}(s))\boldsymbol{q}'(s), \\ \boldsymbol{e}''(s) &= \boldsymbol{x}''(s) - \boldsymbol{J}'(\boldsymbol{q}(s))\boldsymbol{q}'(s) - \boldsymbol{J}(\boldsymbol{q}(s))\boldsymbol{q}''(s), \end{aligned} \tag{2.26}$$

where $\boldsymbol{J}' = \sum_1^n (\partial \boldsymbol{J}/\partial q_i)\, q_i'$. From Siciliano et al. [2008], an asymptotically stable behavior of the dynamics of $\boldsymbol{e}$ is ensured if

$$\begin{aligned} & \boldsymbol{e}'' + \boldsymbol{K}_d\boldsymbol{e}' + \boldsymbol{K}_p\boldsymbol{e} = 0, \\ \text{or} \qquad & (\boldsymbol{x}''(s) - \boldsymbol{J}'(\boldsymbol{q}(s))\boldsymbol{q}'(s) - \boldsymbol{J}(\boldsymbol{q}(s))\boldsymbol{q}''(s)) + \boldsymbol{K}_d\boldsymbol{e}' + \boldsymbol{K}_p\boldsymbol{e} = 0, \end{aligned} \tag{2.27}$$

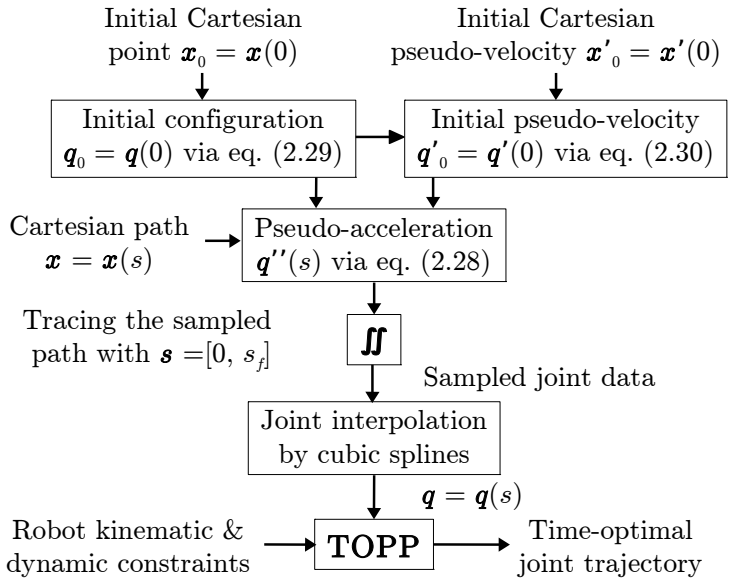


Figure 2.2: The overall scheme for the proposed *ACC* solution.

where $\boldsymbol{K}_p > 0$ and $\boldsymbol{K}_d > 0$ are diagonal gain matrices. The error dynamics (2.27) can be rearranged and a null-space term can be added to obtain the vector of joint accelerations as

$$\boldsymbol{q}'' = \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\left(\boldsymbol{x}'' - \boldsymbol{J}'\boldsymbol{q}' + \boldsymbol{K}_d\boldsymbol{e}' + \boldsymbol{K}_p\boldsymbol{e}\right) + \left(\boldsymbol{I} - \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\boldsymbol{J}\right)\boldsymbol{q}''_0, \tag{2.28}$$

whereas the null-space term will not affect the Cartesian error dynamics since $\boldsymbol{J}\left(\boldsymbol{I} - \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\boldsymbol{J}\right) = \boldsymbol{0}$. Therefore, perfect tracking of the trajectories defined in the workspace is assured.

With the second-order *ACC* method, a parameterized path $\boldsymbol{q} = \boldsymbol{q}(s)$ in the joint space will be generated by double numerical integration of (2.28), used together with (2.24) and (2.23). In the second-order schemes, since the pseudo joint acceleration is computed directly, there is no need for a numerical derivation as in the first-order schemes. The second step of the minimum-time planning procedure is identical to that of first-order schemes, e.g., *PGV* or *WPV*.

Finding an initial configuration

To start a first- or a second-order redundancy resolution scheme, either a consistent initial configuration $\boldsymbol{q}(0) = \boldsymbol{q}_0$ is assigned, or it should be determined so as to match the end-effector path at the start, i.e., $\boldsymbol{f}(\boldsymbol{q}(0)) = \boldsymbol{x}(0) = \boldsymbol{x}_0$. Indeed, being the robot redundant, there is an infinite number of such initial robot configurations. To find the most efficient $\boldsymbol{q}_0$ for our motion task, a preliminary kinematic control scheme is used in the *time domain*, similar to (2.28) in the *space domain*,

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}(-\dot{\boldsymbol{J}}\dot{\boldsymbol{q}} - \boldsymbol{K}_d\dot{\boldsymbol{x}} + \boldsymbol{K}_p\boldsymbol{e}_0) + (\boldsymbol{I} - \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}\boldsymbol{J})\ddot{\boldsymbol{q}}_0, \tag{2.29}$$

with $\ddot{\boldsymbol{q}}_0 = -\delta_1 \nabla_{\boldsymbol{q}} k_1 - \boldsymbol{D}\dot{\boldsymbol{q}}$, $k_1(\boldsymbol{q})$ computed as in (2.23), $\boldsymbol{e}_0 = \boldsymbol{x}_0 - \boldsymbol{f}(\boldsymbol{q})$, and $\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}$. In this preliminary phase, the robot may start from any configuration, but typically one still corresponding to the initial point of the Cartesian path (i.e., with $\|\boldsymbol{e}_0\| = 0$). Equation (2.29) is then integrated forward until $\|\dot{\boldsymbol{q}}\| < \epsilon$, a specific threshold set, e.g., to 10^{-3}. The robot joints will move mainly according to the null-space projection term in (2.29), while the first term is used to keep the robot end-effector in the initial Cartesian position.

For the second-order scheme (2.28), we need in addition a suitable initial pseudo-velocity $\boldsymbol{q}'(0) = \boldsymbol{q}'_0$. Using the available $\boldsymbol{q}_0$, this can be computed as

$$\boldsymbol{q}'(0) = \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}(\boldsymbol{q}_0)\boldsymbol{x}'(0) - \delta \left(\boldsymbol{I} - \boldsymbol{J}^{\#}_{\boldsymbol{Q}_\gamma}(\boldsymbol{q}_0)\boldsymbol{J}(\boldsymbol{q}_0) \right) \nabla_{\boldsymbol{q}} k_1(\boldsymbol{q}_0). \tag{2.30}$$

The overall computational scheme for our proposed method is shown in Fig. 2.2.

Comparison with a global time-optimal solution

In order to compare the two-step optimization methods with a global time-optimal solution on a Cartesian path, we proposed the procedure in Fig. 2.3. Defining the state and control vectors as $\boldsymbol{\chi}(t) = (\boldsymbol{q} \quad \dot{\boldsymbol{q}})^T$ and $\boldsymbol{u}(t) = \boldsymbol{\tau}$ respectively, a point-to-point (PTP) minimum time problem can be formulated as

$$\begin{aligned}
&\min_{\boldsymbol{u}} L(\boldsymbol{\chi}(t), \boldsymbol{u}(t)) = \int_0^{t_f} 1 \; dt = t_f \\
&\text{s.t. } \boldsymbol{\tau} = \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{n}, \\
&\left.\begin{array}{l} \boldsymbol{q}_{min} \le \boldsymbol{q} \le \boldsymbol{q}_{max}, \\ \dot{\boldsymbol{q}}_{min} \le \dot{\boldsymbol{q}} \le \dot{\boldsymbol{q}}_{max}, \\ \boldsymbol{\tau}_{min} \le \boldsymbol{\tau} \le \boldsymbol{\tau}_{max}, \end{array}\right\} \quad \text{(state and control input constraints)} \\
&\left.\begin{array}{l} \boldsymbol{q}_0 = \boldsymbol{q}(t_0), \; \dot{\boldsymbol{q}}_0 = \dot{\boldsymbol{q}}(t_0), \\ \boldsymbol{C}(t_f, \boldsymbol{\chi}_f) = \boldsymbol{0}, \; \dot{\boldsymbol{q}}_f = \dot{\boldsymbol{q}}(t_f), \end{array}\right\} \quad \text{(boundary constraints)}
\end{aligned} \tag{2.31}$$

where

$$\boldsymbol{C}(t_f, \boldsymbol{\chi}_f) = \boldsymbol{f}(\boldsymbol{q}_f) - \boldsymbol{x}_f = \boldsymbol{f}(\boldsymbol{q}(t_f)) - \boldsymbol{x}(t_f) = \boldsymbol{0}. \tag{2.32}$$

The resulting nonlinear programming (NLP) problem can be solved by numerical methods, e.g. based on direct collocation Diehl et al. [2006], yielding a joint trajectory $\boldsymbol{q}^*(t)$ and global minimum time t_f^*. Next, we associate to this motion the resulting Cartesian path, suitably expressed in a parametrized form $\boldsymbol{x}(s)$. Finally, this will be the input to two-step methods that handle robot redundancy. They will generate solution trajectories $\boldsymbol{q}^*_{\text{method}}(t)$ and associated motion times $t^*_{f_{\text{method}}}$, with method $= \{ACC,\ WPV,\ PGV\}$, and the results can be compared to each other and to the global minimum time solution on the same Cartesian path.

As a matter of fact, this fair procedure is needed since it is still prohibitive in general to address by numerical methods the global minimum time problem for redundant robots along predefined Cartesian paths (i.e., providing directly $\boldsymbol{x}(s)$ as input to the PTP problem). Indeed, two-step methods can only return longer motion times than the global optimal time. On the other hand, two-step approaches are computationally more efficient and accept any Cartesian path to start with.

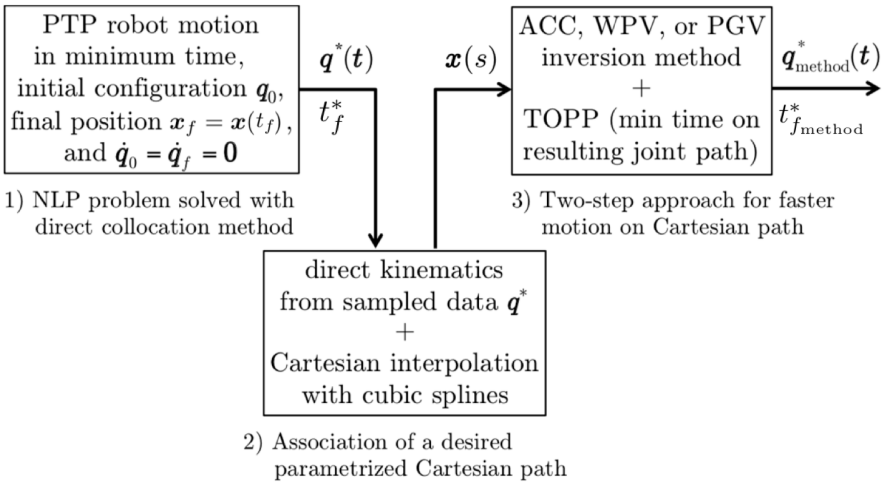


Figure 2.3: Comparison procedure between the global time-optimal solution and the solutions obtained on the same Cartesian path with two-step methods for redundant robots.

2.4 Results

We report here various simulations and experiments performed to illustrate the comparison between the proposed solution *ACC* in (2.20) and the methods *WPV* in (2.15) and *PGV* in (2.12). All methods have been implemented in MATLAB. Three study cases are considered: a 3R planar arm and the UR10 manipulator in simulations, and a 7R KUKA LWR 4 lightweight robot in simulations and experiments.

3R planar arm

The first robot considered is a 3R planar arm ($n = 3$) with links of equal length $l = 0.5$ [m], uniformly distributed mass $m_l = 1$ [kg], and moment of inertia $I_l = m_l l^2/12$. The end-effector position ($m = 2$) should follow a path on the horizontal plane, so that the degree of redundancy is $n - m = 1$. Torque and velocity limits have been set respectively to ± 20 [Nm] and ± 10 [rad/s], for all three joints.

Three different collections of comparisons will be introduced. In the first one, the arm should follow in the Cartesian plane a predefined straight linear path of 1.25 [m] length where

$$\boldsymbol{x}(s) = \begin{pmatrix} as \\ b - s \end{pmatrix}, \qquad s \in [0, 1], \tag{2.33}$$

with $a = 0.75$ [m] and $b = 1.2$ [m]. The initial configuration was predefined to be $\boldsymbol{q}(0) = \begin{pmatrix} \pi/4 & \pi/4 & \pi/4 \end{pmatrix}^T$, which corresponds to $\boldsymbol{x}(0)$, without using the preliminary optimization (2.29). To verify only the effect of dynamic bounds along the whole path, kinematic constraints were not considered.

Table 2.1: Minimum motion times for the first comparison collection with the 3R planar arm along a predefined linear task using three different inverse differential solution methods with the best parameters used for each method.

Method	t_f^* [s]
PGV ($\delta = 0.0$)	0.6082
PGV ($\delta = 0.005$)	0.5535
WPV ($\gamma = 1.3$)	0.4346
ACC ($\gamma = 0.8$, $\delta_2 = 0.1$, $\delta_{1,\max} = 100$, $\boldsymbol{D} = 0.5 \cdot \boldsymbol{I}$)	**0.3840**

For each method, in the first step, the parametrized path (2.33) is input to the ACC, PGV, and WPV methods, which generate different joint paths, sampled every $\Delta s = 0.001$. For better accuracy, all the available joint configurations samples are used within the cubic splines interpolation. In the second (and common) step, the time-optimal motion is obtained on each joint path using TOPP Pham [2014]. Since the motion task was planned to be rest-to-rest (zero joint velocity at the start and at the end), the constraints $\dot{s}(0) = \dot{s}(1) = 0$ were also imposed in TOPP.

Figure 2.4 shows a stroboscopic view of the best solution found for each method, while the associated minimum times are reported in Tab. 2.1. For this robot and motion task, the proposed second-order solution ACC achieves the smallest minimum time compared to the other two methods, with an improvement of 30% over PGV and 12% over WPV. As shown by Fig. 2.4, each solution produces in fact a different path in the joint space, which leads to a different MVC_t curve and a different optimal trajectory in the phase plane. Although the MVC_t related to WPV is higher than the one obtained with PGV, allowing in principle larger pseudo-velocities and thus faster motion, this feature is not exploited efficiently and the optimal trajectory remains far from this curve in the intermediate range of s values. Instead, with the ACC solution the obtained MVC_t is the highest one, and the time-optimal trajectory is able to cover most of the area lying under it. Figure 2.5 shows the joint velocities and torques obtained using the ACC method: for every value of the parameter s there is at least one joint saturating its torque limit. There is indeed no velocity saturation since kinematic constraints have not been considered. The joints start and end with zero velocities.

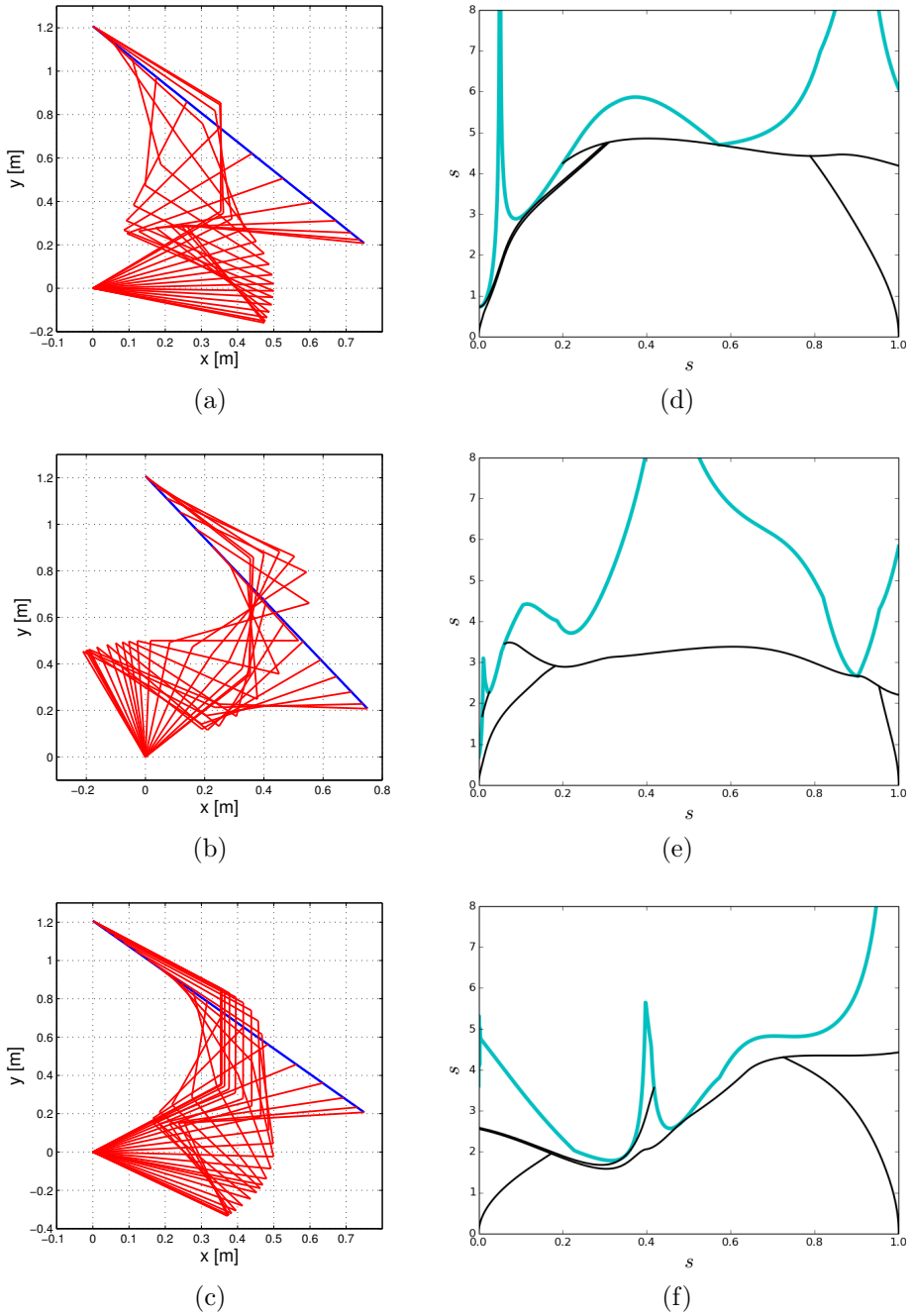


Figure 2.4: First comparison collection: Stroboscopic motions of the 3R planar arm and optimal trajectories in the phase plane using differential inverse kinematics solutions: (a,d) with the proposed *ACC*; (b,e) with *WPV*; (c,f) with *PGV*. In (d,e,f), the cyan curves represent the MVC_t and the black lines are the obtained time-optimal profiles. Only the dynamic constraints are considered.

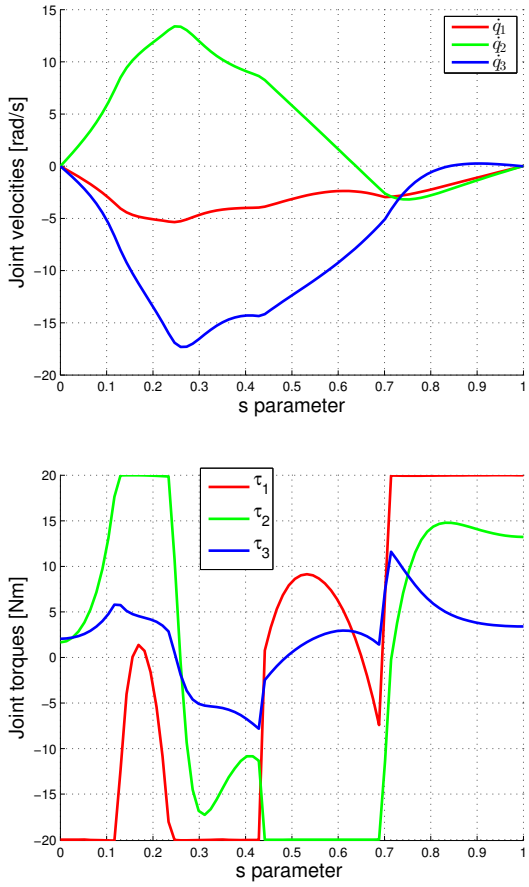


Figure 2.5: Joint velocities and torques in the s space for the 3R planar arm with the *ACC* solution along a predefined linear task. Only the dynamic constraints are considered.

In the second collection, different comparisons with global minimum time optimization along the same Cartesian paths are achieved considering the dynamic and kinematic constraints. For a given initial configuration $\boldsymbol{q}_0$, a desired final end-effector position $\boldsymbol{x}_f$ and zero initial/final joint velocities ($\dot{\boldsymbol{q}}_0 = \dot{\boldsymbol{q}}_f = \mathbf{0}$), the global PTP minimum time optimization problem (2.31) is solved using the direct collocation method in the trajectory optimization library *OptimTraj* Matthew [2016]. In this case the equality constraint (2.32) can be written via the direct kinematics as

$$\begin{aligned}\boldsymbol{C}(t_f, \boldsymbol{\chi}_f) = \\ \begin{pmatrix} l_1 \cos q_1(t_f) + l_2 \cos(q_1(t_f) + q_2(t_f)) + l_3 \cos(q_1(t_f) + q_2(t_f) + q_3(t_f)) \\ l_1 \sin q_1(t_f) + l_2 \sin(q_1(t_f) + q_2(t_f)) + l_3 \sin(q_1(t_f) + q_2(t_f) + q_3(t_f)) \end{pmatrix} \\ - \boldsymbol{x}_f = \mathbf{0}.\end{aligned} \tag{2.34}$$

Table 2.2: Motion tasks for the second comparison collection with the 3R planar arm.

	Initial configuration $\boldsymbol{q}_0$ [rad]	Final Cartesian position $\boldsymbol{x}_f$ [m]
Task 1	$(3\pi/8, \pi/4, -\pi/4)^T$	(-0.4,0.8)
Task 2	$(\pi/4, \pi/4, \pi/4)^T$	(-0.4,0.4)
Task 3	$(\pi/4, \pi/4, \pi/4)^T$	(0,-1)

Table 2.3: Minimum motion times [s] for the second comparison collection with the 3R planar arm, using global optimization and two-step solution methods (the best parameters used for each method are indicated).

	Direct collocation	ACC $\{\gamma, \delta_2, \delta_{1,\max}, \boldsymbol{D}\}$	WPV $\{\gamma\}$	PGV $\{\delta\}$
Task 1	0.2499	0.2689 $\{0.655, 1, 210, 2\boldsymbol{I}\}$	0.3357 $\{0.5\}$	0.4394 $\{0\}$
Task 2	0.2699	0.3021 $\{1, 0.8, 3500, 160\boldsymbol{I}\}$	0.3100 $\{1\}$	0.3867 $\{0.1\}$
Task 3	0.4415	0.5332 $\{0, 0.6, 300, 16\boldsymbol{I}\}$	0.5513 $\{0.1\}$	0.5550 $\{1\}$

We considered three different motion tasks with the boundary conditions specified in Tab. 2.2. In order to apply a two-step solution method in the presence of redundancy, we follow the procedure in Fig. 2.3: from the sequence of robot configurations in the time-optimal trajectory $\boldsymbol{q}^*(t)$, a corresponding sequence of end-effector positions is computed via the direct kinematics of the 3R robot arm, and then interpolated in the Cartesian space using cubic splines. This parametrized path is used as a reference for the ACC, PGV, and WPV methods and the following procedures to get the optimal time motion are the same as previous. For illustration, the resulting Cartesian paths corresponding to the globally PTP time optimal motion specified in Tab. 2.2 are shown in figures (2.8-2.10) together with the stroboscopic views of the best solutions found for each method.

Table 2.3 reports the comparative results obtained on the three motion tasks of Tab. 2.2, together with the parameters used for each method/task. The resulting globally optimal joint velocity and torque profiles for the first motion task are shown in Fig. 2.6. The proposed second-order solution ACC returns the fastest motion time among the three methods, i.e., also the closest to the global optimal solution. The resulting minimum time for the first motion task is only 7.6% longer than the global optimal solution. On the three tasks, the average increase of the motion time for the ACC, WPV, and PGV methods is, respectively, 13.4%, 24.7%, and 48.2% with respect to the global optimal solution. Figure 2.7 shows the joint velocities and torques obtained using the ACC method for the first motion task: at every instant, at least one joint is saturated either to its torque or velocity limit.

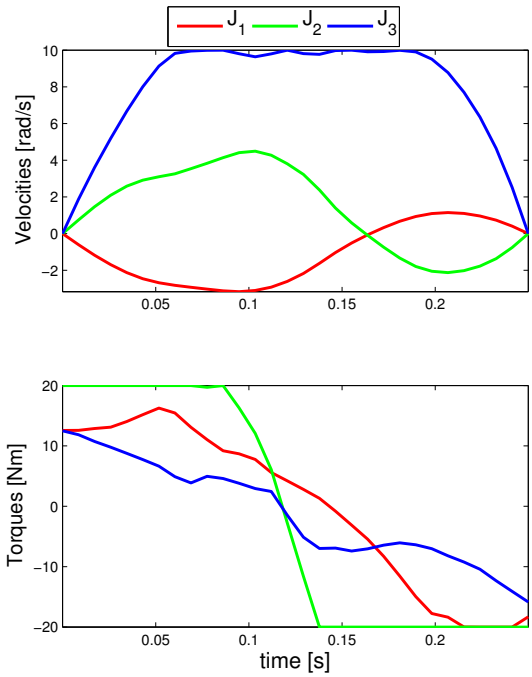


Figure 2.6: Global time-optimal joint velocities and torques for the 3R planar arm on the Task 1 of Tab. 2.2 using direct collocation method.

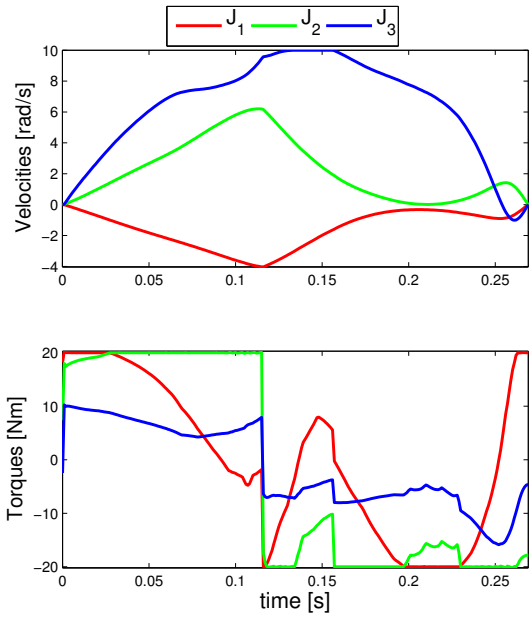


Figure 2.7: Joint velocities and torques for the 3R planar arm with the *ACC* solution on the Task 1 of Tab. 2.2.

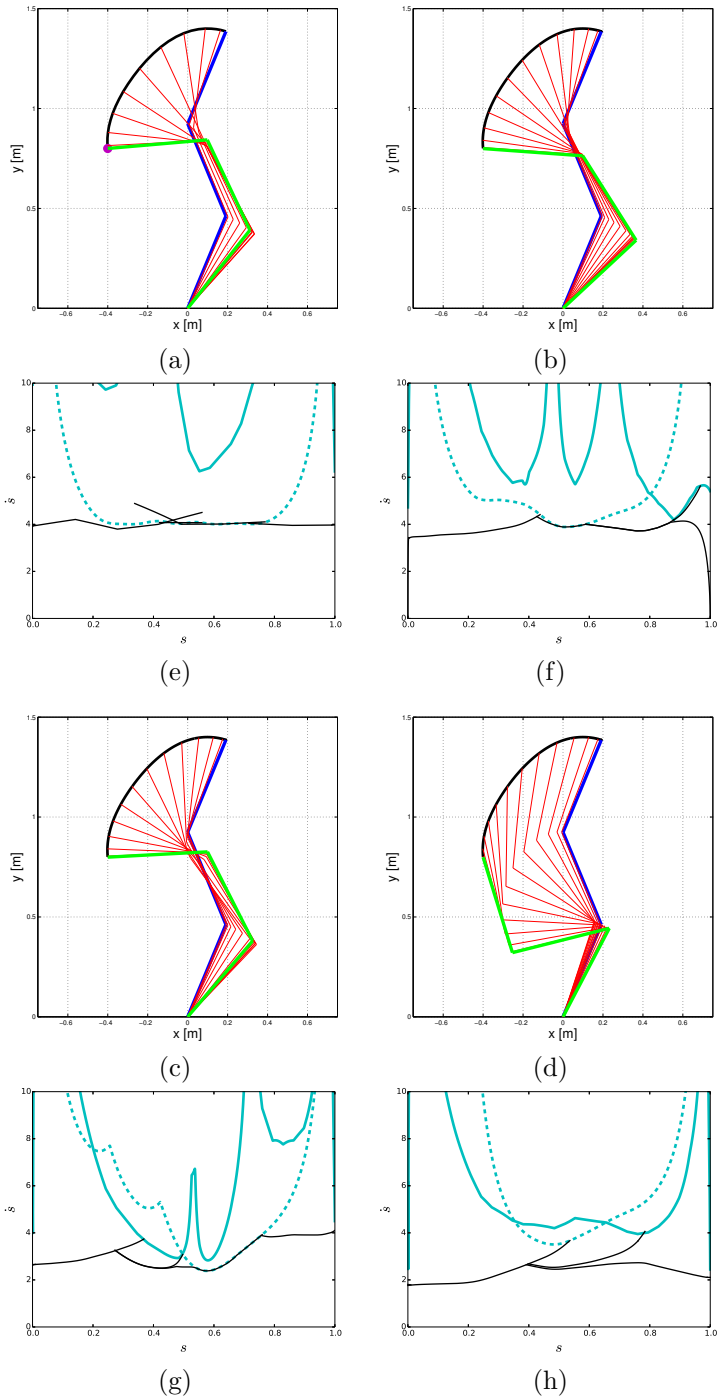


Figure 2.8: Stroboscopic motion of the 3R planar arm and optimal trajectories in the phase plane using different solutions for the first motion task in Tab. 2.2: (a-e) with direct collocation method; (b-f) with the proposed *ACC*; (c-g) with *WPV*; (d-h) with *PGV*. The initial configuration $\boldsymbol{q}_0$ (in blue) is always the same. The purple point in (a) is the final position $\boldsymbol{x}_f$ used to generate the Cartesian path from the PTP optimization. The resulting Cartesian path $\boldsymbol{x}(s)$ (in black) is used then by all two-step methods, while the obtained final configurations are shown in green. In (e,f,g,h), the cyan lines are the maximum velocity curves MVC_t (solid) and MVC_v (dotted), while the black lines are the obtained time-optimal profiles.

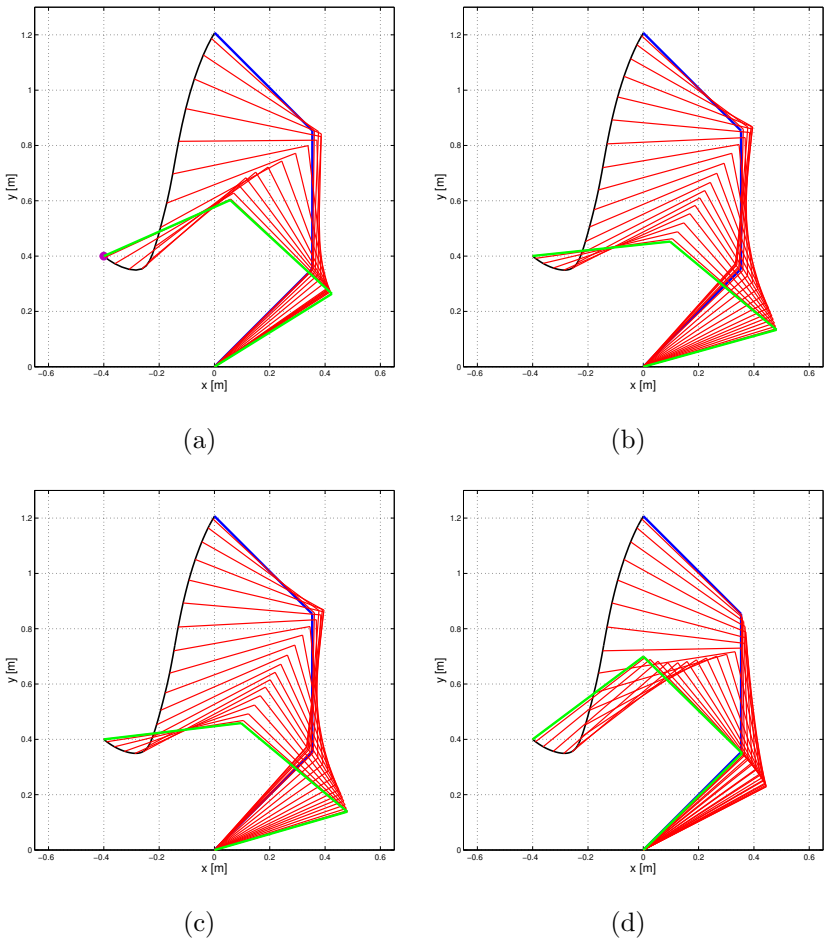


Figure 2.9: Stroboscopic motion of the 3R planar arm using different solutions for the Task 2 in Tab. 2.2: (a) with direct collocation method; (b) with the proposed *ACC*; (c) with *WPV*; (d) with *PGV*. Same legends as in Fig. 2.8.

Figure 2.8 shows stroboscopic views of the best solutions found for each method on the first motion task, and the associated evolutions in the phase plane $(s, \dot{s})$. Each solution produces in fact a different path in the joint space, which leads also to different maximum velocity curves and associated optimal trajectories. In this case, the MVC curves of the *PGV* method are higher than those of *WPV*, but this feature is not exploited efficiently. Instead, the *ACC* solution leads to the highest MVC curves, and the optimal trajectory is able to cover most of the underlying area.

For further comparison, the joint path corresponding to the global time-optimal solution (obtained with the direct collocation method) were also fed into the TOPP algorithm. As expected, the resulting minimum time using TOPP is exactly equal to the optimal time obtained from the direct collocation method. The phase-plane plot in Fig. 2.8(e) clearly shows how the optimal trajectory fully exploits the area below the MVC_v curve.

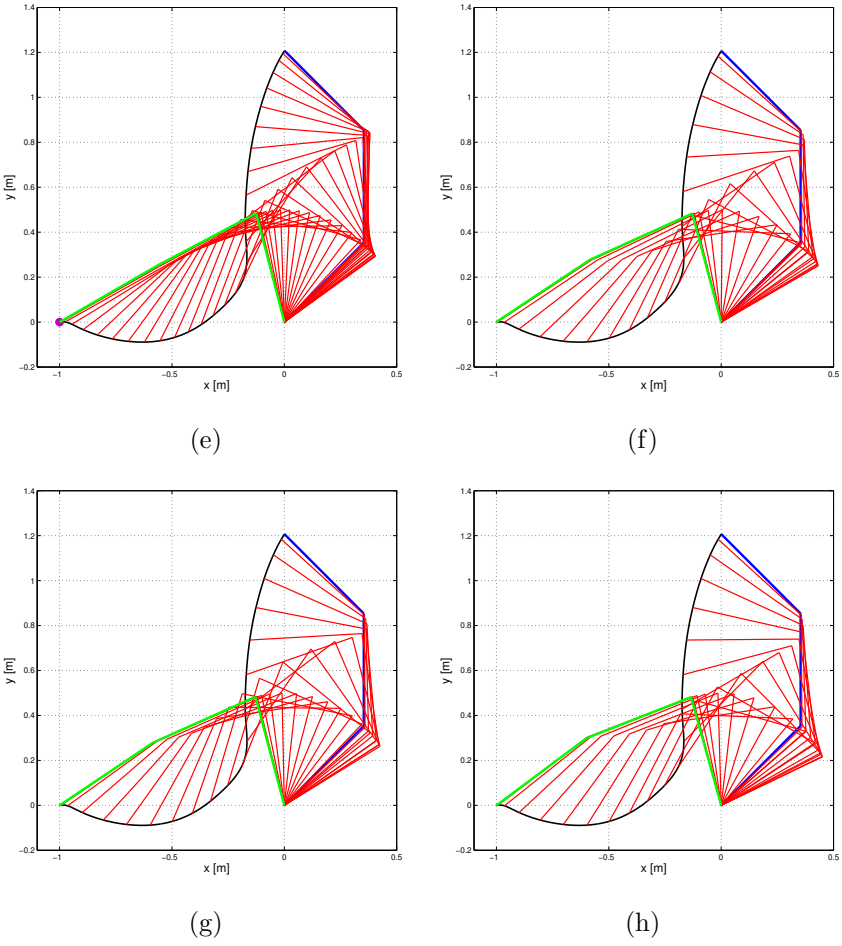


Figure 2.10: Stroboscopic motion of the 3R planar arm using different solutions for the Task 3 in Tab. 2.2: (a) with direct collocation method; (b) with the proposed ACC; (c) with WPV; (d) with PGV. Same legends as in Fig. 2.8.

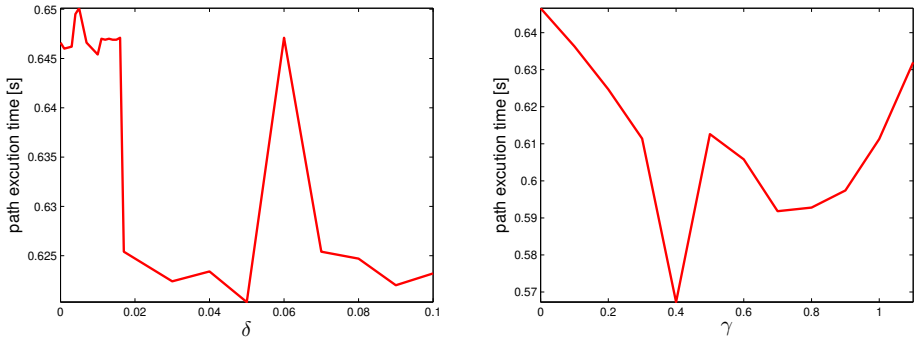


Figure 2.11: Using the PGV (left) and WPV (right) methods with different values of δ and γ parameters, respectively. For each method, the trajectory cannot be re-parameterized in time for values of the parameter lying outside the range shown in the associated figure.

The last comparisons collection will done by replacing the terminal constraint in the Cartesian space (2.34) by the specification of a final configuration $\boldsymbol{q}_f$ as

$$\boldsymbol{C}(t_f, \boldsymbol{\chi}_f) = \boldsymbol{q}(t_f) - \boldsymbol{q}_f = \boldsymbol{0}. \tag{2.35}$$

With this new formulation, once the global time-optimal solution has been found, e.g., using again the algorithms in the *OptimTraj* library, the associated Cartesian path and only the initial robot configuration $\boldsymbol{q}_0$ are used by the differential inversion methods for the redundant robot.

Figure 2.12 shows the robot motion corresponding to the different solution methods. The optimal execution time found with the direct collocation method is $t_f^* = 0.4544$ s, while the best execution times in the ACC, WPV, and PGV solutions are 0.4502, 0.5673 and 0.6220 s, respectively. In this example, the motion time $t_{f_{ACC}}^* = 0.4502$ s obtained using our proposed ACC method is even smaller than the 'optimal' value obtained with the direct collocation method. Indeed, this can happen because the final configuration in the ACC solution is left unconstrained, as it was instead in the formulation (2.35) of the collocation method; this provides more flexibility in exploiting the robot redundancy. The previous examples show that the proposed ACC method has the capability to return solutions that are close to the global optimal one and outperforms the previous two-step methods.

For fair comparisons, the different solutions in the previous examples start always from the same initial configuration which has been chosen randomly. Also, for each method and for each motion task, the tuning of parameters is done separately so as to achieve the best possible performance for each task. Specific ranges are chosen for each parameter and the best values are searched on a discretized grid by evaluating a large number of simulations. For illustration, we show an example of how this is done for the motion task in Fig. 2.12. The best performance with the WPV and PGV methods is obtained according to the path execution times in Fig. 2.11. In the ACC solution, the γ and $\boldsymbol{D}$ parameters are more influential than δ_2 and $\delta_{1,max}$. For efficiency, the first two parameters are tuned together, and then kept fixed to tune the latter ones.

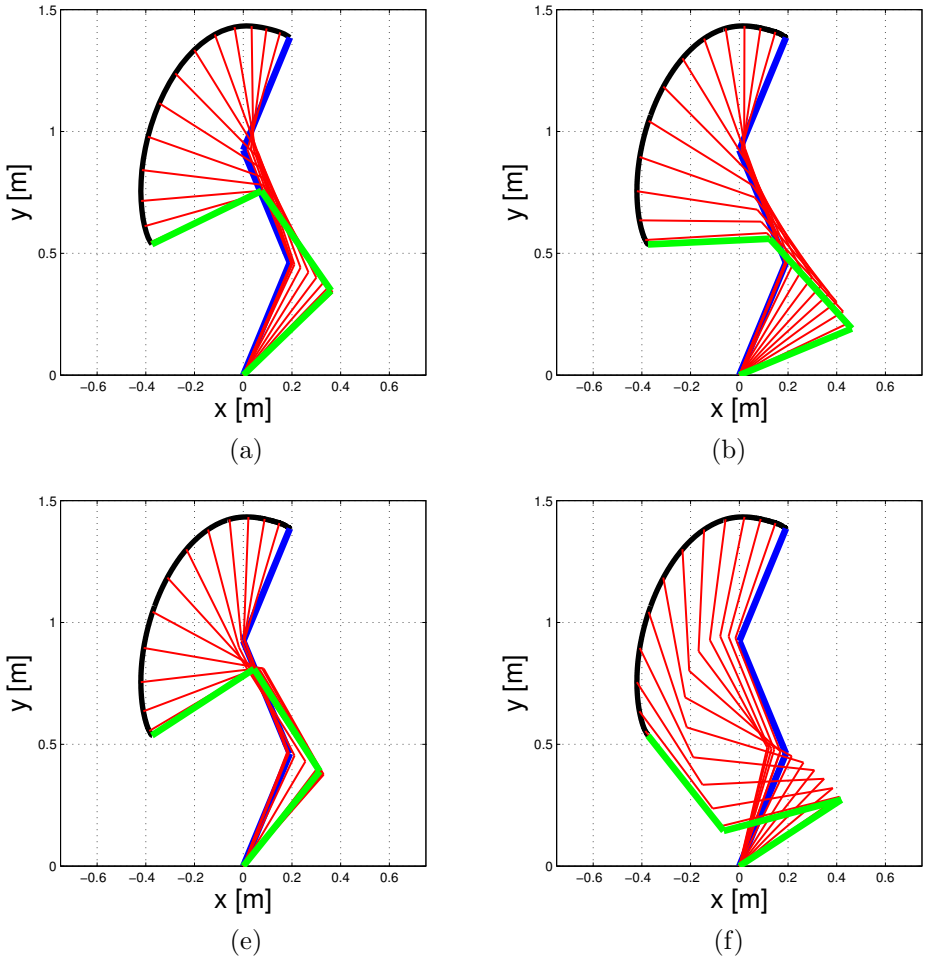
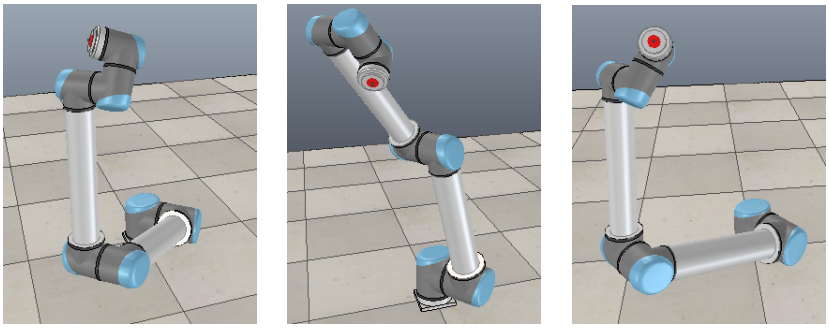


Figure 2.12: Stroboscopic motions of the 3R planar arm using different solution methods. (a) Direct collocation method: the blue and green arms are the constrained initial and final configurations, respectively, and the black curve is the Cartesian path resulting from the time-optimal solution. (b) *ACC*, (c) *WPV*, and (d) *PGV* methods: here, the blue arm is the given initial configuration $\boldsymbol{q}_0$, the black Cartesian curve is a problem constraint in common, while the green arms are the different resulting final configurations obtained with the three methods.

Table 2.4: Minimum motion times along a linear path for the UR10 manipulator using ACC method from three different initial configurations (with V-REP views).

Initial configuration $\boldsymbol{q}_0$ [rad]	$t^*_{f_{ACC}}$ [s]
$\boldsymbol{q}_0^{(a)} = (\ 0.87 \quad -0.28 \quad -1.29 \quad -2.05 \quad -2.49\ )^T$	**0.5924**
$\boldsymbol{q}_0^{(b)} = (\ 0.67 \quad -1.26 \quad 0.41 \quad -4.69 \quad -5.18\ )^T$	0.6437
$\boldsymbol{q}_0^{(c)} = (\ 0.62 \quad -0.28 \quad -1.34 \quad -2.17 \quad -0.76\ )^T$	0.6511



(a) (b) (c)

UR10 manipulator

The second case study considered is the 6R UR10 manipulator from Universal Robots, see Appendix A. In all computations, the robot dynamic model presented in Gaz et al. [2018] has been used, together with the nominal symmetric joint torque and velocity limits provided by the manufacturer in Tab. A.1. Different motion tasks were assigned to the end-effector position in the 3D Cartesian space ($m = 3$) as a function of the parameter s. Since joint 6 of the robot does not affect these positional tasks, it was kept at rest. Therefore, we work with only $n = 5$ joints and the degree of redundancy is $n - m = 2$.

The first task is a linear path of 0.75 [m] length. To study the influence of the initial configuration of the manipulator, the ACC method with the same parameters was applied starting from the three different configurations shown in Tab. 2.4, all associated to the desired initial position $\boldsymbol{x}(0)$. Starting with the configuration (a) found with (2.29), reductions of $8 - 9\%$ were obtained w.r.t. the two other random initializations. When executing the linear task from the initial configuration (a) in Tab. 2.4, the ACC solution shown in Figs. 2.13 and 2.14 leads to the best minimum time, with 23.9% and 35.8% reductions w.r.t. the WPV and PGV methods, respectively (see Tab. 2.5).

The second assigned task is a circular path of $R = 0.25$ [m] radius in the xz-plane, with the initial configuration computed using (2.29). Also in this case, the ACC solution illustrated in Fig. 2.15 outperforms in terms of minimum motion time the WPV and PGV methods, respectively by 40% and 33.7%. The

two time-optimal trajectories obtained using the ACC solution match the curve MVC_v almost along the entire path, following the bounds specified by α and β in (2.5) only right at the beginning and towards the end of the motion. In both cases, the MVC_v curve is much lower than MVC_t, and the robot reaches its velocity limits quickly because of its large torque/acceleration capabilities. Also, the joint torque and velocity profiles in Figs. (2.14–2.15) confirm that there is at least one joint saturated to its torque or velocity limit at every point of the path.

Table 2.5: Minimum motion times for the UR10 manipulator using three different inverse differential solution methods.

Method	Task	t_f^* [s]
PGV ($\delta = 0.3$)	linear	0.9231
WPV ($\gamma = 0.3$)	linear	0.7786
ACC ($\gamma = 1$, $\delta_2 = 0.1$, $\delta_{1,\max} = 1000$, $\boldsymbol{D} = \boldsymbol{I}$)	linear	**0.5924**
PGV ($delta = 0.2$)	circular	1.7034
WPV ($\gamma = 0.4$)	circular	1.883
ACC ($\gamma = 1$, $\delta_2 = 0.2$, $\delta_{1,\max} = 1000$, $\boldsymbol{D} = \boldsymbol{I}$)	circular	**1.1293**

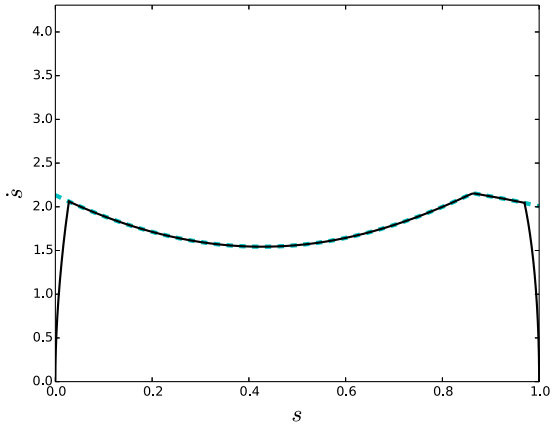


Figure 2.13: Linear task with the UR10 manipulator using the ACC solution. The time-optimal profile (black lines) in the $(s, \dot{s})$ plane is almost superposed with the maximum velocity curve MVC_v (the MVC_t curve is too high to be shown).

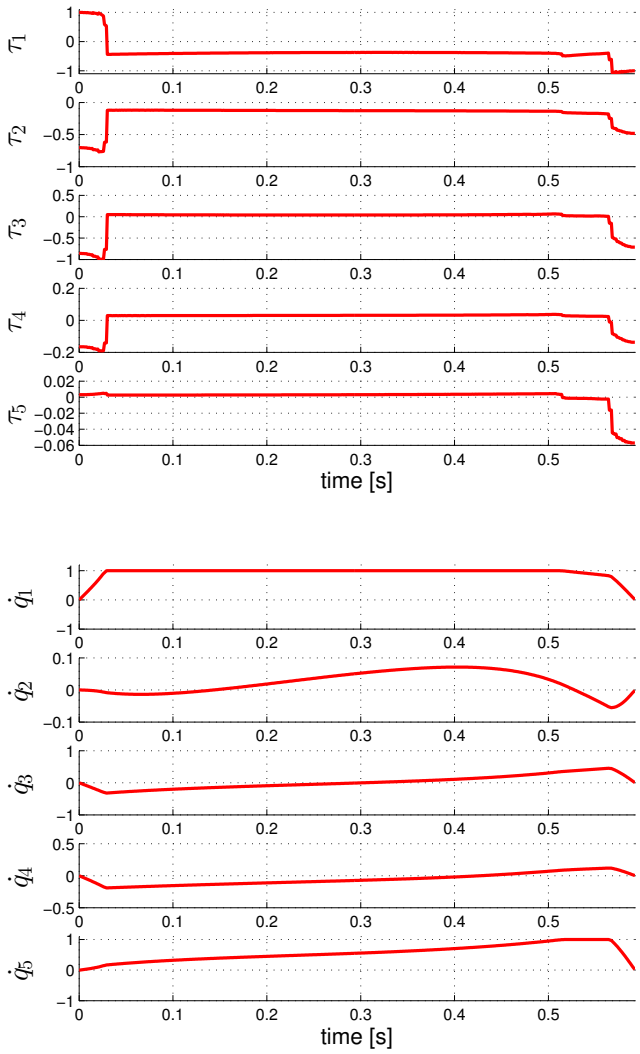


Figure 2.14: Normalized joint torques and joint velocities for the linear task with the UR10 manipulator using the ACC solution.

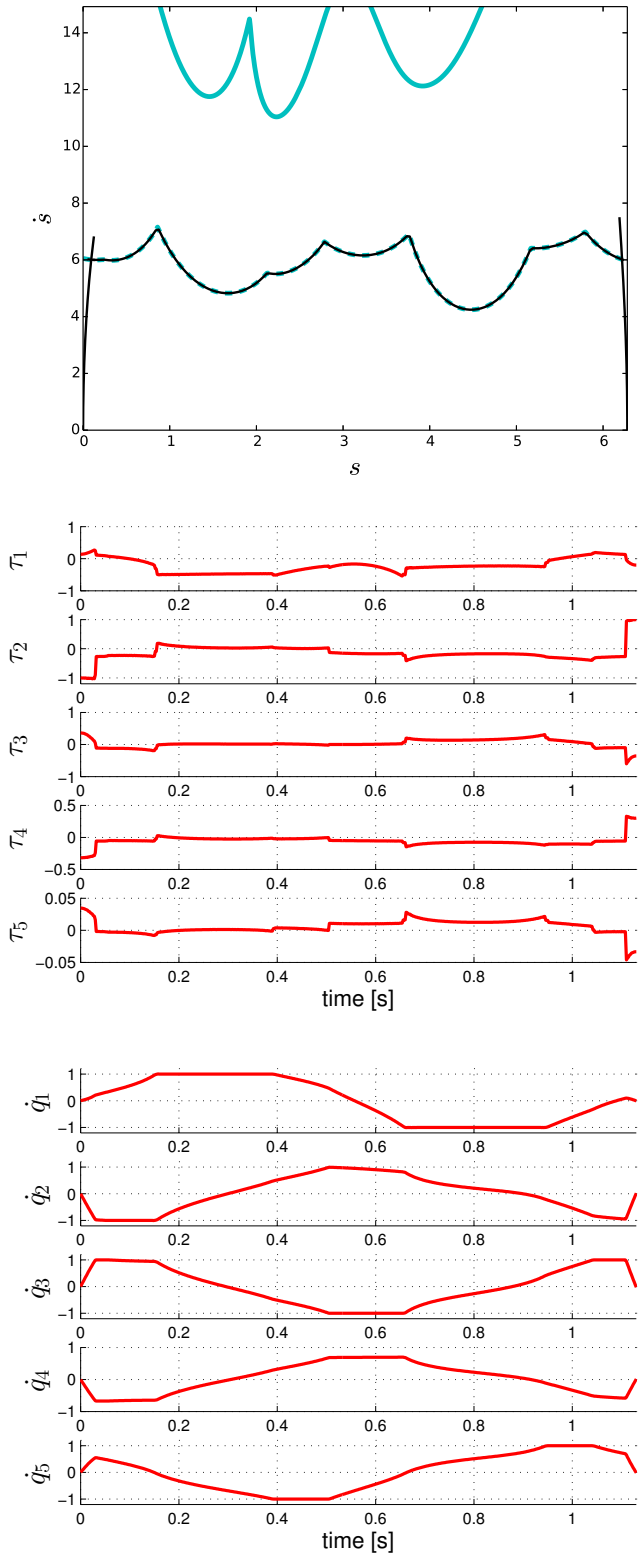
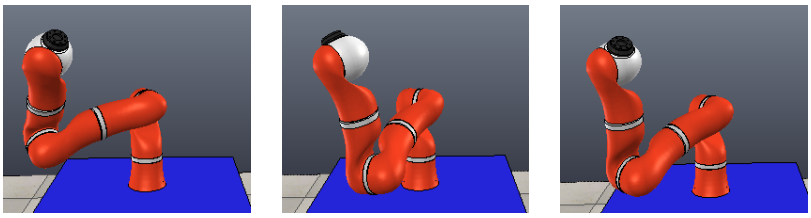


Figure 2.15: Circular task with the UR10 manipulator using the ACC solution. In the $(s, \dot{s})$ plane [top], the cyan lines are the maximum velocity curves MVC_t (solid) and MVC_v (dotted).

Table 2.6: Minimum motion times along a linear path for the KUKA LWR robot using the ACC method from three different initial configurations (with V-REP views).

Initial configuration $\boldsymbol{q}_0$ [rad]	$t^*_{f_{ACC}}$ [s]
$\boldsymbol{q}_0^{(a)} = (\,-1.12 \;\; 1.80 \;\; -0.55 \;\; 1.71 \;\; 2.43 \;\; 0.29\,)^T$	**0.4743**
$\boldsymbol{q}_0^{(b)} = (\,-0.34 \;\; 1.94 \;\; 0.16 \;\; 1.71 \;\; 1.20 \;\; -0.43\,)^T$	0.5144
$\boldsymbol{q}_0^{(c)} = (\,-0.72 \;\; 1.94 \;\; -0.08 \;\; 1.73 \;\; 1.60 \;\; 0.51\,)^T$	0.5242



(a) (b) (c)

KUKA LWR IV

As a third case study, we have considered a 7R KUKA LWR lightweight robot (see Appendix B) and compared the different inverse differential methods using two Cartesian tasks defined for the position of the end-effector flange center ($m = 3$). Since the rotation of joint 7 has no effect on it, the final flange was frozen resulting in only $n = 6$ active joints, with a redundancy degree $n-m = 3$. All computations were done using the dynamic model identified in Gaz et al. [2014] together with the nominal symmetric joint torque and velocity limits provided by the manufacturer in Tab. B.1.

The first motion task was a linear path of length 0.66 [m]. The ACC method with the same parameters was applied starting from the three different configurations given in Tab. 2.6, the first one obtained using (2.29) and the other two chosen randomly, all associated to the same initial position $\boldsymbol{x}_0 = (-0.4, 0.25, 0.3)$ [m]. The solution obtained when starting with the configuration (a) provided a reduction of the minimum time by 8.5-10.5%. When executing the linear motion task from the initial configuration (a) in Tab. 2.6 using the other two-step methods, the proposed ACC solution leads to the fastest motion time, with an improvement of 22.6% and 31.7% over the WPV and PGV solutions.

We considered a second motion task along an ellipse in the 3D space, with major and minor axes $r_M = 0.2$ and $r_m = 0.1$ [m], starting the robot at rest from the configuration $\boldsymbol{q}_0 = \big(1.15 \;\; -0.54 \;\; 0.10 \;\; 1.47 \;\; -0.30 \;\; 0.76\big)^T$ [rad], which corresponds to $\boldsymbol{x}_0 = (0.2, 0.6, 0.2)$ [m]. Again, the ACC solution provided the best result, with a reduction of the motion time by 14.6% over the other two methods. These results are summarized in Tab. 2.7, which reports also the (best) set of parameters used for each method/task.

Table 2.7: Minimum motion times for the KUKA LWR robot using different two-step solution methods.

Method	Task	t_f^* [s]
PGV ($\delta = 0.3$)	linear	0.6901
WPV ($\gamma = 1$)	linear	0.6127
ACC ($\gamma = 0.5$, $\delta_2 = 0.4$, $\delta_{1,\max} = 100$, $\boldsymbol{D} = 5\boldsymbol{I}$)	linear	**0.4743**
PGV ($\delta = 0.6$)	ellipse	1.2
WPV ($\gamma = 0.35$)	ellipse	1.2
ACC ($\gamma = 0.5$, $\delta_2 = 0.4$, $\delta_{1,\max} = 100$, $\boldsymbol{D} = 5\boldsymbol{I}$)	ellipse	**1.0245**

In view of the good results obtained with the ACC method, the two motion tasks on the linear and the elliptic paths were implemented in experiments on the KUKA LWR IV robot using the FRI library KUK [2011] in position control mode. Due to residual uncertainty in the robot dynamic model, the ACC solution was re-generated in a conservative way, using only 95% of the maximum available nominal torques and joint velocities. The new motion times were 0.496 s for the linear task and 1.081 s for the ellipse task (compare with Tab. 2.7). Figure 2.17 shows the experimental results along the ellipse task where the joint velocities and torques *normalized* with respect their nominal values in Tab. B.1. The minimum time planned torque (in red) of the second joint saturates at the start, near the middle, and toward the end of the trajectory. In the rest of the trajectory, the second and fourth joint velocities saturate in turn, consistently with the optimal trajectory in the phase plane.

During task execution with the KUKA LWR, the torques are measured by the available joint torque sensors. The differences between planned and executed/measured torques in Figs. 2.17 and 2.16 are due to unmodeled dynamics (motor friction, joint elasticity) neglected in the optimization, measurement noise (encoders and torque sensors), as well as non-idealities of the low-level robot controller. Because of the latter, the joint torques cannot follow perfectly the planned discontinuities of the optimal torques at the switching points. Despite of this, the Cartesian path tracking performance in Figs. 2.19 and 2.18 is still reasonable for such a fast robot motion. High errors occur near the switching points.

For comparison, the linear task is performed experimentally using PGV and WPV methods with 95% of robot limits. The corresponding Cartesian errors are shown in Fig. 2.20, and the error norm for different two-step solutions is shown in Fig. 2.21. The largest peak error is obtained with the ACC method, which is also the one with the fastest motion time. On the other hand, the tracking error vanishes as the motion comes to an end, whereas some residual error is left with the other two methods.

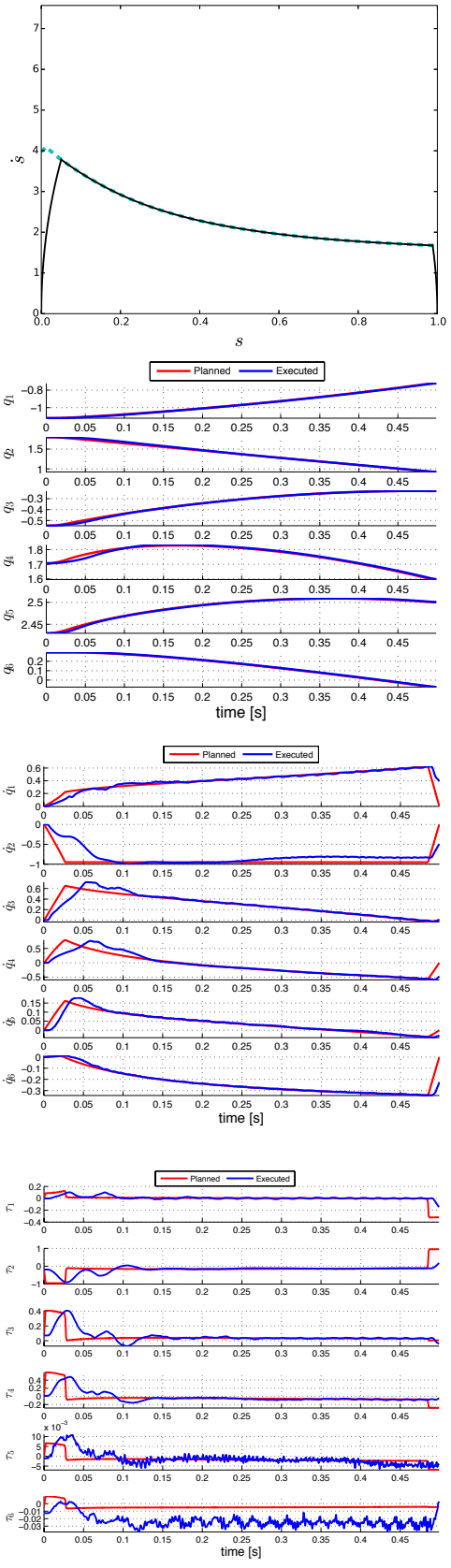


Figure 2.16: [from top] Optimal phase-plane trajectory in the minimum time experiment with the KUKA LWR robot on a linear path using the ACC method. Planned and executed joint positions. Normalized joint velocities and torques.

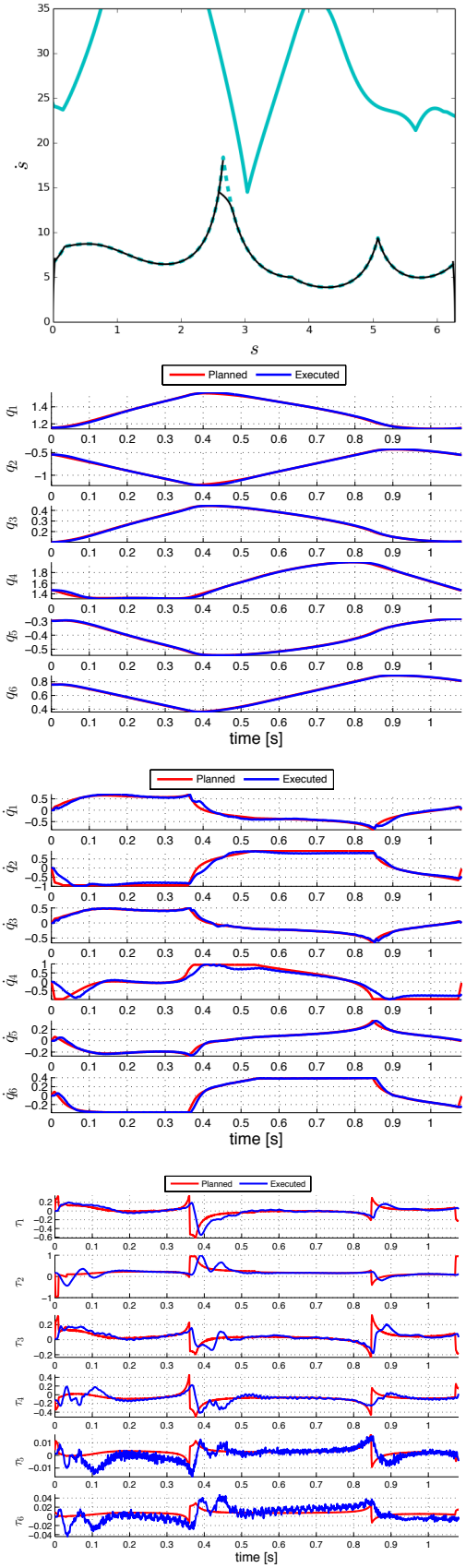


Figure 2.17: [from top] Optimal phase-plane trajectory in the minimum time experiment with the KUKA LWR robot on an ellipse path using the *ACC* method. Planned and executed joint positions. Normalized joint velocities and torques.

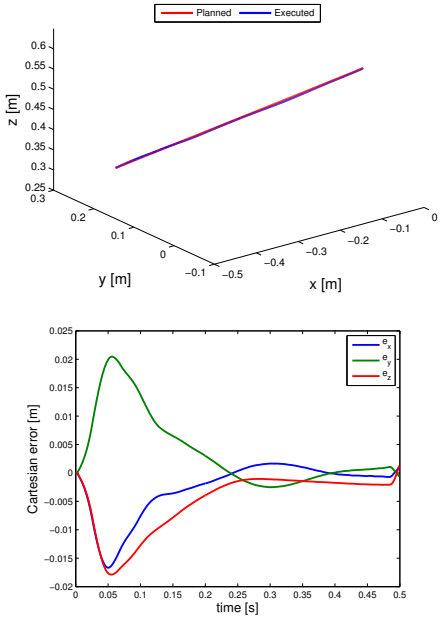


Figure 2.18: Planned and executed linear path [top]. The Cartesian error components in the minimum time experiment with a KUKA LWR IV robot using ACC method.

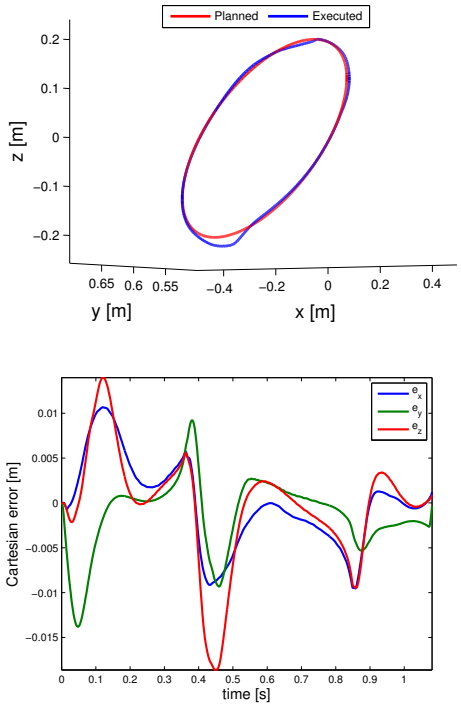


Figure 2.19: Planned and executed ellipse path [top] and Cartesian error components in the minimum time experiment with a KUKA LWR IV robot using ACC method.

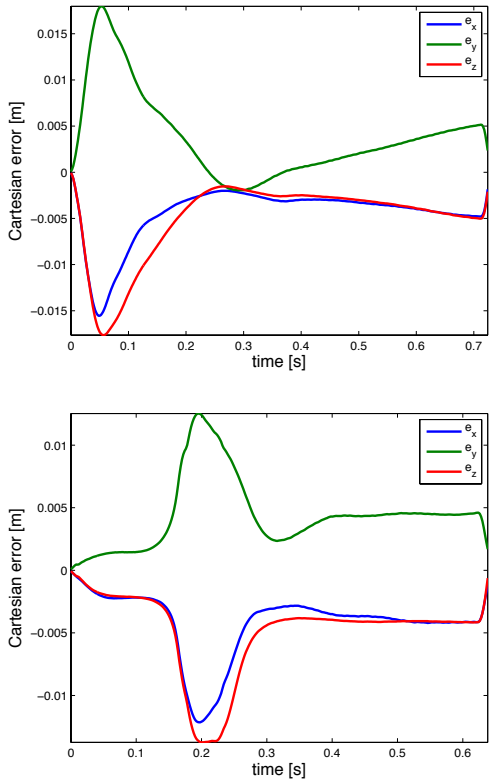


Figure 2.20: Using *PGV* method [top] and *WPV* method [bottom]. Cartesian error components in the minimum time experiments with a KUKA LWR IV robot and 95% of the maximum available nominal torques and joint velocities.

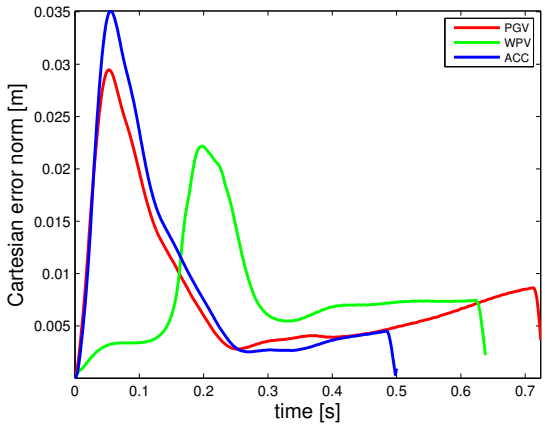


Figure 2.21: Cartesian error norm for the KUKA LWR robot tracing a linear path using different two-step solutions with 95% of the maximum available nominal torques and joint velocities.

Table 2.8: Mean Cartesian error norm for the KUKA LWR robot tracing a linear path using the ACC method with 95% $\dot{\boldsymbol{q}}^{\max}$ and different $\boldsymbol{\tau}^{\max}$ percentages.

% $\boldsymbol{\tau}^{\max}$	t_f^*[s]	Mean Cartesian error norm [m]
35	0.556	0.0068
45	0.530	0.0072
55	0.518	0.0075
65	0.510	0.0080
75	0.505	0.0087
85	0.501	0.0094
95	0.496	0.0095

A trade-off between faster motion times and better tracking performance can be achieved by downrating the maximum available nominal joint torque in Tab. B.1. From Fig. 2.21 and Tab. 2.8, using ACC solution with only 35% $\boldsymbol{\tau}^{\max}$ returns better optimal time and less mean Cartesian error norm than using other two solutions with 95% $\boldsymbol{\tau}^{\max}$. The latter trade-off can also be achieved by including additional constraints, such as torque rate bounds that eliminate critical discontinuities in the solution Constantinescu and Croft [2000].

Our two-step second-order method leads to faster motion times, but is still intended currently for off-line planning situations only. Real-time limitations are distributed between both steps, and depend on the length of the original Cartesian path, the path parameter sampling, the number of joints, and the complexity of the used robot dynamics, leading to running times in the order of seconds. On the other hand, finding the accurate global minimum time with a constrained solution trajectory by means of general numerical optimization techniques requires at present minutes to hours of computation.

3 Torque Optimization Control

3.1 Introduction

In this chapter, we introduce two basic variations to the minimum torque norm (*MTN*) scheme of Hollerbach and Suh [1987] that address the issue of unstable joint motion in redundant robots. In the first scheme, we propose the minimization of the joint torque norm over successive discrete-time samples within a short preview window (possibly, in the next sampling instant). Suitable dynamic approximations are introduced in the estimation of the future (but proximal) robot state, so as to keep a linear-quadratic (*LQ*) formulation for the problem. Such a dynamic optimization scheme, that we denote model-based preview (*MBP*), can be seen as a compromise between local and global redundancy resolution methods, trying to inherit the best of both worlds— real-time simplicity and proven stability.

In the second scheme, we choose to command torque by minimizing the norm of the difference with respect to a (opposing) torque proportional to the current generalized momentum of the robot. This induces a natural dynamic damping of the joint velocities, preventing their oscillations or growth. We label this a minimum torque norm solution with damping (*MTND*). The two proposed modifications can also be combined, leading to a model-based preview scheme with damping (*MBPD*). Again, these schemes are in the form of well-posed *LQ* problems, providing efficiently a closed-form solution.

The rest of the chapter is organized as follows. The general formulation of the instantaneous torque optimization as a *LQ* problem is reviewed in Sec. 3.2. Section 3.3 presents the derivation of the proposed optimization methods with model-based preview, as well as the damping extensions. Section 3.4 reviews the required trajectory planning for the desired Cartesian time law. Section 3.5 reports comparative simulation and experimental results using different robot systems.

The problem described in this Chapter has been addressed in the work by Al Khudir et al. [2019].

3.2 Instantaneous minimum torque solution

The standard linear-quadratic optimization problem for local torque minimization in (1.37) can be written as

$$\min_{\ddot{\boldsymbol{q}}^*} H = \frac{1}{2}\left\|\boldsymbol{\tau}^*\right\|^2 = \frac{1}{2}\,\ddot{\boldsymbol{q}}^{*T}\boldsymbol{Q}\ddot{\boldsymbol{q}}^* + \boldsymbol{r}^T\ddot{\boldsymbol{q}}^* + s \tag{3.1a}$$

$$\text{s.t. } \boldsymbol{\tau}^* = \boldsymbol{Q}^{\frac{1}{2}}\ddot{\boldsymbol{q}}^* + \boldsymbol{Q}^{-\frac{1}{2}}\boldsymbol{r} \tag{3.1b}$$

$$\boldsymbol{b} = \boldsymbol{A}\ddot{\boldsymbol{q}}^*, \tag{3.1c}$$

assuming that $\boldsymbol{Q}$ is a positive definite (weighting) matrix, matrix $\boldsymbol{A}$ has full (row) rank, $\boldsymbol{b} \in \mathbb{R}^n$, $\boldsymbol{r} \in \mathbb{R}^n$, and s is a scalar. All torque optimization problems in this chapter can be formulated as (3.1) for specific choices of $\boldsymbol{Q}$, $\boldsymbol{A}$, $\boldsymbol{b}$ and $\boldsymbol{r}$ (while s is irrelevant). Equation (3.1c) represents the task (kinematic) constraint, while (3.1b) simply provides the command torque $\boldsymbol{\tau}^*$ from the dynamic model equation (1.32) when the joint acceleration solution $\ddot{\boldsymbol{q}}^*$ is being used.

The unique solution to (3.1) is obtained with the method of Lagrange multipliers starting from the Lagrangian L definition

$$L = H + \boldsymbol{\lambda}^T\left(\boldsymbol{A}\ddot{\boldsymbol{q}} - \boldsymbol{b}\right), \tag{3.2}$$

and impose the necessary (here also sufficient) conditions for a constrained minimum of H:

$$\nabla_{\ddot{\boldsymbol{q}}^*} L = \left(\frac{\partial L}{\partial \ddot{\boldsymbol{q}}^*}\right)^T = \boldsymbol{Q}\ddot{\boldsymbol{q}}^* + \boldsymbol{r} + \boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{0}, \tag{3.3}$$

$$\nabla_{\boldsymbol{\lambda}} L = \left(\frac{\partial L}{\partial \boldsymbol{\lambda}}\right)^T = \boldsymbol{A}\ddot{\boldsymbol{q}}^* - \boldsymbol{b} = \boldsymbol{0}. \tag{3.4}$$

Solving for $\ddot{\boldsymbol{q}}^*$ from (3.3)

$$\ddot{\boldsymbol{q}}^* = -\boldsymbol{Q}^{-1}(\boldsymbol{r} + \boldsymbol{A}^T\boldsymbol{\lambda}), \tag{3.5}$$

and replacing in (3.4) yields the multiplier

$$\boldsymbol{\lambda} = -\left(\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^T\right)^{-1}\left(\boldsymbol{b} + \boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{r}\right). \tag{3.6}$$

Replacing (3.6) in (3.5)

$$\ddot{\boldsymbol{q}}^* = \boldsymbol{Q}^{-1}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^T\right)^{-1}\left(\boldsymbol{b} + \boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{r}\right) - \boldsymbol{Q}^{-1}\boldsymbol{r}, \tag{3.7}$$

which can be rewritten as

$$\ddot{\boldsymbol{q}}^* = \boldsymbol{A}_{\boldsymbol{Q}}^{\#}\left(\boldsymbol{b} + \boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{r}\right) - \boldsymbol{Q}^{-1}\boldsymbol{r}, \tag{3.8}$$

with the weighted pseudoinverse

$$\boldsymbol{A}_{\boldsymbol{Q}}^{\#} = \boldsymbol{Q}^{-1}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^T\right)^{-1}.$$

Note that the solution (3.8) can be written also as

$$\ddot{\boldsymbol{q}}^* = \boldsymbol{A}_{\boldsymbol{Q}}^{\#}\,\boldsymbol{b} - \left(\boldsymbol{I} - \boldsymbol{A}_{\boldsymbol{Q}}^{\#}\,\boldsymbol{A}\right)\boldsymbol{Q}^{-1}\boldsymbol{r} \tag{3.9}$$

to emphasize the presence of a control action in the null-space of the task matrix $\boldsymbol{A}$, represented by the term $\boldsymbol{Q}^{-1}\boldsymbol{r}$.

Let a sequence of control instants $t_k = kT_s$, $k = 0, 1, \ldots$, be given for a fixed sampling time $T_s > 0$. Denote by $\boldsymbol{q}_k = \boldsymbol{q}(t_k)$ and $\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}(t_k)$ the position and velocity (i.e., the state) of the robot at time $t = t_k$. We can select the acceleration $\ddot{\boldsymbol{q}}_k$, or equivalently the torque $\boldsymbol{\tau}_k$, at t_k by solving the problem

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}}_k} H_1 \;&= \tfrac{1}{2}\|\boldsymbol{\tau}_k\|^2 \\ \text{s.t.} \quad &\boldsymbol{\tau}_k = \boldsymbol{M}_k\ddot{\boldsymbol{q}}_k + \boldsymbol{n}_k \\ &\ddot{\boldsymbol{x}}_k = \boldsymbol{J}_k\ddot{\boldsymbol{q}}_k + \boldsymbol{h}_k, \end{aligned} \tag{3.10}$$

where we used the shorthand notations

$$\begin{aligned} \boldsymbol{M}_k &= \boldsymbol{M}(\boldsymbol{q}_k), & \boldsymbol{n}_k &= \boldsymbol{n}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k), \\ \boldsymbol{J}_k &= \boldsymbol{J}(\boldsymbol{q}_k), & \boldsymbol{h}_k &= \boldsymbol{h}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k). \end{aligned}$$

Problem (3.10) is in the form (3.1) with the substitutions

$$\begin{gathered} \ddot{\boldsymbol{q}}^* = \ddot{\boldsymbol{q}}_k, \quad \boldsymbol{Q} = \boldsymbol{M}_k^2, \quad \boldsymbol{A} = \boldsymbol{J}_k, \\ \boldsymbol{\tau}^* = \boldsymbol{\tau}_k, \quad \boldsymbol{r} = \boldsymbol{M}_k\boldsymbol{n}_k, \quad \boldsymbol{b} = \ddot{\boldsymbol{x}}_k - \boldsymbol{h}_k, \quad s = \tfrac{1}{2}\boldsymbol{n}_k^T\boldsymbol{n}_k. \end{gathered} \tag{3.11}$$

Thus, the optimal acceleration $\ddot{\boldsymbol{q}}^* = \ddot{\boldsymbol{q}}_k$ is given by (3.8) and the associated optimal torque $\boldsymbol{\tau}^* = \boldsymbol{\tau}_k$ is then obtained from (3.1b). Using (3.9), we see that the null-space action that minimizes the joint torque norm is given by $\boldsymbol{M}_k^{-1}\boldsymbol{n}_k$. This is indeed the solution found in Hollerbach and Suh [1987] and denoted here as *MTN*.

For a given Cartesian task $\boldsymbol{x} = \boldsymbol{x}_d(t)$, the direct use of $\ddot{\boldsymbol{x}}_k = \ddot{\boldsymbol{x}}_{d,k} = \ddot{\boldsymbol{x}}_d(t_k)$ in the expression of $\boldsymbol{b}$ in (3.11) provides an open-loop solution. In order to reduce the Cartesian tracking error during motion, a stabilizing PD feedback control term is inserted at the task level in $\boldsymbol{b}$, obtaining

$$\boldsymbol{b} = \ddot{\boldsymbol{x}}_{d,k} + \boldsymbol{K}_d(\dot{\boldsymbol{x}}_{d,k} - \dot{\boldsymbol{x}}_k) + \boldsymbol{K}_p(\boldsymbol{x}_{d,k} - \boldsymbol{x}_k) - \boldsymbol{h}_k, \tag{3.12}$$

with $m \times m$ (diagonal) gain matrices $\boldsymbol{K}_d > 0$ and $\boldsymbol{K}_p > 0$.

3.3 Model-based preview of evolution

To prevent a long term unstable behavior of the solution that instantaneously minimizes torque we will include an estimate of the future robot state in the optimization process. At time $t = t_k$, consider a preview window $T = pT_s$, for some integer $p \geq 1$. At the sampling instant $t_{k+p} = t_k + T$, we shall associate a suitable approximation of the robot state. In the following, we shall consider that $T = T_s$ (or $p = 1$), but the same formulas will be used also as an approximated model for a larger $T > 0$.

Suppose that a constant acceleration $\ddot{\boldsymbol{q}}_k$ is applied during a preview window $T = T_s$ in the time interval $[t_k, t_{k+1})$. The following discrete-time evolution holds then exactly for the robot state

$$\begin{aligned} \dot{\boldsymbol{q}}_{k+1} &= \dot{\boldsymbol{q}}(t_{k+1}) = \dot{\boldsymbol{q}}_k + \ddot{\boldsymbol{q}}_k\, T, \\ \boldsymbol{q}_{k+1} &= \boldsymbol{q}(t_{k+1}) = \boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T + \tfrac{1}{2}\, \ddot{\boldsymbol{q}}_k\, T^2. \end{aligned} \tag{3.13}$$

We can indeed associate from (1.32) a unique $\boldsymbol{\tau}_k$ to any $\ddot{\boldsymbol{q}}_k$. Imposing a constant acceleration to the robot during the sampling interval may be difficult, and one may apply instead a constant value $\boldsymbol{\tau}_k$. In this case, joint acceleration would no longer remain constant over sampling intervals, and the resulting expressions (3.13) would become only approximate.

Similarly to (3.10), which is in fact the no preview solution for $T = 0$, we formulate the following optimization problem,

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_{k+1}} \; & H_2 \;= \tfrac{1}{2} \left(\omega_k \|\boldsymbol{\tau}_k\|^2 + \omega_{k+1} \|\boldsymbol{\tau}_{k+1}\|^2 \right) \\ \text{s.t.} \; & \boldsymbol{\tau}_k = \boldsymbol{M}_k \ddot{\boldsymbol{q}}_k + \boldsymbol{n}_k \\ & \ddot{\boldsymbol{x}}_k = \boldsymbol{J}_k \ddot{\boldsymbol{q}}_k + \boldsymbol{h}_k \\ & \boldsymbol{\tau}_{k+1} = \boldsymbol{M}_{k+1} \ddot{\boldsymbol{q}}_{k+1} + \boldsymbol{n}_{k+1} \\ & \ddot{\boldsymbol{x}}_{k+1} = \boldsymbol{J}_{k+1} \ddot{\boldsymbol{q}}_{k+1} + \boldsymbol{h}_{k+1}, \end{aligned} \tag{3.14}$$

where we used the notations

$$\begin{aligned} \boldsymbol{M}_{k+1} &= \boldsymbol{M}(\boldsymbol{q}_{k+1}), \quad \boldsymbol{n}_{k+1} = \boldsymbol{n}(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1}), \\ \boldsymbol{J}_{k+1} &= \boldsymbol{J}(\boldsymbol{q}_{k+1}), \quad \boldsymbol{h}_{k+1} = \boldsymbol{h}(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1}). \end{aligned}$$

For generality, we introduced in (3.14) also the two constants $\omega_k \geq 0$ and $\omega_{k+1} \geq 0$ (with $\omega_k^2 + \omega_{k+1}^2 \neq 0$) that relatively weigh the torque norms at the current and the next instants.

When plugging the expressions coming from eq. (3.13) into the various dynamic terms that need to be evaluated at time $t = t_{k+1}$, the above formulation loses the original structure of a LQ problem in the unknown joint acceleration. This is in fact due to the nonlinear dependence of the inertia matrix, of the Christoffel symbols, and of the gravity vector on $\boldsymbol{q}$, as well as to the quadratic dependence of the Coriolis and centrifugal terms on $\dot{\boldsymbol{q}}$ (see (1.33)), whereas position and velocity at t_{k+1} depend in turn linearly on $\ddot{\boldsymbol{q}}_k$. Therefore, the constraints of the exact formulation (3.14) are still linear in the unknown $\ddot{\boldsymbol{q}}_{k+1}$, but no longer linear in the unknown $\ddot{\boldsymbol{q}}_k$; moreover, the objective function is no longer quadratic in the $\ddot{\boldsymbol{q}}_k$. This makes the problem (3.14) impossible to solve in a closed form, as opposed to (3.10).

On the other hand, by 'freezing' the dependencies of $\boldsymbol{M}_{k+1}$, $\boldsymbol{J}_{k+1}$, $\boldsymbol{n}_{k+1}$, and $\boldsymbol{h}_{k+1}$ to their value at time $t = t_k$, or even by just removing the dependence of these terms from $\ddot{\boldsymbol{q}}_k$, would make the formulation (3.14) *separable* in two independent sub-problems, one depending only on $\ddot{\boldsymbol{q}}_k$ and the other only on $\ddot{\boldsymbol{q}}_{k+1}$. Thus, the benefit of linking the decision on which is the optimal choice for the current acceleration to the resulting effect and similar decision at the preview instant would be completely lost (i.e., the whole sense of resorting to a preview).

As a result, for a more practical formulation that would lead to an effective solution of a joint LQ problem, we shall:

1. keep the same constraints at the current instant, to guarantee that $\ddot{\boldsymbol{q}}_k$ realizes the task at $t = t_k$;
2. preserve a forward coupling between the current acceleration and the data/command at the preview instant, allowing a linear dependence of the constraints and a quadratic dependence of the objective function on $\ddot{\boldsymbol{q}}_k$.

Consider then again the task constraint (1.29). We approximate it at time $t = t_{k+1}$ as follows:

$$\begin{aligned}
\ddot{\boldsymbol{x}}_{k+1} &= \boldsymbol{J}(\boldsymbol{q}_{k+1})\ddot{\boldsymbol{q}}_{k+1} + \dot{\boldsymbol{J}}(\boldsymbol{q}_{k+1})\dot{\boldsymbol{q}}_{k+1} \\
&\approx \boldsymbol{J}(\boldsymbol{q}_{k+1})\ddot{\boldsymbol{q}}_{k+1} + \frac{\boldsymbol{J}(\boldsymbol{q}_{k+1}) - \boldsymbol{J}(\boldsymbol{q}_k)}{T}\,\dot{\boldsymbol{q}}_{k+1} \\
&\approx \boldsymbol{J}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T)\ddot{\boldsymbol{q}}_{k+1} + \frac{\boldsymbol{J}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T) - \boldsymbol{J}(\boldsymbol{q}_k)}{T}\,\dot{\boldsymbol{q}}_{k+1} \\
&= \boldsymbol{J}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T)\ddot{\boldsymbol{q}}_{k+1} + \frac{\boldsymbol{J}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T) - \boldsymbol{J}(\boldsymbol{q}_k)}{T}\,(\dot{\boldsymbol{q}}_k + T\ddot{\boldsymbol{q}}_k) \\
&= \boldsymbol{J}_{k^+}\ddot{\boldsymbol{q}}_{k+1} + (\boldsymbol{J}_{k^+} - \boldsymbol{J}_k)\,\ddot{\boldsymbol{q}}_k + \boldsymbol{h}_{k^+},
\end{aligned} \tag{3.15}$$

with the notation

$$\begin{aligned}
\boldsymbol{J}_{k^+} &= \boldsymbol{J}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T), \\
\boldsymbol{h}_{k^+} &= \frac{\boldsymbol{J}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T) - \boldsymbol{J}(\boldsymbol{q}_k)}{T}\,\dot{\boldsymbol{q}}_k = \frac{\boldsymbol{J}_{k^+} - \boldsymbol{J}_k}{T}\,\dot{\boldsymbol{q}}_k.
\end{aligned}$$

Next, consider the squared norm of the torque at time $t = t_{k+1}$ and its expression through the dynamic model (1.32). Taking into account the factorization (1.34) of quadratic velocity terms, we proceed with the following approximation:

$$\begin{aligned}
\|\boldsymbol{\tau}_{k+1}\|^2 &= \|\boldsymbol{M}(\boldsymbol{q}_{k+1})\ddot{\boldsymbol{q}}_{k+1} + \boldsymbol{c}(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1}) + \boldsymbol{g}(\boldsymbol{q}_{k+1})\|^2 \\
&= \ddot{\boldsymbol{q}}_{k+1}^T \boldsymbol{M}_{k+1}^2 \ddot{\boldsymbol{q}}_{k+1} + 2\left(\boldsymbol{c}_{k+1} + \boldsymbol{g}_{k+1}\right)^T \boldsymbol{M}_{k+1}\ddot{\boldsymbol{q}}_{k+1} \\
&\quad + \left(\boldsymbol{c}_{k+1} + \boldsymbol{g}_{k+1}\right)^T \left(\boldsymbol{c}_{k+1} + \boldsymbol{g}_{k+1}\right) \\
&\approx \ddot{\boldsymbol{q}}_{k+1}^T \boldsymbol{M}_{k^+}^2\, \ddot{\boldsymbol{q}}_{k+1} + 2\left(\boldsymbol{S}_{k^+}\dot{\boldsymbol{q}}_{k+1} + \boldsymbol{g}_{k^+}\right)^T \boldsymbol{M}_{k^+}\, \ddot{\boldsymbol{q}}_{k+1} \\
&\quad + \dot{\boldsymbol{q}}_{k+1}^T \boldsymbol{S}_{k^+}^T \boldsymbol{S}_{k^+}\dot{\boldsymbol{q}}_{k+1} + 2\,\boldsymbol{g}_{k^+}^T \boldsymbol{S}_{k^+}\dot{\boldsymbol{q}}_{k+1} + \boldsymbol{g}_{k^+}^T \boldsymbol{g}_{k^+} \\
&= \ddot{\boldsymbol{q}}_{k+1}^T \boldsymbol{M}_{k^+}^2\, \ddot{\boldsymbol{q}}_{k+1} + 2\left(\boldsymbol{S}_{k^+}\left(\dot{\boldsymbol{q}}_k + T\ddot{\boldsymbol{q}}_k\right) + \boldsymbol{g}_{k^+}\right)^T \boldsymbol{M}_{k^+}\, \ddot{\boldsymbol{q}}_{k+1} \\
&\quad + \left(\dot{\boldsymbol{q}}_k + T\ddot{\boldsymbol{q}}_k\right)^T \boldsymbol{S}_{k^+}^T \boldsymbol{S}_{k^+}\left(\dot{\boldsymbol{q}}_k + T\ddot{\boldsymbol{q}}_k\right) + 2\,\boldsymbol{g}_{k^+}^T \boldsymbol{S}_{k^+}\left(\dot{\boldsymbol{q}}_k + T\ddot{\boldsymbol{q}}_k\right) \\
&\quad + \boldsymbol{g}_{k^+}^T \boldsymbol{g}_{k^+},
\end{aligned} \tag{3.16}$$

with the notation

$$\begin{aligned}&\boldsymbol{M}_{k^+} = \boldsymbol{M}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T), \quad \boldsymbol{g}_{k^+} = \boldsymbol{g}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T),\\ &\boldsymbol{S}_{k^+} = \boldsymbol{S}(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T, \dot{\boldsymbol{q}}_k) = \mathrm{col}\{\dot{\boldsymbol{q}}_k^T \boldsymbol{C}_i(\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k T)\}.\end{aligned}$$

With these expressions at hand which satisfy our desired guidelines, we finally replace the original nonlinear optimization problem (3.14) with the following LQ approximation:

$$\begin{aligned}\min_{\ddot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_{k+1}} \; & H_2 \; = \tfrac{1}{2}\left(\omega_k \|\boldsymbol{\tau}_k\|^2 + \omega_{k+1}\|\boldsymbol{\tau}_{k+1}\|^2\right)\\ \text{s.t.} \; & \boldsymbol{\tau}_k = \boldsymbol{M}_k \ddot{\boldsymbol{q}}_k + \boldsymbol{n}_k\\ & \ddot{\boldsymbol{x}}_k = \boldsymbol{J}_k \ddot{\boldsymbol{q}}_k + \boldsymbol{h}_k\\ & \boldsymbol{\tau}_{k+1} = \boldsymbol{M}_{k^+} \ddot{\boldsymbol{q}}_{k+1} + \boldsymbol{S}_{k^+}\left(\dot{\boldsymbol{q}}_k + T\ddot{\boldsymbol{q}}_k\right) + \boldsymbol{g}_{k^+}\\ & \ddot{\boldsymbol{x}}_{k+1} = \boldsymbol{J}_{k^+} \ddot{\boldsymbol{q}}_{k+1} + \left(\boldsymbol{J}_{k^+} - \boldsymbol{J}_k\right)\ddot{\boldsymbol{q}}_k + \boldsymbol{h}_{k^+}.\end{aligned} \tag{3.17}$$

In particular, replacing the given expressions of the two torques in the objective function, the latter takes the form

$$H_2 = \tfrac{1}{2}\begin{pmatrix} \ddot{\boldsymbol{q}}_k & \ddot{\boldsymbol{q}}_{k+1}\end{pmatrix}^T \boldsymbol{Q}\begin{pmatrix}\ddot{\boldsymbol{q}}_k\\ \ddot{\boldsymbol{q}}_{k+1}\end{pmatrix} + \boldsymbol{r}^T\begin{pmatrix}\ddot{\boldsymbol{q}}_k\\ \ddot{\boldsymbol{q}}_{k+1}\end{pmatrix} + s, \tag{3.18}$$

with

$$\begin{aligned}\boldsymbol{Q} &= \begin{pmatrix}\omega_k \boldsymbol{M}_k^2 + \omega_{k+1} T^2 \boldsymbol{S}_{k^+}^T \boldsymbol{S}_{k^+} & \omega_{k+1} T \boldsymbol{S}_{k^+}^T \boldsymbol{M}_{k^+}\\ \text{symm} & \omega_{k+1}\boldsymbol{M}_{k^+}^2\end{pmatrix},\\ \boldsymbol{r} &= \begin{pmatrix}\omega_k \boldsymbol{M}_k(\boldsymbol{S}_k \dot{\boldsymbol{q}}_k + \boldsymbol{g}_k) + \omega_{k+1} T \boldsymbol{S}_{k^+}^T\left(\boldsymbol{S}_{k^+}\dot{\boldsymbol{q}}_k + \boldsymbol{g}_{k^+}\right)\\ \omega_{k+1}\boldsymbol{M}_{k^+}\left(\boldsymbol{S}_{k^+}\dot{\boldsymbol{q}}_k + \boldsymbol{g}_{k^+}\right)\end{pmatrix},\end{aligned} \tag{3.19}$$

and the scalar (irrelevant for the optimization)

$$s = \frac{\omega_k}{2}\boldsymbol{n}_k^T \boldsymbol{n}_k + \frac{\omega_{k+1}}{2}\left(\boldsymbol{S}_{k^+}\dot{\boldsymbol{q}}_k + \boldsymbol{g}_{k^+}\right)^T\left(\boldsymbol{S}_{k^+}\dot{\boldsymbol{q}}_k + \boldsymbol{g}_{k^+}\right).$$

Similarly, the task constraint in (3.17) can be rewritten in matrix format as

$$\boldsymbol{A}\begin{pmatrix}\ddot{\boldsymbol{q}}_k\\ \ddot{\boldsymbol{q}}_{k+1}\end{pmatrix} = \boldsymbol{b}, \tag{3.20}$$

where

$$\boldsymbol{A} = \begin{pmatrix}\boldsymbol{J}_k & \boldsymbol{0}\\ \boldsymbol{J}_{k^+} - \boldsymbol{J}_k & \boldsymbol{J}_{k^+}\end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix}\ddot{\boldsymbol{x}}_k - \boldsymbol{h}_k\\ \ddot{\boldsymbol{x}}_{k+1} - \boldsymbol{h}_{k^+}\end{pmatrix}. \tag{3.21}$$

Provided that matrix $\boldsymbol{Q}$ in (3.19) is positive definite (see Appendix C) and matrix $\boldsymbol{A}$ has full (row) rank equal to $2m$, the solution to (3.17) has the closed-form expression (3.8) when using in (3.1) the substitutions (3.19), (3.21), and

$$\begin{aligned}\ddot{\boldsymbol{q}}^* &= (\ddot{\boldsymbol{q}}_k \;\; \ddot{\boldsymbol{q}}_{k+1})^T,\\ \|\boldsymbol{\tau}^*\|^2 &= \omega_k\|\boldsymbol{\tau}_k\|^2 + \omega_{k+1}\|\boldsymbol{\tau}_{k+1}\|^2.\end{aligned} \tag{3.22}$$

This model-based preview solution will be denoted as *MBP*. In the implementation, $\ddot{\boldsymbol{q}}_{k+1}$ is discarded and only $\ddot{\boldsymbol{q}}_k$ from (3.22) will be used at the instant $t = t_k$. In this case, a feedback control action in the form (3.12) will be added only inside the term $\ddot{\boldsymbol{x}}_k$ in (3.21), whereas we shall keep $\ddot{\boldsymbol{x}}_{k+1} = \ddot{\boldsymbol{x}}_{d,k+1}$ for the preview instant.

Inclusion of dynamic damping in the null space

In problems (3.10) and (3.14), the joint torque is optimized to be in norm as close as possible to zero. Instead, one could minimize the norm of the difference with respect to a suitable desired target torque. Define

$$\boldsymbol{\tau}_{D_k} = -\boldsymbol{D}_k \boldsymbol{M}_k \dot{\boldsymbol{q}}_k, \tag{3.23}$$

where $\boldsymbol{D}_k$ is a non-negative diagonal (damping) gain matrix and $\boldsymbol{M}_k\dot{\boldsymbol{q}}_k$ is the generalized momentum of the robot at $t = t_k$. When $\boldsymbol{\tau}_k = \boldsymbol{\tau}_{D_k}$ and the damping matrix is in the form $\boldsymbol{D}_k = d\boldsymbol{I} > 0$, from (1.32) and (1.34) the joint acceleration $\ddot{\boldsymbol{q}}_k$ becomes

$$\ddot{\boldsymbol{q}}_k = -\boldsymbol{M}_k^{-1}(\boldsymbol{S}_k\dot{\boldsymbol{q}}_k + \boldsymbol{g}_k) - d\dot{\boldsymbol{q}}_k. \tag{3.24}$$

Observing eq. (3.24), the effect of the target torque (3.23) is always to work against the current joint velocity, acting as a damper on the joint motion of the robot manipulator. This will reduce, or even eliminate, whipping effects that could happen when the joint velocity becomes too large.

In this case, the problem (3.10) is modified so as to minimize the objective function

$$\begin{aligned} H_3 &= \tfrac{1}{2}\|\boldsymbol{\tau}_k - \boldsymbol{\tau}_{D_k}\|^2 \\ &= \tfrac{1}{2}\|\boldsymbol{M}_k\ddot{\boldsymbol{q}}_k + (\boldsymbol{S}_k + \boldsymbol{D}_k\boldsymbol{M}_k)\dot{\boldsymbol{q}}_k + \boldsymbol{g}_k\|^2. \end{aligned} \tag{3.25}$$

The solution is obtained by substituting in (3.8)

$$\begin{aligned} \ddot{\boldsymbol{q}}^* = \ddot{\boldsymbol{q}}_k, \quad \boldsymbol{Q} = \boldsymbol{M}_k^2, \quad \boldsymbol{r} = \boldsymbol{M}_k((\boldsymbol{S}_k + \boldsymbol{D}_k\boldsymbol{M}_k)\dot{\boldsymbol{q}}_k + \boldsymbol{g}_k), \\ \boldsymbol{A} = \boldsymbol{J}_k, \quad \boldsymbol{b} = \ddot{\boldsymbol{x}}_k - \boldsymbol{h}_k. \end{aligned} \tag{3.26}$$

This solution will be denoted as *MTND* (i.e., *MTN* with damping). The only difference between the expressions (3.11) and (3.26), respectively in the *MTN* and *MTND* solutions, is in the $\boldsymbol{r}$ term, namely in an additional damping effect appearing in the null space of the task Jacobian.

In the case of a torque optimization with model-based preview (3.17), in place of (3.25) we consider the objective function

$$H_4 = \tfrac{1}{2}\left(\omega_k\|\boldsymbol{\tau}_k - \boldsymbol{\tau}_{D_k}\|^2 + \omega_{k+1}\|\boldsymbol{\tau}_{k+1} - \boldsymbol{\tau}_{D_{k^+}}\|^2\right), \tag{3.27}$$

where $\boldsymbol{\tau}_{D_{k^+}} = -\boldsymbol{D}_{k+1}\boldsymbol{M}_{k^+}(\dot{\boldsymbol{q}}_k + T\ddot{\boldsymbol{q}}_k)$. The resulting optimal solution $\ddot{\boldsymbol{q}}^* = (\ddot{\boldsymbol{q}}_k \;\; \ddot{\boldsymbol{q}}_{k+1})^T$ is again described by (3.8), using the substitutions (3.21) for $\boldsymbol{A}$ and $\boldsymbol{b}$. For $\boldsymbol{Q}$ and $\boldsymbol{r}$, the same expressions (3.19) can be used, after replacing the $\boldsymbol{S}_k$ term in $\boldsymbol{r}$ with $(\boldsymbol{S}_k + \boldsymbol{D}_k\boldsymbol{M}_k)$ and each $\boldsymbol{S}_{k^+}$ in $\boldsymbol{r}$ and $\boldsymbol{Q}$ with $(\boldsymbol{S}_{k^+} + \boldsymbol{D}_{k+1}\boldsymbol{M}_{k^+})$. This solution is denoted as *MBPD*. A feedback control action on the task error is included as in the undamped cases.

3.4 Trajectory planning

In Chapter 2, the trajectory planning done using TOPP in the s parameter space to control the joints motion in minimum time. While here we would like to control the end effector motion according to desired Cartesian velocity and

acceleration to achieve the task in a specific time. This is also can be done in the s space Siciliano et al. [2008]. Suppose that the desired positional Cartesian task is defined as

$$\boldsymbol{x} = \boldsymbol{x}(s), \quad s \in [0, s_f], \qquad s = s(t), \quad t \in [0, t_f]. \tag{3.28}$$

Differentiating (3.28) once and twice with respect to time yields

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{x}'\dot{s}, \\ \ddot{\boldsymbol{x}} &= \boldsymbol{x}'\ddot{s} + \boldsymbol{x}''\dot{s}^2. \end{aligned} \tag{3.29}$$

where a dot $\dot{(\,)}$ and a prime $(\,)'$ denote same as in (2.2). Let the s parameter defined as

$$s(t) = \frac{\sigma(t)}{l}, \quad \sigma \in [0, l], \tag{3.30}$$

where l is the arc length of the desired Cartesian path. This leads to $s_{end} = 1$ where $\sigma(t)$ gives the current length of the path. To plan a rest-to-rest motion while following the path with trapezoidal speed profile, the $\sigma(t)$ can be defined as

$$\sigma(t) = \begin{cases} \frac{1}{2}a_{\max}t^2, & t \in [0, t_s], \\ v_{\max}t - \frac{v_{\max}^2}{2a_{\max}}, & t \in [t_s, t_f - t_s], \\ -\frac{1}{2}a_{\max}(t - t_f)^2 + v_{\max}t_f - \frac{v_{\max}^2}{a_{\max}}, & t \in [t_f - t_s, t_f], \end{cases} \tag{3.31}$$

where the desired maximum Cartesian velocity and acceleration are v_{max} and a_{max} respectively. t_s is the time needed to reach the maximum/zero velocity applying the maximum acceleration/deceleration. In this case, t_s and t_{end} are computed as

$$\begin{aligned} t_s &= \frac{v_{\max}}{a_{\max}}, \\ t_f &= \frac{la_{\max} + v_{\max}^2}{a_{\max}v_{\max}}. \end{aligned} \tag{3.32}$$

Note that the coast phase in the acceleration ($\ddot{s} = \ddot{\sigma} = 0$) exists only if $l > \frac{v_{\max}^2}{a_{\max}}$. From (3.31) and (3.30), the first and second s derivatives in (3.29) are computed as

$$\dot{s} = \dot{\sigma} = \begin{cases} a_{\max}t, & t \in [0, t_s], \\ v_{\max}, & t \in [t_s, t_f - t_s], \\ a_{\max}(t_f - t), & t \in [t_f - t_s, t_f], \end{cases} \tag{3.33}$$

$$\ddot{s} = \ddot{\sigma} = \begin{cases} a_{\max}, & t \in [0, t_s], \\ 0, & t \in [t_s, t_f - t_s], \\ -a_{\max}, & t \in [t_f - t_s, t_f]. \end{cases}$$

To perform a rest-to-rest motion using the different introduced torque optimization methods, the desired Cartesian profiles in (3.12) are computed using the equations (3.28) and (3.29).

3.5 Results

We report here various simulations and experiments performed to illustrate the comparison between the different introduced inverse differential solutions for torque optimization. Three study cases are considered: a 3R planar arm and the UR10 manipulator in simulations, and a 7R KUKA LWR 4 lightweight robot in simulations and experiments.

3R planar arm

To illustrate the comparison between the different solutions for local torque optimization, we have considered the same case study presented in Hollerbach and Suh [1987]. The 3R planar arm ($n = 3$) has links of equal length $l = 1$ [m], uniformly distributed mass $m_l = 10$ [kg] and moment of inertia $I_l = m_l l^2/12$. The end-effector position ($m = 2$) should follow a linear path of short length ($L_1 = 0.2828$ [m]), or one four times longer ($L_2 = 1.1738$ [m]). The path is traced with a rest-to-rest timing law having bang-bang acceleration of magnitude $a_{\max} = \sqrt{2} = 1.4142$ [m/s^2]. The robot starts with $\dot{\boldsymbol{q}}(0) = \boldsymbol{0}$. The degree of robot redundancy is $n - m = 1$ in this case.

The methods have been implemented in MATLAB with a fixed integration step $T_s = 0.001$ [s]. The *MBP* and *MBPD* methods use equal weights $\omega_k = \omega_{k+1} = 1$ in the objective function. The joint damping matrix is chosen as $\boldsymbol{D}_k = 10\,\boldsymbol{I}$, constant at all discrete times for both *MTND* and *MBPD* methods. In all methods, the feedback gains on the Cartesian task error were $\boldsymbol{K}_p = 10\,\boldsymbol{I}$ and $\boldsymbol{K}_d = \boldsymbol{I}$. The short preview window can be chosen directly as the next sampling instant, i.e., $T = T_s$. More in general, the optimization process can be used to minimize instead the norm of the torques at the current instant and at any other instant in the short future, i.e., $T > T_s$. For longer T, it is expected that the accuracy of the robot state estimation decreases. When $T = 0$, the *MBP* method collapses into the original *MTN* one.

Figure 3.1 shows stroboscopic views for the four addressed methods along the short and the long path, respectively. In this case, the preview window was $T = 100T_s = 0.1$ [s] for the *MBP* and *MBPD* methods. All optimization methods performed in the same way on the short path, with a rather symmetric behavior with respect to the half-motion time as shown in Fig. 3.2. All methods completed the task with zero final joint velocities in practice, although this was not an explicit constraint in the optimization problem.

Figure 3.3 shows the norms of the joint velocity and torque on the longer path for the different methods. While a common behavior is found in the first half of the motion, i.e., until $t_s = \sqrt{L_2/a_{\max}} \approx 0.91$ [s], each method completes the task in a different way during the deceleration phase. With the *MTN* method, it is clear how the contentious increase of the joint velocities leads to the sudden and high peak in the joint torques near the end of the task. Using instead our proposed methods this undesired behavior is eliminated, with the *MBP* method having the minimum values for the joint torques norm. The *MTND* and *MBPD* methods produced almost the same behavior, with the joint velocities vanishing at the end of the task. The Cartesian task error was in all cases negligible, in the order of 10^{-4} [m].

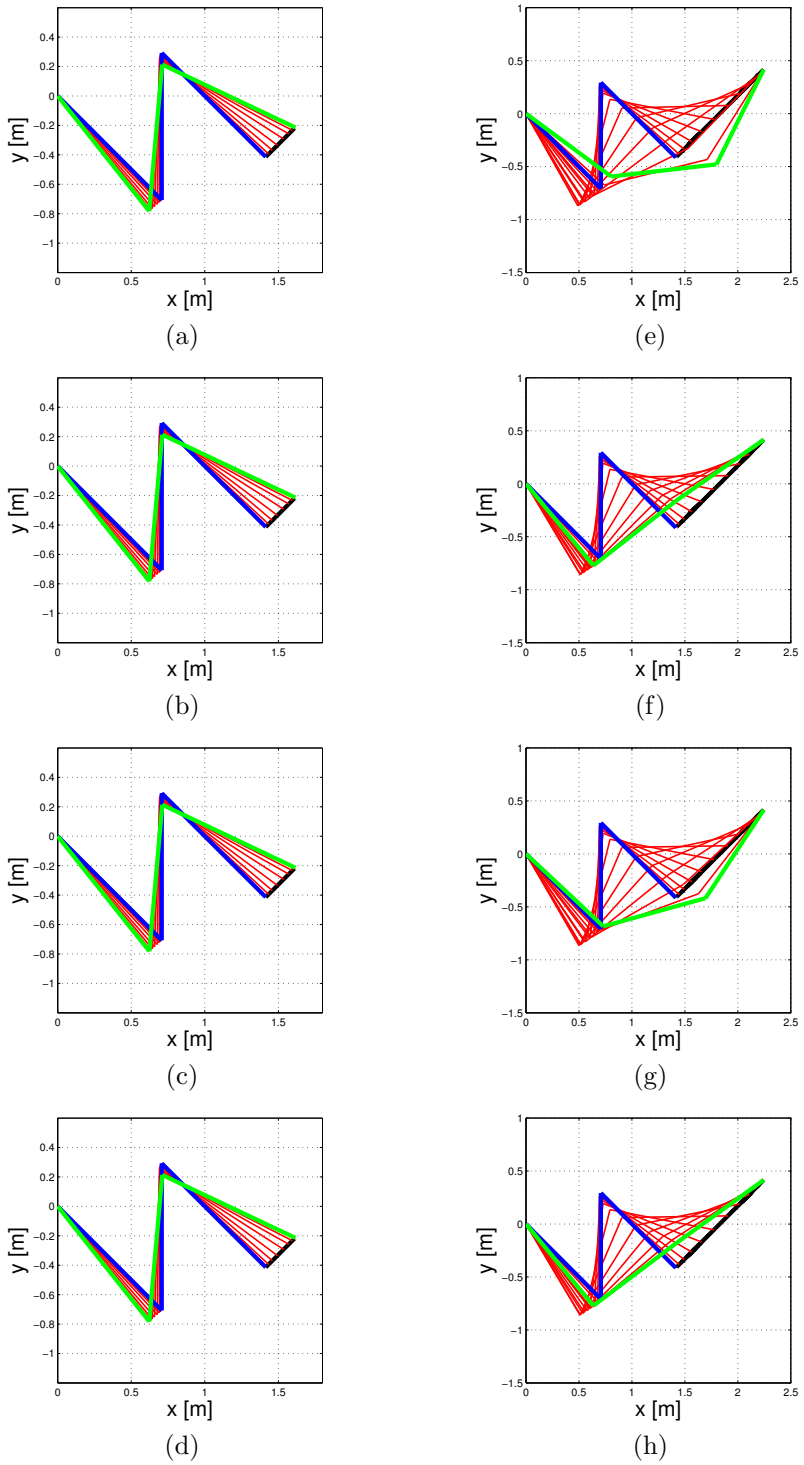


Figure 3.1: Stroboscopic motion of the 3R planar arm on a short [left] and a long [right] Cartesian path using different torque optimization methods: (a-e) *MTN*; (b-f) *MTND*; (c-g) *MBP*; (d-h) *MBPD*. The desired linear paths are in black. Initial and final arm configurations are shown in blue and green, respectively.

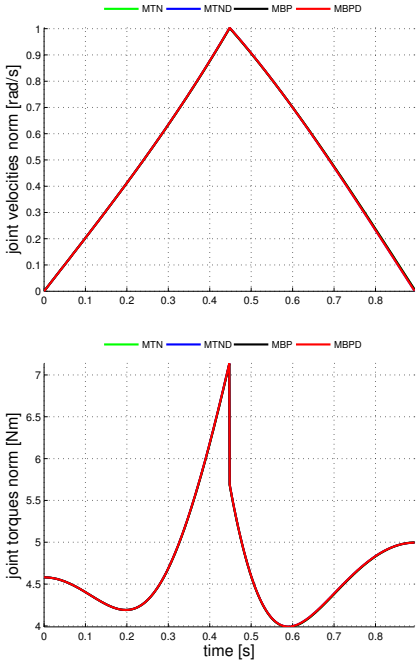


Figure 3.2: Joint velocity norms [top] and torque norms [bottom] for the 3R planar arm using different solutions on the short trajectory.

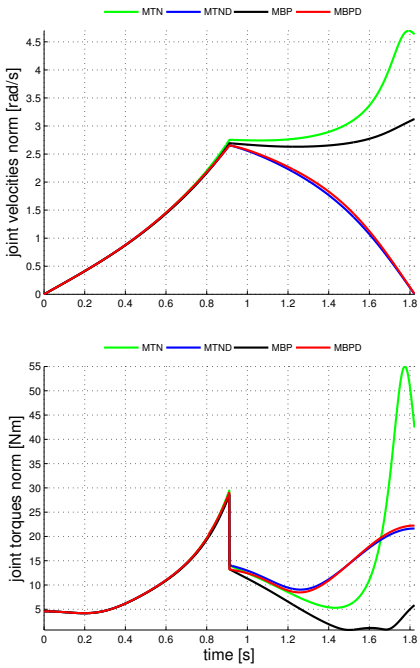


Figure 3.3: Joint velocity norms [top] and torque norms [bottom] for the 3R planar arm using different solutions on the long trajectory.

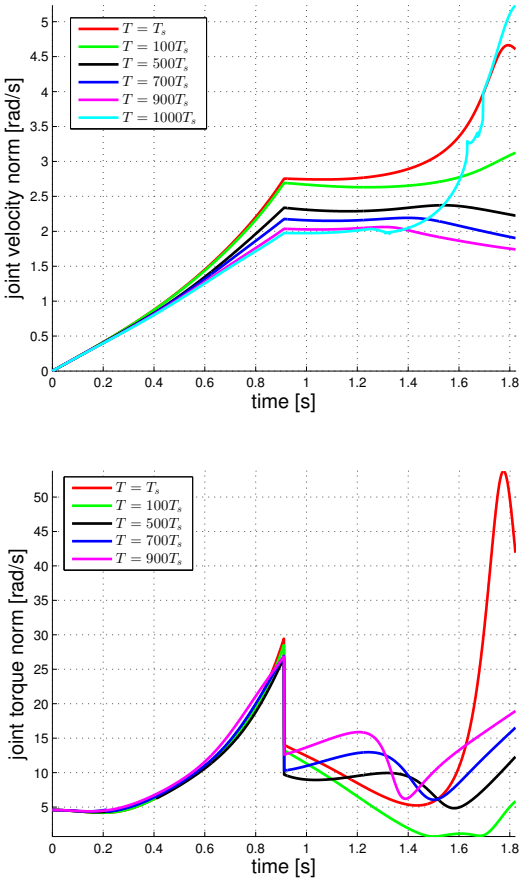


Figure 3.4: Joint velocity norms [top] and torque norms [bottom] for the 3R planar arm on the long trajectory with the *MBP* method using different preview windows T. Simulations run at $T_s = 1$ [ms]. The torque norm for $T = 1000T_s = 1$ [s] is not shown being too large.

Figure 3.4 shows the joint velocity and torque norms obtained in multiple simulations on the long linear task, when using the *MBP* solution with different preview values T. Too short values, e.g., $T = T_s = 1$ [ms], had no practical effect and returned the same bad behavior of the *MTN* method. For an intermediate set of preview values, i.e., $T \in [0.1, 0.9]$ s, the robot loses the undesired torque peak and the behaviors will be similarly good. For the *MBP* method, a preview of $T = 100\,T_s = 0.1$ [s] gives the best result in terms of torque norm, while $T = 900\,T_s = 0.9$ s gives the best result for the joint velocity norm. In general, joint velocities become lower and torques higher when increasing T, until a limit is reached where the peaks appear back (here, for $T = 1000\,T_s = 1$ [s]) and robot motion is unacceptable again. Indeed, the best choice for T will depend on the desired trajectory and on the dynamics of the specific robot. Nonetheless, for a given robot/trajectory pair, the existence of an interval of good performance for the preview method appears to be robust.

Table 3.1: Performance indices of the UR10 simulation along circular task using different torque optimization schemes.

Performance index [Nm]	*MTN*	*MTND*	*MBP*	*MBPD*
Mean torque norm	**63.5**	63.0	63.2	63.1
Peak torque norm	**63.9**	63.7	63.7	63.7

UR10 manipulator

The second case study considers simulations with the UR10 manipulator, see Appendix A. The robot system has to follow a Cartesian task defined only for the position of the end-effector ($m = 3$). The final joint was frozen, since it has no effect on the task, resulting in only $n = 5$ active joints, with a redundancy degree $n - m = 2$. In all computations, the robot dynamic model presented in Gaz et al. [2018] has been used. The simulations are performed using C++ and V-REP with a sampling time $T_s = 10$ [ms].

The positional task is consist of a half circular path of radius $R = 0.6$ $[m]$ for an arc-length of $l_c = 1.885$ [m] to be traced in $t_f = 10.4$ $[s]$. The desired maximum Cartesian velocity and acceleration are $v_{\max} = 0.2$ $[m/s]$ and $a_{\max} = 0.2$ $[m/s^2]$ respectively. For illustration, the timing law for the desired task is shown in Fig. 3.5. The damping matrix was $\boldsymbol{D}_k = 10\boldsymbol{I}$ for *MTND* method, while for the *MBPD* method $\boldsymbol{D}_k = \boldsymbol{D}_{k+1} = \boldsymbol{I}$. For both *MBP* and *MBPD* methods, the weighting parameters in the objective function were $\omega_k = \omega_{k+1} = 1$.

Figure 3.6 shows the joint velocity norm corresponding to multiple simulations of the task using the *MBP* solution with different preview values T. The joint velocity norm becomes smoother for increasing values T, until a specific limit is reached ,(in this case $T = 60T_s$), where instability appears along the whole path in shape of high frequency oscillations of velocity.

Using $T = 50T_s$ for *MBP* and *MBPD* methods, the evolutions of the torque norms in Fig. 3.8 are almost constant along the path for all methods, i.e., the difference between the torque norms is less than 1 [Nm]. However, the *MTN* method suffers from frequent torque oscillations. The peaks in torque norms close to the beginning and ending of the task are because of the acceleration discontinuity in the time law, see Fig. 3.5. From Tab. 3.1, the *MTN* method returns the maximum mean torque norms over the entire motion, i.e., $\frac{1}{t_f}\int_0^{t_f} \|\boldsymbol{\tau}\|_2 \, dt$. Figure 3.7 shows quite clearly the rapid oscillatory behavior of the joint velocities generated by the original *MTN* method during the most part of the motion. Using *MTND* method, the velocity oscillations are eliminated but a high peak still appear at the beginning and ending of the task. While the *MBP* and *MBPD* methods eliminate any undesired behavior.

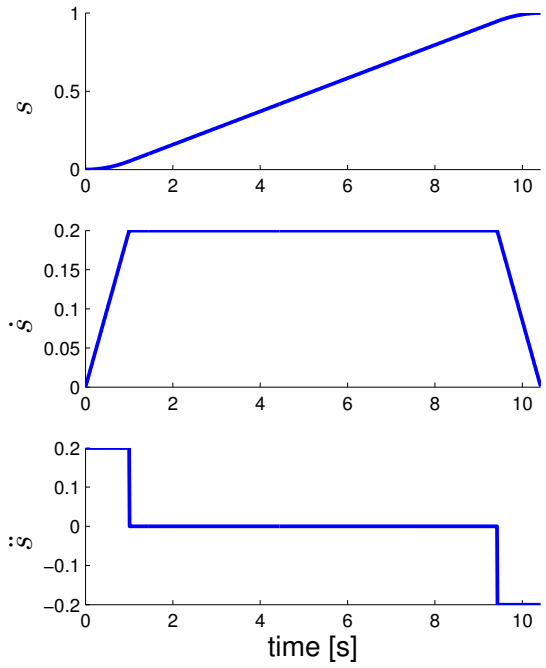


Figure 3.5: Timing law for the circular task with UR10 robot. It prescribes a rest-to-rest motion while following the path with trapezoidal speed profile.

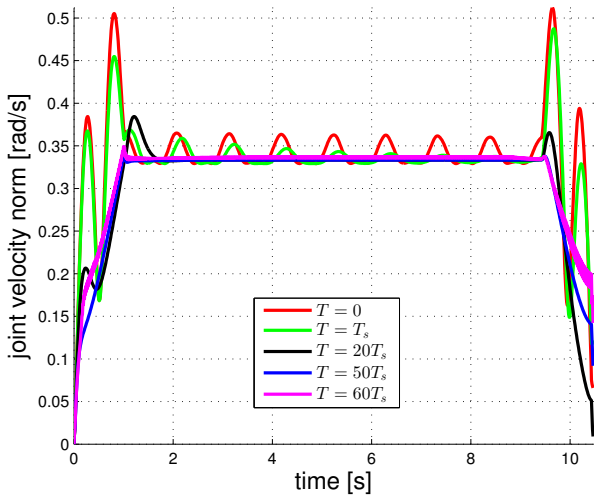


Figure 3.6: Joint velocity norms for the UR10 robot using the *MBP* solution with different time previews (simulation along circular path with $T_s = 0.01$ [s]).

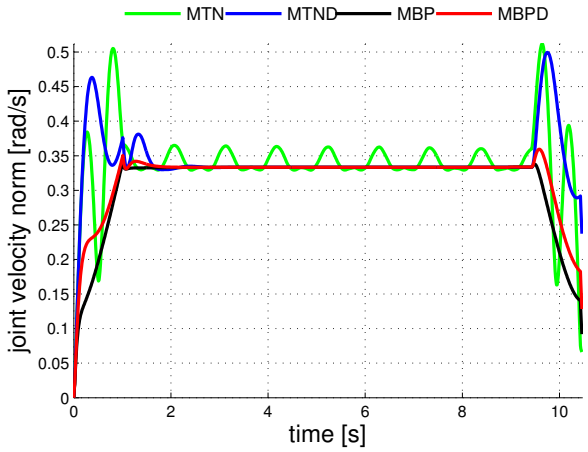


Figure 3.7: Joint velocity norms for the UR10 robot using different optimization solutions along circular path in simulations.

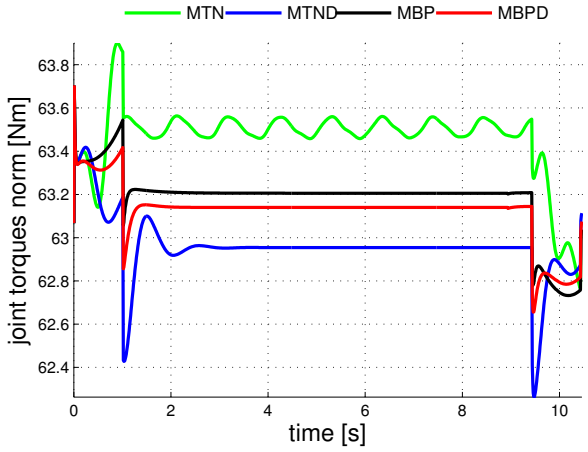


Figure 3.8: Joint torque norms for the UR10 robot using different optimization solutions along circular path in simulations.

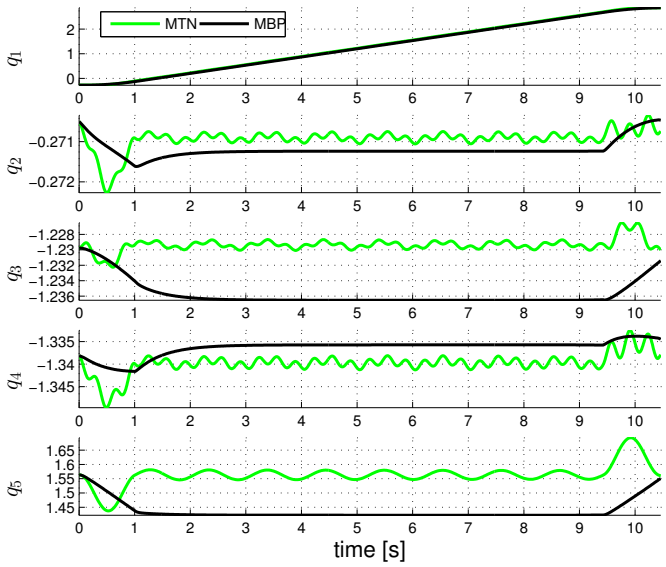


Figure 3.9: Joint positions for the UR10 robot using *MTN* and *MBP* solutions along circular path in simulations.

KUKA LWR IV

As a third case study, we considered a 7R KUKA LWR robot, in Appendix B, and performed simulations using V-REP and our C++ code. Next, for the 7R KUKA LWR 4 robot in our lab, we considered as task the execution of the same positional Cartesian tasks defined in simulations. This is mainly to compare the performance and results between simulations and real experiments. For this, all model-based computations were done using the accurate dynamic model identified in Gaz et al. [2014].

Simulations

The robot task is to follow two Cartesian trajectories defined for the position of the end-effector flange center ($m = 3$). Since the rotation of joint 7 has no effect on the task, the final flange was frozen resulting in only $n = 6$ active joints for the robot, and thus with a degree of task redundancy equal to $n - m = 3$. The timing law prescribes a rest-to-rest motion on the path with a trapezoidal speed profile. The robot starts with $\dot{\boldsymbol{q}}(0) = \mathbf{0}$. In the integration routine, we used $T_s = 10$ [ms] as the fixed step. This value is also the lowest refresh rate for V-REP.

The first motion task is on a relatively short linear path of length $l = 0.5$ [m] to be traced in $t_f = 3.5$ [s]. The second task is longer and consists in following a circular path of radius $R = 0.2$ [m] for an arc length $l_c = 1.256$ [m] and a motion time $t_f = 7.28$ [s]. For both tasks, the desired maximum Cartesian velocity and acceleration were chosen as $v_{\max} = 0.2$ [m/s] and $a_{\max} = 0.2$ [m/s^2], respectively. The weights in the objective function of both *MBP* and *MBPD*

Table 3.2: Mean joint torques norm in [Nm] for the KUKA LWR task along linear path using different torque optimization schemes.

Platform	*MTN*	*MTND*	*MBP*	*MBPD*
V-REP simulation	**35.9**	35.8	**35.9**	35.8
Lab. experiment	36.3	**36.6**	36.4	**36.6**

Table 3.3: Mean joint torques norm in [Nm] for the KUKA LWR task along circular path using different torque optimization schemes.

Platform	*MTN*	*MTND*	*MBP*	*MBPD*
V-REP simulation	34.1	34.1	34.1	34.1
Lab. experiment	35.0	35.0	35.0	35.0

methods were $\omega_k = \omega_{k+1} = 1$. The damping matrix was constantly $\boldsymbol{D}_k = 5\,\boldsymbol{I}$ for the *MTND* and *MBPD* methods, while $\boldsymbol{D}_{k+1} = \boldsymbol{I}$ was used in the *MBPD* method. In all methods, the feedback gains on the Cartesian task error were $\boldsymbol{K}_p = 300\,\boldsymbol{I}$ and $\boldsymbol{K}_d = 50\,\boldsymbol{I}$.

The simulation environment allowed us to test again the performance of the *MBP* method for different preview windows T. Figure. 3.10 shows in particular the behavior of the joint velocity norm along the linear path. As expected, large oscillations are found for $T = 0$ (the *MTN* method). Between $T = T_s = 10$ [ms] and $T = 10\,T_s = 100$ [ms], the *MBP* method works fine. For $T = 15\,T_s = 150$ [ms] or larger, the joint velocity norm becomes unacceptable again, i.e., much larger and oscillatory in nature.

Using a preview window $T = T_s = 10$ [ms]. Figures (3.11–3.13)(a) show the results obtained in the simulations along the linear path. The fast oscillatory behavior of the joint velocity norm generated by the *MTN* method during most part of the motion is quite evident in Fig. 3.12(a). More specifically, this instability appears mostly in the fifth joint position, but also in the third and sixth joint profiles (see Fig. 3.13(a)). It should be noted that the evolutions of the torque norms in Fig. 3.11(a) differ only slightly between the four methods, and the mean torque norms over the entire motion are all very similar, see Tab. 3.2.

In contrast, the proposed *MBP* and *MBPD* methods eliminate any undesired behavior, both in the joint velocities and in the torques. Both damped methods achieve better results in forcing the joint velocities toward zero at the end. On the other hand, the *MTND* method has still some residual oscillations in the joint velocity.

Figures (3.14–3.16)(a) reports the results for the simulations along the circular path. The third joint position in Fig. 3.16(a) experiences the main oscillations in this case, while Fig. 3.15(a) shows a clear increasing trend of these in the joint velocity norm when using the *MTN* method. The *MBP*, *MBPD* and *MTND* methods eliminate any undesired behavior. Similarly to the linear task, the evolutions of the torque norms differ again very slightly between the four methods in Fig. 3.14(a), and the mean torque norms over the entire motion are equal, see Tab. 3.3.

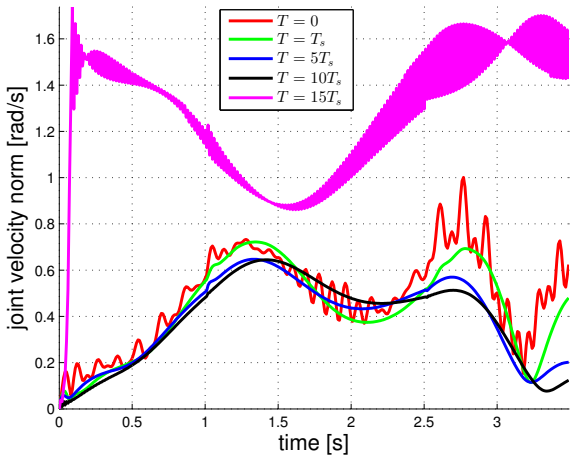


Figure 3.10: Joint velocity norms for the KUKA LWR using the *MBP* solution with different time previews (simulation along a linear path with $T_s = 10$ [ms]).

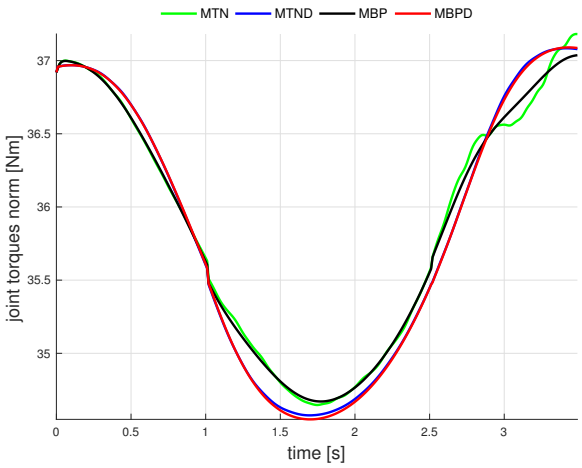
Experiments

The same two tasks considered in simulations were achieved also experimentally. The different torque optimization methods were implemented using C++. Experiments were performed using the position control mode of the LWR through the KUKA FRI library, feeding as reference the instantaneous motion obtained from the torque optimization schemes, with a sampling time $T_s = 5$ [ms]. The torques measured by the joint torque sensors during task execution are used to assess the obtained performance. Torque data have been processed through a low-pass filter to eliminate measurement noise.

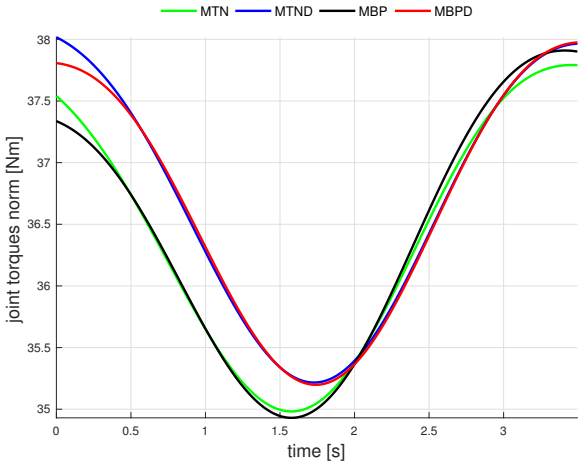
The control parameters for different methods were equal to ones used for simulations. Joint torques measured (by joint-torque sensors) during the tasks are used to evaluate the mean torque norms and assess the quality of performance. Because of measurements noise, torque data are processed through a low-pass filter before doing computations.

Although the experiments and simulations were done using different sampling time T_s, the obtained results for them were quite similar, see Figs. 3.11–3.16. The residual small differences between simulations and experiments are due to model uncertainty, unmodeled dynamics (motor friction, joint elasticity), measurement noise (encoders and torque sensors), and the filtering post-process. Also, because of this filtering, the joint torque norms during the whole experiments are smother whereas the curves starts slightly misplaced in Figs. 3.11(b) and 3.14(b).

In both tasks, joint oscillations during the *MTN* method do not affect the execution of the desired Cartesian trajectory, also thanks to the presence of the feedback control action on the task error. The maximum of the Cartesian error norm in Fig. 3.17 is about 7×10^{-4} [m] for the circular path (and was even less for the linear path).

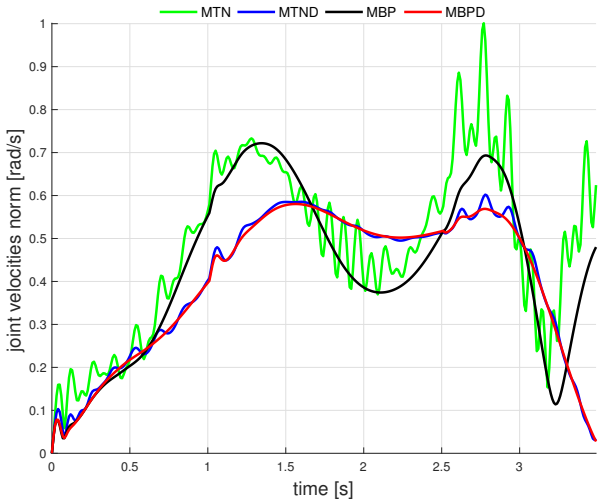


(a) V-REP simulations.

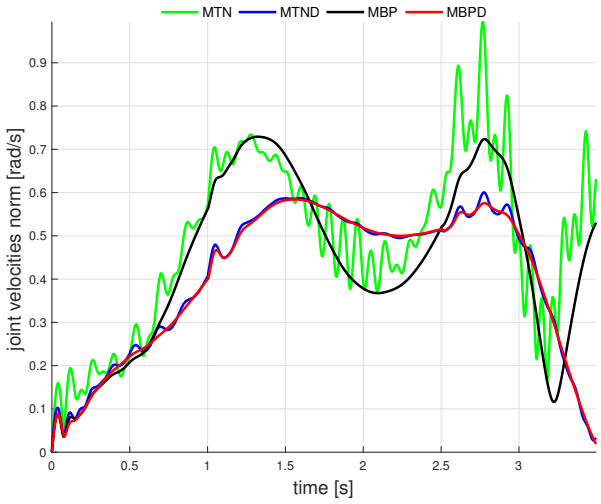


(b) Lab. experiments.

Figure 3.11: Joint torque norms for the KUKA LWR using different optimization solutions along a linear path in simulations and experiments.

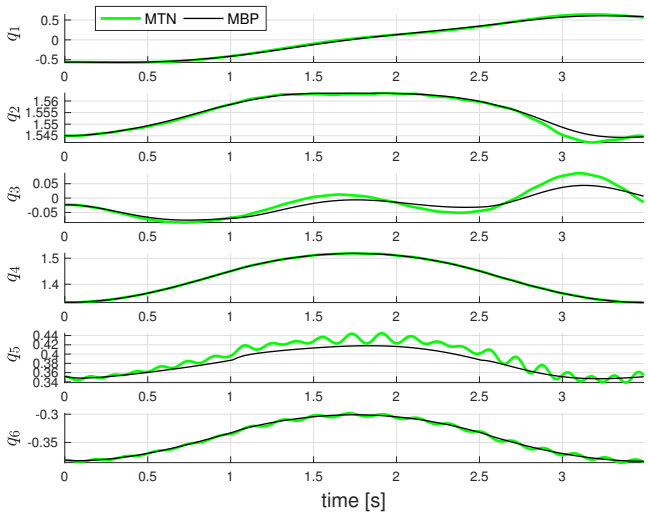


(a) V-REP simulations.

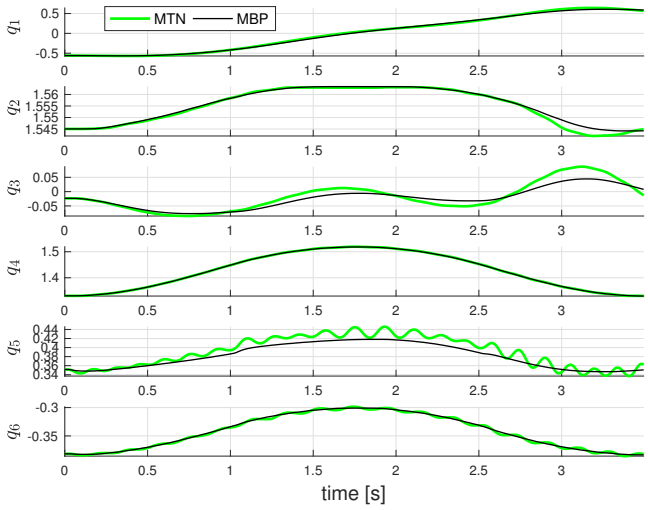


(b) Lab. experiments.

Figure 3.12: Joint velocity norms for the KUKA LWR using different optimization solutions along a linear path in simulations and experiments.

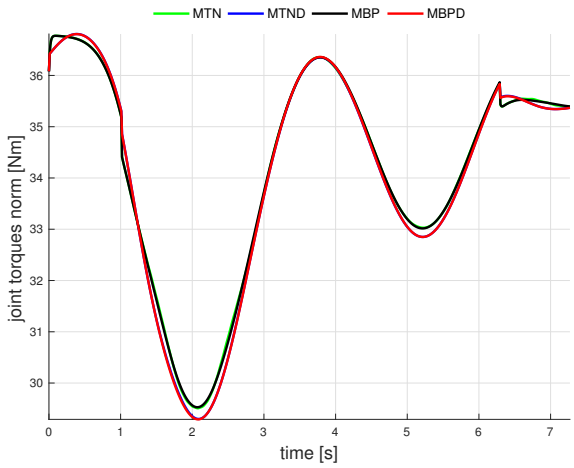


(a) V-REP simulations.

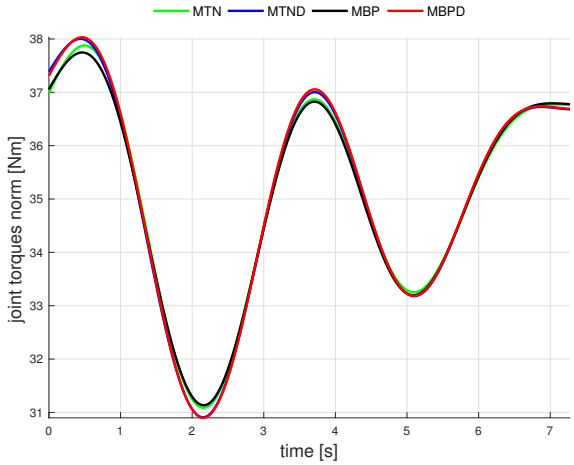


(b) Lab. experiments.

Figure 3.13: Joint positions for the KUKA LWR using *MTN* and *MBP* solutions along a linear path in simulations and experiments.

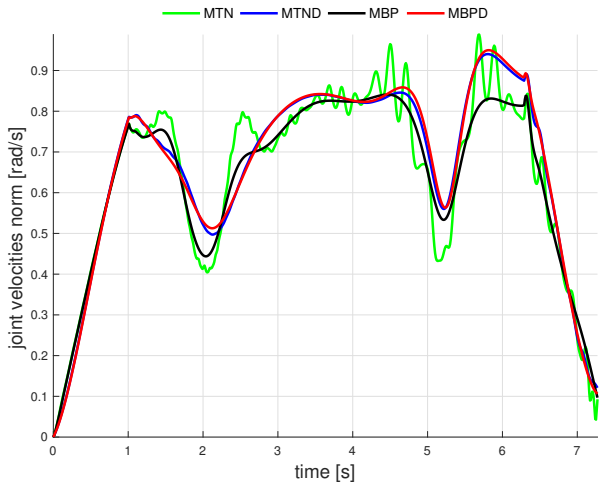


(a) V-REP simulations.

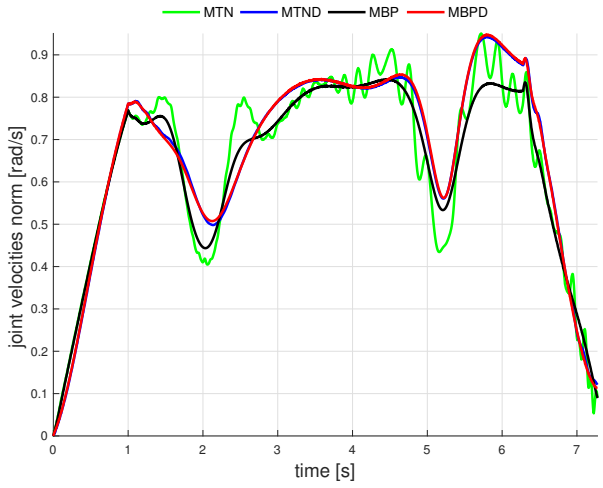


(b) Lab. experiments.

Figure 3.14: Joint torque norms for the KUKA LWR using different optimization solutions along a circular path in simulations and experiments.

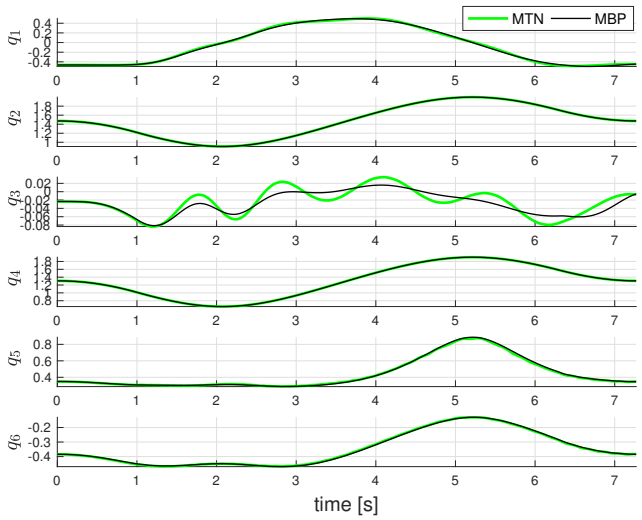


(a) V-REP simulations.

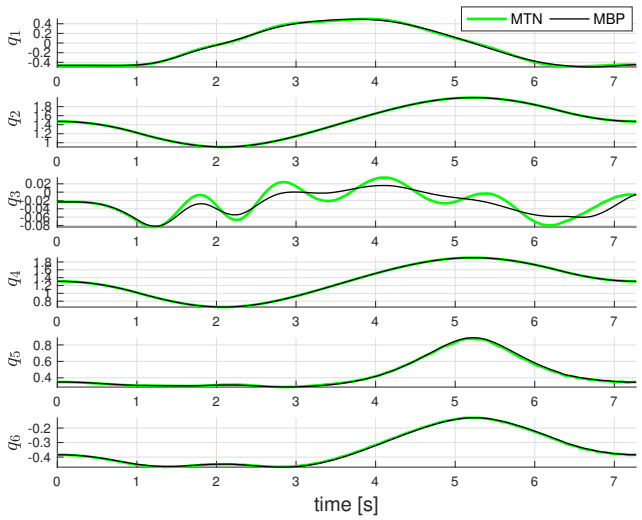


(b) Lab. experiments.

Figure 3.15: Joint velocity norms for the KUKA LWR using different optimization solutions along a circular path in simulations and experiments.

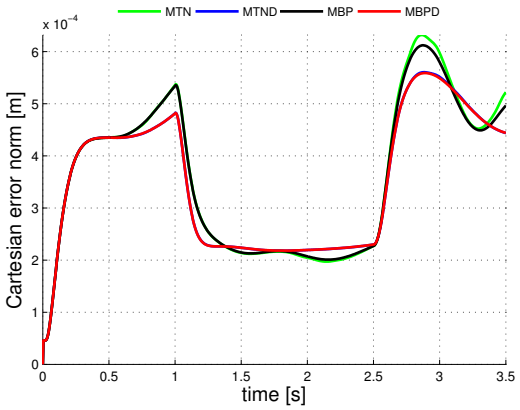


(a) V-REP simulations.

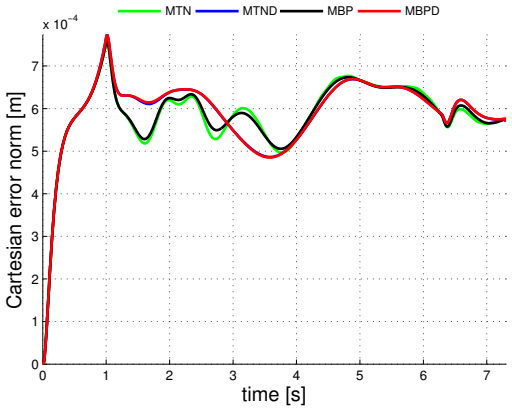


(b) Lab. experiments.

Figure 3.16: Joint positions for the KUKA LWR using *MTN* and *MBP* solutions along a circular path in simulations and experiments.



(a) Linear task.



(b) Circular task.

Figure 3.17: Cartesian error norm along the linear and circular paths in the KUKA LWR experiments using different torque optimization methods.

Conclusion

Different optimization techniques can be used to exploit robot redundancy w.r.t one or more desired objectives (Chapter 1). Local optimization methods can lead to closed form solutions suitable for on-line applications, while global methods have complex formulations and include numerical procedures.

In this thesis the robot redundancy resolution is optimized for two different separated objectives. First, we presented a two-step method that addresses in an approximate but effective way the minimum time control problem for redundant robots moving along a given Cartesian path (Chapter 2). In a first step, a local second-order inverse kinematic method was used to map Cartesian paths into joint paths, while in a second step an established minimum time planning algorithm provides the optimal solution under joint velocity and torque bounds. As ingredients in our method, we used weighted pseudoinversion, optimized an inertia-related criterion, and included a damping term in the null-space of the task Jacobian. Working at the second-order level allows obtaining smoother paths while including dynamic issues.

Based on the extensive tests on various paths and for different robots, which are reported only in part here, we have found consistent improvements in the obtained motion times over similar approaches that use first-order inverse solutions at the velocity level. As shown experimentally, the combination of our second-order solution method with the TOPP algorithm leads to reasonable performance in tracking minimum time trajectories.

Our two-step second-order method leads to faster motion times, but is still intended currently for off-line planning situations only. Real-time limitations are distributed between both steps, and depend on several aspects, leading to running times in the order of seconds. On the other hand, finding the accurate global minimum time with a constrained solution trajectory by means of general numerical optimization techniques requires at present minutes to hours of computation. We plan to pursue computationally more efficient implementations of the present method, as well as other semi-global methods that can run in real time, such as model predictive control along Cartesian paths for redundant robots that minimizes the motion time to go. Also, we would like to apply the proposed techniques on the point-to-point time optimal control for redundant robot. The joints friction and joint elasticity will be also considered.

As a second optimization objective, we proposed different torque optimization methods to address the instability issue in redundant robots when the norm of the joint torque is instantaneously minimized (Chapter 3). The model-based short preview scheme *MBP* optimizes the norm of the joint torque at the current and at a single future but close instant, anticipating the possible dramatic growth of joint torques. Alternatively, the introduction of a desired momentum-damping joint torque in the null space of the task can reduce the associated drift in joint velocities. These two local control schemes can be used separately (*MTND*) or in combination (*MBPD*), leading to robot behaviors that are consistently stable in performing short and long task trajectories, without peaks or oscillations in torques or velocities. In general, *MBPD* seems the best optimization method since it generates smooth motion with lower residual joint velocity at the end and with a similar torque demand of the other methods. Despite the selection of a best preview instant is likely to depend on the robot own dynamics, we found in all cases a favourable insensitivity of the achieved robot performance when the preview instant was chosen in a non-vanishing intermediate range.

Future work will address an adaptive choice of the preview window, as well as the use of different weights for the torques at the two instants considered in the optimization scheme with short preview. The preview concept and the momentum-damping technique could be used also for other types of dynamic optimization problems (e.g., minimum energy).

A

Notes on the Universal Robots UR10

The UR10 robot by the Universal Robots considered in this work in Fig. A.1 is a lightweight rigid industrial manipulator that consist of six revolute joints. Its total weight is 28.9 kg, with a payload of 10 kg. The robot is equipped with joints position and motor current sensors. Table A.1 contains the position, velocity, and torque limits for each joint. The robot kinematics are computed according to the link frames in Fig. A.1, which are chosen according to the Denavit-Hartenberg convention (Denavit and Hartenberg [1955], Siciliano et al. [2008]). The associated parameters are given in Tab. A.2. Although the manufacturer distributes the numerical values of all dynamic parameters of the UR10 manipulator, it is found experimentally in Gaz et al. [2018] that using these parameters lead to un sufficiently reliable dynamic model. According to that, all UR10 computations in this work were done using the alternative dynamic model presented in Gaz et al. [2018].

In this work we considered UR10 robot in simulations only. We used the V-REP software, provided by Coppelia Robotics [2015], where the robot model is supplied. Using this program, static and dynamic simulations can be achieved using different sampling rates. In the static simulations both the robot and the environment react only according to the desired commands. While during the dynamic mode and by using the different supported dynamic engines, the robot is expected to behave realistically and similarly to the real system. However the un modeled dynamics make the robot behavior different from the real system. In this work all simulations done with UR10 robot were in the static mode with the minimum possible sampling rate 10[ms].

Table A.1: Position, torque and velocity limits for the UR10 robot.

Joint number	Range of motion [rad]	Maximum torque [Nm]	Maximum velocity [rad/s]
1	$\pm 2\pi$	330	2.10
2	$\pm 2\pi$	330	2.10
3	$\pm 2\pi$	150	3.14
4	$\pm 2\pi$	56	3.14
5	$\pm 2\pi$	56	3.14
6	$\pm 2\pi$	56	3.14

Table A.2: Denavit-Hartenberg parameters of the UR10 robot.

Link number	a_i [m]	α_i [rad]	d_i [m]	θ_i [rad]
1	0	$-\pi/2$	$d_1 = 0.128$	q_1
2	$a_2 = 0.6127$	0	0	q_2
3	$a_3 = 0.5716$	0	0	q_3
4	0	$-\pi/2$	$d_4 = 0.1639$	q_4
5	0	$\pi/2$	$d_5 = 0.1157$	q_5
6	0	0	$d_6 = 0.922$	q_6

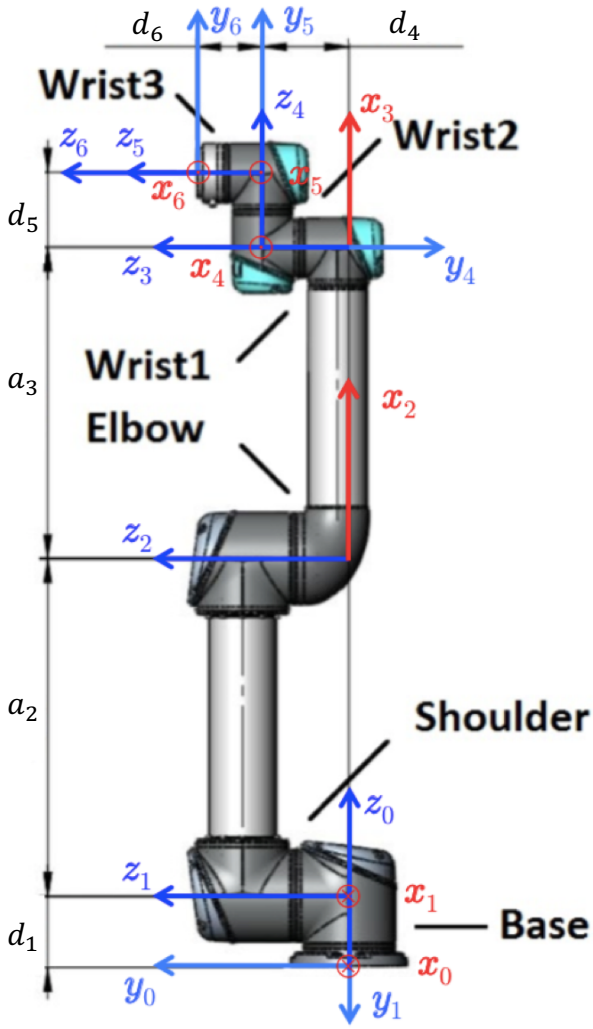


Figure A.1: Denavit-Hartenberg frames of the UR10 robot.

Notes on the KUKA LWR IV robot

The KUKA LWR IV robot (*Light Weight Robot*) considered in this work Fig. B.1, is usually used for research development and consists of seven revolute elastic joints. However, in all study cases of this work, it is considered as a rigid robot. Its total weight is approximately 16 kg, with a rated payload of 7 kg. All joints are equipped with position sensors on the motor and link sides, and with a joint torque sensor. Table B.1 contains the position, velocity, and torque limits for each joint. The robot kinematics are computed according to the link frames in Fig. B.1. The associated parameters are given in Tab. B.2. The constructor, KUKA, has still not released a public version of its dynamic model. In this work we computed all the robot dynamic features depending on the reverse engineering approach by Gaz et al. [2014].

For simulations, we used the robot model provided by the V-REP. For the supplied KUKA LWR robot model, several aspects make its behavior different from the real system Cefalo [2015]. First, the motor dynamics and the low level electronic controllers are not modeled. Also, the friction is neglected and the mass distribution is considered uniform. In this work all simulations done with KUKA LWR were in the static mode with the minimum possible sampling rate 10[ms].

For the experiments, the KUKA LWR IV system in Fig. B.2 is used. The robot controller is connected to a remote PC node via an Ethernet connection and the Fast Research Interface library (FRI) by KUK [2011] is used through Microsoft Visual studio with C++ programming language to set up the desired control architecture. The robot system can work with the sampling rates $1, 2$ and 5 [ms].

Table B.1: Joint position, torque and velocity limits of the KUKA LWR IV.

Joint number	Range of motion [rad]	Maximum torque [Nm]	Maximum velocity [rad/s]
1	± 2.97	176	1.92
2	± 2.09	176	1.92
3	± 2.97	100	2.23
4	± 2.09	100	2.23
5	± 2.97	100	3.56
6	± 2.09	38	3.21
7	± 2.97	38	3.21

Table B.2: Denavit-Hartenberg parameters of the KUKA LWR IV.

Link number	a_i [m]	α_i [rad]	d_i [m]	θ_i [rad]
1	0	$\pi/2$	0	q_1
2	0	$-\pi/2$	0	q_2
3	0	$-\pi/2$	$d_3 = 0.4$	q_3
4	0	$\pi/2$	0	q_4
5	0	$\pi/2$	$d_5 = 0.39$	q_5
6	0	$-\pi/2$	0	q_6
7	0	$\pi/2$	$d_7 = 0.078$	q_7

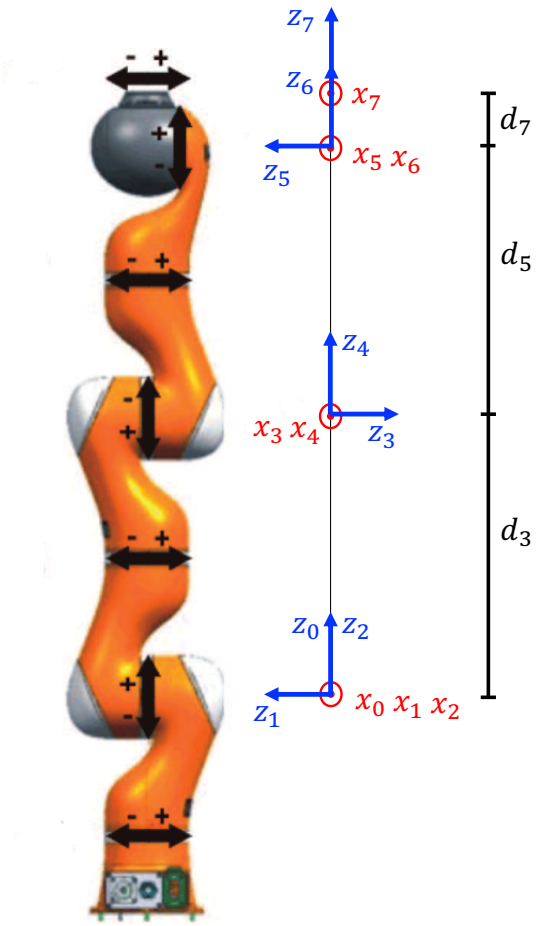


Figure B.1: Denavit-Hartenberg frames of the KUKA LWR IV: All x-axes point toward the viewer (frames are displaced sideways for better clarity).

Figure B.2: The KUKA LWR system at DIAG Robotics Lab. (A) The LWR body. (B) The KR C2 lr robot controller unit. (C) PC node. (D) Kuka control panel.

C

Positive definiteness of $\boldsymbol{Q}$ in preview-based methods

We provide a simple proof that the $\boldsymbol{Q}$ matrix in (3.19) for the *MBP* method will always be symmetric and positive definite, as claimed. A $n \times n$ matrix $\boldsymbol{Q}$ is symmetric when $\boldsymbol{Q}^T = \boldsymbol{Q}$, and positive definite iff

$$\boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v} > 0, \text{ for all } \boldsymbol{v} \neq \boldsymbol{0}.$$

The symmetry of $\boldsymbol{Q}$ follows immediately from the construction of this matrix and from the fact that the robot inertia matrix $\boldsymbol{M}(\boldsymbol{q})$ is itself symmetric for all configurations Siciliano et al. [2008]. Splitting now $\boldsymbol{v} \neq \boldsymbol{0}$ in two parts $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ according to the block matrix structure in (3.19), $\boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v}$ can be written as

$$\begin{pmatrix} \boldsymbol{v}_1^T & \boldsymbol{v}_2^T \end{pmatrix} \boldsymbol{Q} \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} = w_k ||\boldsymbol{M}_k \boldsymbol{v}_1||^2 + w_{k^+} ||\boldsymbol{M}_{k^+} \boldsymbol{v}_2 + T \boldsymbol{S}_{k^+} \boldsymbol{v}_1||^2.$$

Being $||\boldsymbol{v}_i|| > 0$ for all $\boldsymbol{v}_i \neq \boldsymbol{0}$, and since $\boldsymbol{M}_k$ and $\boldsymbol{M}_{k^+}$ are positive definite, then $\boldsymbol{M}_k \boldsymbol{v}_1 \neq \boldsymbol{0}$ and $\boldsymbol{M}_{k^+} \boldsymbol{v}_2 \neq \boldsymbol{0}$ for all $\boldsymbol{v}_i \neq \boldsymbol{0}$. Now consider the two cases:

- $\boldsymbol{v}_1 = \boldsymbol{0}, \boldsymbol{v}_2 \neq \boldsymbol{0}$. In this case $\boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v} = w_{k^+} ||\boldsymbol{M}_{k^+} \boldsymbol{v}_2||$, and since $w_{k^+} \neq 0$ then $\boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v} > 0$.
- $\boldsymbol{v}_1 \neq \boldsymbol{0}$. Since $w_k > 0$, then the term $w_k ||\boldsymbol{M}_k \boldsymbol{v}_1||^2 > 0$. Since $w_{k^+} > 0$ and the norm of a vector is strictly non-negative, then the term $w_{k^+} ||\boldsymbol{M}_{k^+} \boldsymbol{v}_2 + T \boldsymbol{S}_{k^+} \boldsymbol{v}_1||^2$ is non-negative. Thus, also in this case it follows that $\boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v} > 0$.

For the $\boldsymbol{Q}$ matrix of the *MBPD* method, the same previous procedure can be used to prove its symmetry and positive definiteness.

Bibliography

K. Al Khudir and A. De Luca. Faster motion on cartesian paths exploiting robot redundancy at the acceleration level. *IEEE Robotics and Automation Letters*, 3(4):3553–3560, 2018.

K. Al Khudir, G. Halvorsen, L. Lanari, and A. De Luca. Stable torque optimization for redundant robots using a short preview. *IEEE Robotics and Automation Letters,* Accepted on January, 2019.

F. Basile and P. Chiacchio. A contribution to minimum-time task-space path-following problem for redundant manipulators. *Robotica*, 21(2):137–142, 2003.

J. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. of Robotics Research*, 4(3):3–17, 1985.

M. Cefalo. Notes on the KUKA LWR4 dynamic model, 2015. URL http://www.coppeliarobotics.com/contributions/LBR4p_dynamic_model.pdf.

T.H. Chen, F.T. Cheng, Y.Y. Sun, and M.H. Hung. Torque optimization schemes for kinematically redundant manipulators. *J. of Robotic Systems*, 11(4):257–269, 1994.

P. Chiacchio. Exploiting redundancy in minimum-time path following robot control. In *Proc. American Control Conf.*, pages 2313–2318, 1990.

P. Chiacchio and M. Concilio. The dynamic manipulability ellipsoid for redundant manipulators. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 95–100, 1998.

S. Chiaverini, G. Oriolo, and I. Walker. Kinematically redundant manipulators. In *Springer handbook of robotics*, pages 245–268. 2008.

D. Constantinescu and E.A. Croft. Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *J. of Robotic Systems*, 17(5):233–249, 2000.

Coppelia Robotics. V-rep virtual robot experimentation platform, 2015. URL http://www.coppeliarobotics.com.

A. De Luca and L. Ferrajoli. Exploiting robot redundancy in collision detection and reaction. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3299–3305, 2008.

A. De Luca and G. Oriolo. The reduced gradient method for solving redundancy in robot arms. *IFAC Proceedings Volumes*, 23(8):133–138, 1990.

J. Denavit and R.S. Hartenberg. A kinematic notation for low pair mechanisms based on matrices. *Trans. of ASME, J. of Applied Mechanics*, 22:215–221, 1955.

M. Diehl, H.G. Bock, H. Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006.

V. Duchaine, S. Bouchard, and C.M. Gosselin. Computationally efficient predictive robot control. *IEEE/ASME Trans. on Mechatronics*, 12(5):570–578, 2007.

F. Flacco and A. De Luca. Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties. *Robotics and Autonomous Systems*, 70:191–201, 2015.

M. Galicki. Time-optimal controls of kinematically redundant manipulators with geometric constraints. *IEEE Trans. on Robotics and Automation*, 16(1): 89–93, 2000.

C. Gaz, F. Flacco, and A. De Luca. Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1386–1392, 2014.

C. Gaz, E. Magrini, and A. De Luca. A model-based residual approach for human-robot collaboration during manual polishing operations. *Mechatronics*, 2018.

A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016.

J. Hollerbach and K. Suh. Redundancy resolution of manipulators through torque optimization. *IEEE J. of Robotics and Automation*, 3(4):308–316, 1987.

K. Kazerounian and Z. Wang. Global versus local optimization in redundancy resolution of robotic manipulators. *Int. J. of Robotics Research*, 7(5):3–12, 1988.

O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. on Robotics and Automation*, 3(1):43–53, 1987.

S. Kim, K. Park, and J. Lee. Redundancy resolution of robot manipulators using optimal kinematic control. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 683–688, 1994.

KUKA.FastResearchInterface 1.0. KUKA System Technology (KST), D-86165 Augsburg, Germany, 2011. Version 2.

D. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer, 3rd edition, 2010.

S. Ma. A balancing technique to stabilize local torque optimization solution of redundant manipulators. *J. of Robotic Systems*, 13(3):177–185, 1996a.

S. Ma. Local torque minimization of redundant manipulators with considering end-motion joint velocities. In *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 1477–1482, 1996b.

S. Ma and D.N. Nenchev. Local torque minimization for redundant manipulators: A correct formulation. *Robotica*, 14(2):235–239, 1996.

S. Ma and M. Watanabe. Time optimal path-tracking control of kinematically redundant manipulators. *JSME Int. J. Ser. C Mechanical Systems, Machine Elements and Manufacturing*, 47(2):582–590, 2004.

D. Martin, J. Baillieul, and J. Hollerbach. Resolution of kinematic redundancy using optimization techniques. *IEEE Trans. on Robotics and Automation*, 5 (4):529–533, 1989.

K. Matthew. OptimTraj: Trajectory optimization library for matlab, 2016. URL https://github.com/MatthewPeterKelly/OptimTraj.

E. Mingo Hoffman, A. Laurenzi, L. Muratore, N. G. Tsagarakis, and D. G. Caldwell. Multi-priority Cartesian impedance control based on quadratic programming optimization. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 309–315, 2018.

Y. Nakamura and H. Hanafusa. Optimal redundancy control of robot manipulators. *Int. J. of Robotics Research*, 6(1):32–42, 1987.

K. O'Neil. Divergence of linear acceleration-based redundancy resolution schemes. *IEEE Trans. on Robotics and Automation*, 18(4):625–631, 2002.

K. O'Neil and Y. C. Chen. Instability of pseudoinverse acceleration control of redundant mechanisms. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2575–2582, 2000.

J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying framework for robot control with redundant DOFs. *Autonomous Robots*, 24 (1):1–12, 2008.

Q. Pham. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Trans. on Robotics*, 30(6):1533–1540, 2014.

Ph. Poignet and M. Gautier. Nonlinear model predictive control of a robot manipulator. In *Proc. 6th Int. Work. on Advanced Motion Control*, pages 401–406, 2000.

A. Reiter, A. Müller, and H. Gattringer. On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators. *IEEE Trans. on Industrial Informatics*, 14(4):1681–1690, 2018.

N. Scianca, M. Cognetti, D. De Simone, L. Lanari, and G. Oriolo. Intrinsically stable MPC for humanoid gait generation. In *Proc. 16th IEEE-RAS Int. Conf. on Humanoid Robots*, pages 601–606, 2016.

Z. Shiller. On singular time-optimal control along specified paths. *IEEE Trans. on Robotics and Automation*, 10(4):561–566, 1994.

I. Shim and Y. Yoon. Stabilization constraint method for torque optimization of a redundant manipulator. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2403–2408, 1997.

K. Shin and N. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans. on Automatic Control*, 30(6):531–541, 1985.

B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modeling, Planning and Control*. Springer, 3rd edition, 2008.

J.E Slotine and H.S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Trans. on Robotics and Automation*, 5(1):118–124, 1989.

K. Suh and J. Hollerbach. Local versus global torque optimization of redundant manipulators. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 619–624, 1987.

Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4906–4913, 2012.

D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Trans. on Automatic Control*, 54(10):2318–2327, 2009.

D. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. on man-machine systems*, 10(2):47–53, 1969.

T. Yoshikawa. Dynamic manipulability of robot manipulators. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1033–1038, 1985.

L. Zlajpah. On time optimal path control of manipulators with bounded joint velocities and torques. In *Proc. IEEE Int. on Robotics and Automation*, pages 1572–1577, 1996.