Universita' degli Studi di Genova
Dipartimento di Informatica, Sistemistica e Telematica

Istituto Italiano di Tecnologia
Dipartimento di Robotica, Scienze Cognitive e del Cervello

From humans to humanoids: a study on optimal motor control for the iCub

Serena Ivaldi

A thesis submitted for the degree of *PhilosophiæDoctor (PhD)*

XXII Doctoral School on Humanoid Technologies
April 2011

Thesis supervisors:

Prof. Giorgio Metta (Principal Adviser)
Robotics, Brain and Cognitive Sciences
Italian Institute of Technology
&
Department of System, Communication and Sciences
University of Genova, Italy

Prof. Marco Baglietto
Department of System, Communication and Sciences
University of Genova, Italy

Dr. Francesco Nori
Robotics, Brain and Cognitive Sciences
Italian Institute of Technology

This work has been carried out by Serena Ivaldi during her Ph.D. course in Humanoid technologies, under the joint supervision of Prof. Giorgio Metta and Prof. Marco Baglietto, with the additional supervision of Dr. Francesco Nori at the Robotics, Brain and Cognitive Sciences Department, Italian Institute of Technology, Genova, Italy, directed by Prof. Giulio Sandini. Her Ph.D. has been financially supported by the Italian Ministry of Education, University and Research (MIUR), the Fondazione Istituto Italiano di Tecnologia (IIT), and by the European Union through the projects ROBOTCUB, CHRIS, ITALK, VIACTORS.

1. Reviewer:

2. Reviewer:

3. Reviewer:

Day of the defense:

Signature from head of PhD committee:

If you don’t fail at least 90 percent of the time, you’re not aiming high enough.
Alan Kay

Acknowledgments

There are many people without whom this thesis would not have been possible, and to whom I am greatly indebted.
I am obliged to my supervisors for guiding me through this difficult but exciting experience in research, and to Professor Sandini, who gave me the opportunity to join RBCS and work in such a great research laboratory in IIT. I must thank Giorgio for his sage advice. A special acknowledgment to Riccardo and Olivier for reading the manuscript. My deepest gratitude and appreciation goes to Francesco, whose teaching, guidance and friendship have been invaluable in many senses, and a source of inspiration. If only we had worked together before.
A special acknowledgment to my lab mates, the iCubers*, colleagues and friends whom I have had the pleasure to work with and spent most of the time in the last years, and particularly to Alessandra, Valentina, Monica, Elisa and Ambra, who have been friends more than colleagues. There are many people I should name at this point: either if you shared with me a single moment of my PhD or many, thank you.*
I must thank my family, my sister, my friends, for loving and supporting me during these years. I hope they will be proud of me.
Finally, I will never stop being grateful to Paolo for his love, patient encouragement and support, for staying with me, and still being engaged to me despite my time being absorbed by study and work. This thesis is undoubtedly dedicated to him.

To Paolo

Glossary

CNS	Central Nervous System
EMG	Electro-Myo-Graphy/ic
CPG	Central Pattern generators
M(A)JM	Minimum (Angle) Jerk Model
M(C)TCM	Minimum (Commanded) Torque Change Model
MVT	Minimum Variance Theory
CA	Cognitive Architecture
ERIM	Extended RItz Method
(N)MPC	(Nonlinear) Model Predictive Control
DP	Dynamic Programming
LQ(G)	Linear Quadratic (Gaussian)
DM	Decision Makers
OHL	One-Hidden-Layer
(A)NN	(Artificial) Neural Networks
SVM	Support Vector Machines
(N)MSE	(Normalized) Mean Squared Error
CE(P)	Certainty Equivalence (Principle)
FH	Finite Horizon
RH	Receding Horizon
IH	Infinite Horizon
COD	Curse Of Dimensionality
DOF	Degrees Of Freedom
IK	Inverse Kinematics
FD	Forward Dynamics
CLIK	Closed Loop Inverse Kinematics
COM	Center Of Mass
EOG	Enhanced Oriented Graph
RNE(A)	Recursive Newton-Euler (Algorithm)
FT(S)	Force/Torque (Sensor)
OS	Operating System
IDE	Integrated Development Environment
API	Application Program Interface
PCB	Printed Circuit Board
DSP	Digital Signal Processing
(L)GPL	(Lesser) General Public License
CAD	Computer Aided Design

Synopsis

Robots are going to coexist and interact with humans, sharing the same unstructured environment and cooperating with them in many daily tasks. Even though industrial robots can achieve impressive performances in terms of precision, relying basically on joint position controllers and classical control theory, there is now a wide consensus that such controls are not adequate for the next generation of robots. More specifically, motion control must be improved, with a twofold aim:

- imitating humans to produce more natural and possibly efficient behaviors;
- guaranteeing motion safety.

My research stems from these considerations. In particular, I investigated motion control for the upper limbs of a humanoid robot, focusing on the most important primitive for any manipulation skill, i.e. reaching, taking inspiration from humans. Indeed, computational motor control provides different models describing human motions, that can be used in the attempt of transferring such criteria on robotic platforms. I concentrated on a theoretical framework which allows describing the reaching problem as the result of an optimization process, where the success in reaching the target is not the only important parameter (i.e. bringing the end-effector of a manipulator on the target configuration) but also how the limb moves in effecting such actions, i.e. the criteria which can be used to describe its action. If we see this as an optimization problem, then a stochastic functional optimization problem, with a suitable cost function, state equation and constraints must be designed. Because the solution of functional optimization problems is almost impossible *a priori* in real-time, an approximation technique combined with model predictive control has been addressed, where the solution to such problems is explicitly precomputed via numerical techniques. Various simulations and experiments on a humanoid platform confirmed the feasibility of the proposed approach. Subsequently, I focused on the implementation of a theoretical framework that allows estimating joint torques and external wrenches, under suitable hypotheses, for a wide class of robotic systems, and in particular for humanoids robots. The purpose was to provide a robot a force/torque control framework which, combined with the optimization techniques, would enable human-like movements with active compliance. Experimental results successfully demonstrated the possibility of controlling a complex humanoid robot in a compliant way. This lead to further investigations regarding how to transfer human strategies in varying stiffness and torques during point-to-point movements, using stochastic optimal control strategies. Although some activities related to this topic are still work in progress, preliminary results favor the application of such techniques, suggesting interesting developments.

Contents

Chapter 1

Introduction

It is a common belief that the human body moves *optimally* and that human movements are grounded on feedback and feedforward control processes [Todorov and Jordan, 2002]. Human limbs trajectories during goal-directed movements can be modeled by the optimization of a properly defined cost functional, usually nonlinear and sometimes non-differentiable, subject to sets of linear and nonlinear constraints [Biess et al., 2006, Berret et al., 2008].

In the literature different computational models can be found, describing trajectories in terms of minimization of variance [Harris and Wolpert, 1998], torque change [Uno et al., 1989], jerk [Flash and Hogan, 1985], energy of moto-neurons signals [Guigon et al., 2008], etc.

In humanoid robotics, where reaching is the fundamental action primitive, such models are particularly of interest [Richardson and Flash, 2000], because they do not focus only on the successful reach, but also on the trajectory performed by limbs, and the controls that cause these actions. Through the implementation of such computational models on robotic platforms it is possible to mimic human movements and achieve, within certain approximations, human-like behaviors. The crucial point is not to reproduce human behaviors to make the robot appearing more human-like or natural [Seki and Tadakuma, 2004] but to achieve efficient control and to understand which principles governing the human body can be transferred on a robotic platform, assuming that the human body is the optimal reference, refined by evolution and years of constant learning and improvement [Atkeson et al., 2000, Shadmehr and Wise, 2005]. Analogous reasons explain why optimal control is frequently addressed to solve complex problems like robot stabilization and walking [Lockhart and Ting, 2007, Atkeson and Stephens, 2007, Schultz and Mombaur, 2010, Mombaur et al., 2010].

In this perspective, the robot must be provided with a tool that is able to plan "optimally": if the biological principles describing its motion are known, it must be able to generate proper trajectories and execute desired motions in real-time (possibly without being too much resource-demanding) with a suitable control scheme [Mitrovic et al., 2010].

Unfortunately, implementations on humanoid platforms face computational limits, since most optimal control problems incur in the Curse Of Dimensionality (COD), and even the solution of simplified problems (e.g., after strong hypotheses reducing the complexity of the model) cannot always guarantee the fulfillment of time constraints [Diehl et al., 2009]. Rather than searching for a generalized solution to the planning problem, whose computational limits make

it unsuitable for online real-time control, approaches in the literature usually focus on the optimization of single point-to-point movements [Simmons and Demiris, 2005, Matsui et al., 2006, Seki and Tadakuma, 2004, Tuan et al., 2008].

The corresponding optimal control problems are tackled via numerical methods and nonlinear programming algorithms, but the optimization process requires heavy computations and often prevents the application in real-time. Since closed-form solutions are utterly hard to find (impossible in many cases) approximate solutions have to be considered.

Among the possible options, in this thesis an off-line approximation of the global control law is preferred: the complete precomputation of a neural approximation of an explicit Finite/Receding Horizon (FH/RH) optimal control law (supported by an intermediate control loop to compensate modeling errors) allows finding the controls almost instantly, leaving the machine free for other tasks during on-line execution (e.g. contact detection, learning, etc.) [Ivaldi et al., 2009b]. The proposed solution is globally only suboptimal, and locally optimal.

More specifically, the technique consists of two steps. In the first, off-line, a suitable sequence of approximating functions is trained, so that they can approximate the sequence of optimal control functions of a stochastic Finite Horizon problem. The ERIM is chosen as a functional approximation technique, while the use of feed-forward neural networks guarantees that the optimal solutions can be approximated at any desired degree of accuracy [Barron, 1993, Zoppoli et al., 2002, Kurková and Sanguineti, 2005]. In the on-line phase, a single forward computation of a neural network (consisting of few elementary operations) yields the proper control at each time instant.

Note that conventional Nonlinear Model Predictive Control techniques such as FH/RH usually solve single instances of optimization problems, i.e. each trajectory is the result of an optimization problem (typically varying its boundary conditions); conversely, in the proposed approach a generalized solution is found, for all the possible initial/desired conditions. The generalization is possible by combining functional approximation with stochastic optimal control. Thus, in the on-line phase no further processing is required; the computation of the on-line controls is very fast, consisting only in the evaluation of a functional approximator; real-time performances can be guaranteed; furthermore, the machine controlling the robot does not require an external optimization routine (usually resource consuming), or licensed software, nor specific hardware.

The feasibility of this approach has been empirically demonstrated for the control of different linear and nonlinear systems, such as a nonholonomic mobile robot [Ivaldi et al., 2008c] and planar manipulator; numerical results showing its effectiveness for different cost functions have been presented in [Ivaldi et al., 2008a].

For humanoids, planning can be carried out either in the task or directly in the robot joint space. In the former case, the optimal trajectories can be converted into suitable –joint level– motor commands, exploiting suitable kinematics or dynamics control layers. If Cartesian space is used, for example, and joint velocity or position commands are used to control the robot motion, one can use a classical closed-loop inverse kinematics algorithm (CLIK) for make the "task to space conversion", taking into account the manipulator physical limitations. By tuning the CLIK parameters (regulator, regularized Jacobian pseudo-inverse, etc.) it is possible to achieve great precision and stability in tracking the desired trajectory.

But, if robots are going to coexist with humans, sharing the same unstructured environment and interact with them and their objects, the capability to perform precisely a task must not subordinate to the primary requirement of motion safety. Clearly, suitable force control schemes are necessary to address the tasks with compliance requirements and to guarantee the global safety during motion. An interesting analysis of the effects of uncontrolled impacts of robotic manipulators on humans can be found in [Haddadin et al., 2008a, Haddadin et al., 2008b].

Classically, and especially in industrial environment, great effort has been focused on position control rather than compliance and force control, because the application domain required precise performances (which are normally achieved by stiff, high gain joint position feedback control). The lack of compliance has been traditionally compensated by collision avoidance solutions, where commonly the end-effector trajectory or the manipulator configuration is changed during motion so as to avoid collisions with the surrounding (or the self). The literature in this topic is vast, and outside the scope of the thesis, but the interested reader can refer to [Minguez et al., 2008, Kulic and Croft, 2007, Sisbot et al., 2010].

Recent developments in actuator technology have driven the attention towards systems capable of intrinsic joint-torque control and more in general *passive compliance*: variable impedance/variable stiffness actuators [Eiberger et al., 2010, Albu-Schaffer et al., 2010], series elastic actuators [Pratt and Williamson, 1995], pneumatics and hydraulics actuators, etc. Though being intrinsically compliant and thus safer with respect to DC motors, elastic elements combined with actuators do not guarantee safety, as they can store great amounts of potential energy, which once released can have greater impacts on both robot and environment, as recently shown in [Haddadin et al., 2010a]. Moreover, these solution often require consistent mechanical re-design [Tsagarakis et al., 2009] and the adoption of different forms of power sources [Amundson et al., 2005].

An alternative approach is *active compliance*, or active force, consisting in the regulation of the interaction forces at each instant of time by means of closed-loop force controllers [Sciavicco and Siciliano, 2005]. The principle being that if external forces can be detected or measured with suitable sensors, they can be controlled so as to regulate the interaction forces to the desired value: thus, active force control strategies can be build [Mistry et al., 2010, Calinon et al., 2010, Fumagalli et al., 2010a]. The main advantage of the active regulation over the passive one is the possibility of regulating forces within a wider range of values. One disadvantage is the response delay of the regulator, which typically limits the bandwidth of the controlled system. This approach, given the model of the robot (such as rigid body dynamics model), requires the hardware necessary to measure forces/torques (not only in joints, but at the end-effectors and at any other possible contact point of the robot). Traditionally, the most adopted solution consists in modifying the motor/joint group in order to insert suitable torque sensors, as was done in [Parmiggiani et al., 2009] to integrate joint-torque sensing in the fore-arm of the humanoid iCub, or redesign the robot to include torque sensing, as was done in [Luh et al., 1983] for a Stanford manipulator. As an alternative to placing joint torque sensors, an estimation of motor torques can be obtained from the current absorbed by the motors (feasible only when most of the motor torque is transmitted to the joint – low friction).

Another approach consists in exploiting *Force/Torque Sensors* (FTS): FTS are relatively small and compact, and can be often inserted in the kinematic chain easily when the available

space on the robotic platform is limited and passive elements cannot be inserted without radical changes. In industrial applications, robots are typically equipped with FTS mounted at the end-effector, where the most interaction with the environment occurs. The solution described in this thesis is based instead on a set of *proximal* FTS, instead at the base of the kinematic chains and far from the end-effectors: this configuration allows measuring not only interaction forces acting at the end of the chains, but also forces acting in between the sensor and distal joints. A similar solution has been adopted only once in [Morel and Dubowsky, 1996, Morel et al., 2000] where a single FTS was used to estimate the joint torques in the first 3 Degrees Of Freedom (DOF) of a PUMA manipulator. Here, we propose a method which exploits sets of FTS placed proximally in a multi-branched chain to estimate joint torques of complex kinematic chains. In particular, given the FTS measurements, if a precise dynamical model of the robot is known (i.e. a rigid body model), internal forces and torques can be computed easily by a classical recursive Newton-Euler method, and if suitable assumptions hold, certain external forces can be also estimated. As prove of the effectiveness of this approach, we successfully estimate 32 of the 53 DOF of a full-body humanoid robot.

The theoretical framework for computing simultaneously intrinsic torques and external wrenches applied to single and multiple branches kinematic chains, is based on two fundamentals:

- the *Enhanced Oriented Graph (EOG)*, i.e. a graph representation of kinematic chains, enriched with symbols for representing unknowns and sensor measurements;
- the *Recursive Newton-Euler Algorithm (RNEA)* for the computation of inverse dynamics of fixed and floating base kinematics chains.

A systematic procedure for computing $N + 1$ external wrenches from N internal wrenches (i.e. measurements from FTS) is also given, under certain assumptions. Remarkably, under these conditions all joint torques can be theoretically computed. In order to compute external wrenches, the main requirement is that their application point must be known: this information can be fixed *a priori* for particular robot tasks, but in general must be updated on-the-fly. On the platform used in this work – the iCub – it is provided by a set of tactile sensors, constituting a sort of "artificial skin" [Cannata et al., 2008, Maggiali et al., 2008].

Experiments have been performed in a variety of floating base conditions (e.g. standing, crawling) and with different interactions with the environment. Experimental observations also proved that controlling joint impedance in the robot is fundamental to obtain compliant interaction with the environment and make the robot move in a more safe and natural way. Interestingly, the neural optimal controller has been effective in computing adaptive strategies for controlling stiffness and torque of elastic actuators during point to point movements. Although some experiments on the iCub are work in progress, preliminary results show that it is possible to apply computational motor control models used to investigate human movements onto robots, up to a certain extent, given the physical differences between the two systems.

Contribution of this thesis

In this thesis two theoretical frameworks are proposed. The first is a mathematical tool to implement stochastic optimal motion control on humanoid robots, which in a sense seeks inspiration from computational motor control models. The proposed method consists of a stochastic approximation technique combined with a model predictive control scheme; intermediate controls, at joint level, are introduced to comply with the robot requirements. The second is a theoretical framework for computing the dynamics of a humanoid robot and estimate joint torques and external wrenches. Notably, this tool enabled to create different interfaces for controlling the robot in a compliant way, particularly joint torque control and impedance control. For each of the above, C++ software libraries have been produced, the NeuBot and iDyn library respectively. The latter has been released under GPL license as a part of the iCub Project, while the first is available upon request from the author.

The content of this thesis has been partially published as research papers. Their detailed references are reported in the bibliography section.

The thesis is organized as follows.

In Chapter 2, the humanoid robots which have been used as experimental platforms are described: James, a humanoid torso, and iCub, a fully body humanoid.

In Chapter 3, the optimality principles used to describe motor control are presented. Starting from a brief discussion on the experimental observations performed on humans, we overview the main computational motor control models which have or may have or might influence motor control models in humanoids. Some successful examples of integration of optimal control in robotics are presented, along with a discussion of the main differences between the two systems, which sometimes prevent a straightforward application of neuro-computational models into robots.

In Chapter 4, the reaching and tracking problem are introduced in the optimal control framework, and the approximate solution via the ERIM, combined with NMPC is discussed. Some numerical results are discussed, pointing out the advantages and disadvantages of the method, and particularly of the ERIM. However, its application to different problems, highlights its ability to adapt to different contexts: deterministic, stochastic, with linear and nonlinear systems, *etc*.

In Chapter 5, the closed loop lower level controllers are discussed: first, the Inverse Kinematics, secondly the force/torque control layer. Since the iCub is not provided with joint torque sensing, in Section 5.4 a framework for computing internal torques and external wrenches from a set of proximal FTS (available in the iCub) is presented. Finally, Section 5.5 reports some experiments with the humanoid platforms.

Chapter 6 draws the conclusions and suggests future works.

Chapter 2

The robotic platforms

Two humanoid robotics platforms have been used to validate the theoretical results discussed in this thesis, and to assess the proposed methods with experimental results: the 22 DOF upper-torso James, and the 53 DOF full-body iCub. Experiments have been performed at the Italian Institute of Technology, where both robotics platforms are available.

(a) (b)

Figure 2.1: The humanoid robots James **2.1(a)** and iCub **2.1(b)**.

2.1 The humanoid robot James

James [Jamone et al., 2006] is a 22 DOF torso (see Figure 2.1(a)), with the overall size of a 10 years old boy and a total weight of about 8 kg. It has a head, with moving eyes and neck, a left arm with a highly anthropomorphic hand.

The robot is actuated by 23 rotary DC motors (Faulhaber [Faulhaber, www]). Torque is transmitted to the joints by rubber toothed belts, pulleys and stainless-steel tendons. Cables pulling solutions have been particularly useful in designing the hand, since most of the hand

actuation have been located in the wrist and forearm rather than in the hand itself, where size and weight constraints would have limited the proliferation of DOF. Furthermore, tendon actuation naturally provides certain compliance to the system. Extra compliance has been introduced by means of springs in series with tendons, for example in fingers. The drawbacks of elastic transmission are the nonlinear effects which reveal during rough movements (i.e. when controlling joints with high velocities).

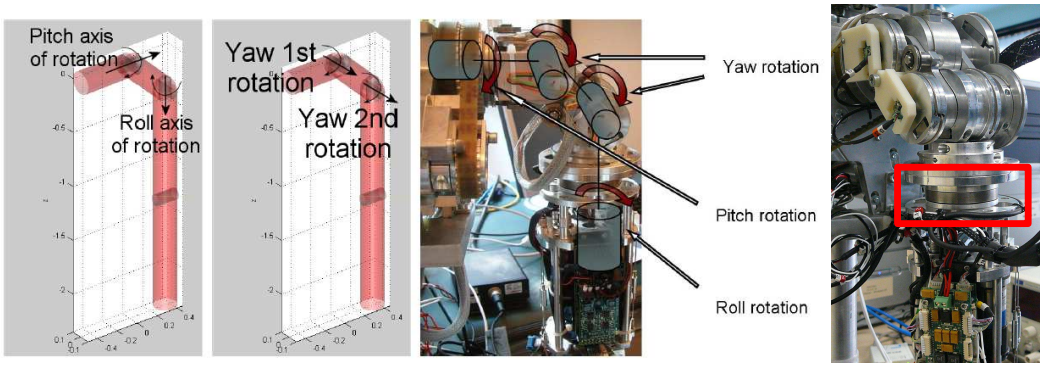
Mechatronics of James

The head has two eyes (i.e. CCD digital cameras, Dragonfly) which can pan and tilt independently, for a total of 4 DOF. A 3-axis orientation tracker (Intersense iCube2) is mounted on the top of the head, to emulate the vestibular system. The tracker, basically a gyroscope, provides an absolute measure of acceleration, velocity and position with respect to the Cartesian axes of a reference frame, thus it is also called inertial sensor. The head is mounted on a two DOF neck, consisting of a tendon driven rigid spring, which allows bending forward (pitch) and laterally (roll) [Nori et al., 2007a]. The actuation is obtained with a peculiar structure, recalling the design of a tendon-driven parallel manipulator: in particular, three steel tendons, separated by 120 degrees, are used to achieve the two motions. On the top of the neck, a custom-made force sensor with a cantilever beam structure is positioned, so as to provide force feedback to the three motors actuating the neck [Fumagalli et al., 2009].

The arm has seven DOF: three in the shoulder, one for the elbow and three in the wrist. In particular, the shoulder consists of three rotative joints, actuated through tendons and pulleys by three DC motors located in the torso: two joints (the ones yielding abduction) are mechanically coupled, as shown in Figure 2.2(a), so as to gather the shoulder a wider range of motion. Notably, thanks to this solution, James can perform wide range movements, for example it can reach its torso, its right shoulder and drive its left arm behind the head: these cannot be performed by even more recent humanoid robots, like iCub. In the middle of the upper arm, a single ATI mini45 FTS [ATI, www] is located, as shown in Figure 2.2(b). When FT sensors are added to a kinematic chain, they are usually placed on the end-effector, i.e. where most interaction occurs. In James the proximal location[1] has been chosen as the remainder of the free space in the upper and fore-arm was entirely occupied by the motors actuating the wrist, elbow and fingers, and DSP boards used to control them. The benefit of this configuration is that the FTS is able to detect interactions with the environment occurring not only on the end-effector (e.g. a grasp) but on the whole arm (e.g. the elbow colliding with an object). This means also that there is not a predetermined contact point, or the whole arm surface is a possible contact point.

A highly anthropomorphic hand, designed for grasping purposes, is the end-effector of the manipulator. The hand has five fingers, actuated by eight motors, and a total of 17 DOF: each of the five fingers has three joints (extension of the distal, middle and proximal phalanxes), and two additional DOF account for the thumb opposition and by the coordinated abduction/adduction of the other four fingers. Tactile information is provided by custom-made sensors, placed along the fingers and the hand palm. These sensors, constituted by a two part

[1] Proximal means far from the end-effector.



(a) (b)

Figure 2.2: **2.2(a)**. James shoulder (3 DOF). The picture shows the three DOF of the shoulder. Notice in particular how the yaw rotation is obtained by a double rotation around two parallel axes (image from [Jamone, 2010]). **2.2(b)** Detail of the upper-arm. The ATI Mini45 FT sensor (red square) is placed below the shoulder group.

silicone elastomer, a miniature Hall Effect sensor and magnet, have been designed specifically for James. More details on their design and application can be found in [Jamone et al., 2006, Jamone, 2010].

Hardware architecture for control

Motor control is distributed on eight Digital Signal Processing (DSP) boards (Freescale DSP-56F807, 80MHz, fixed point 16 bits [Freescale DSP, www]), which perform a fast low-level control loop (1KHz rate). A CAN-bus line allows the communication between the boards and a remote PC, where an ESD CAN-USB is provided. The middleware and the inter-process communication is grounded on YARP [Metta et al., 2006]. Magnetic and incremental encoders are used for the feedback position control loop implemented on the boards. Most of the motors are directly controlled by standard PID controllers, except for the shoulder, neck and eyes motors which require different control strategies to handle various mechanical constraints. The available DSP boards have limited memory and computation capability and cannot support but simple operations, namely low level motor control (basically PID position control), signal acquisition and pre-filtering from the optical encoders. For this reasons, implementing an on-line controller directly on the DSP boards is impossible in the current setup. Reference position and velocity commands can be set by the user through a standard YARP port, communicating in the local network to the so-called “James Interface”: a collection of YARP threads and modules, which acts as a bridge between the device drivers running on the boards and the remote PC. More accurate descriptions of the control architecture and the different low level as well as high-level control strategies, can be found in [Jamone et al., 2006, Fumagalli et al., 2009, Nori et al., 2007a].

Figure 2.3: Some pictures of James's arm moving in the space. Only shoulder and elbow joints are controlled.

2.2 The humanoid robot iCub

The aim of the RobotCub consortium [RobotCub Project, www] has been the development of an open-source infant-like robotic platform, aimed at reproducing the same motor and cognitive abilities of a two years old child [Tsagarakis et al., 2007, Metta et al., 2010]. The iCub not only has the shape of a human baby, but also a complex cognitive architecture reflecting the many processes involved in the functional development [Sandini et al., 2007, Vernon et al., 2007a, Vernon, 2010].

iCub is a 53 DOF full body humanoid robot, of the same size as a two-three years old child. It was designed to crawl on all fours, and sit up with free arms. The most of its DOFs are located in the upper-body, especially in the highly anthropomorphic hands, which allow dexterous and fine manipulation. It has comprehensively proprioceptive, visual, vestibular, auditory and haptic sensory capabilities.

Certain features of the iCub are unique. The peculiar aspect is that it is a completely open system platform: both hardware and software are licensed under the GNU General Public License (GPL), and the middleware used for intra-process communication, YARP, is an open-system too, released under GNU Lesser General Public License (LGPL) [Metta et al., 2006].

Mechatronics of the iCub

The iCub is about $104cm$ tall and weighs $22kg$, with a total of 53 DOF: six in the head (yaw, pitch, and roll in the neck, pan, tilt and vergence in the eyes), three in the torso (yaw, pitch, and roll), seven in each arm (three in shoulder, one in the elbow and three in the wrist), six for each leg (three in hip, one in the knee and two in the ankle), the remainder in the hands.

Actuation is provided by electric motors. The major joints are actuated by brushless DC motors, coupled with harmonic drive gears, so that high torques (up to $40Nm$) are guaranteed for the critical joints, such as hips, spine and shoulders. Head and hands are actuated by smaller DC motors. Most of the joints (e.g. in hands, shoulder, waist) are tendon-driven: this reduces the size of the robot and also introduces a certain elasticity (which can be a drawback if precise controls with high velocities are addressed, an advantage for its intrinsic compliance if safety is addressed).

The neck and the eyes are fully articulated (three DOF each), to support tracking and vergence behaviors.

Each hand (see Figure 2.6) has 5 fingers and 19 joints, and is actuated by 9 motors (since several joints are coupled). The first three fingers (thumb, index and middle finger) are in-

Figure 2.4: The humanoid iCub, fully covered by plastic shells, standing on a metallic mainstay over a mobile platform in the RBCS laboratory at IIT.

dependent and constitute eight DOF; while the fourth and fifth ones, used only for additional support to grasping, are coupled and constitute one single DOF. The hands allow a considerable dexterity though being very small: the palm length is 50mm, 25mm thick; the total width of the hand range from 34 to 60 mm at wrist and fingers respectively. These features are quite unique in humanoid robots with similar dimensions as iCub. This solution is possible because most of the actuation is located in the forearm, and tendons are routed to the hand joints via a wrist mechanism. Each joint is indeed tendon driven. The flexing of the fingers is directly controlled by the tendons, while the extension is based on a spring return mechanism (basically this saves one cable per finger).

The 7 DOF arm does not allow the same motion range as in James (e.g. iCub cannot touch its back) and additional physical constraints such as the body covers prevent a complete exploitation of the robot workspace. To provide better flexibility for manipulation, a 3 DOF waist has been incorporated, to increase the range of motion of the upper body, resulting in a larger workspace for manipulation. Finally, legs have been designed to support crawling and sitting, but are also adequate for standing and walking. The ankle has two DOF, namely flexion/extension and abduction/adduction (foot twist rotation was not implemented).

Additional sensing capabilities

Proprioception is provided at each joint by positional sensors, generally absolute position encoders. Joints positions are then retrieved directly from encoders measurements, while joints

velocities and accelerations are derived from position measurements through a least-squares algorithm based on an adaptive window filter [Janabi-Sharifi et al., 2000, Fumagalli et al., 2010b].

Many other different devices enrich the sensory capabilities of the iCub: digital cameras (for the eyes), gyroscopes, microphones, accelerometers, tactile and force sensors.

The latter three set of sensors are fundamental for the iCub active compliance, as will be discussed in Chapter 5. As shown in Figure 2.7(b) and 2.7(a), iCub is equipped with one inertial sensor (Xsens MTx-28A33G25 [Xsens, www]) located on the head, providing measurements of linear acceleration and angular velocity and acceleration[2]. Four custom-made six-axes FTS (see Figure 2.8), one per leg and arm, are placed proximally with respect to the end-effectors (hands and feet).

Sets of distributed capacitive tactile sensing elements are integrated on most of the plastic shells covering the robot limbs [Cannata et al., 2008], and provide a tactile feedback for possible contacts with the environment. This sort of "artificial skin" is constituted by a layer of capacitive pressure sensors included on a flexible Printed Circuit Board (PCB), with embedded electronics, covered by a silicone foam to protect each taxel (i.e. tactile element) and make the skin also more compliant. An example of the device for the forearm is shown in Figure 2.9.

Moreover, a set of fingertips, one per each finger, provides additional tactile information, for fine manipulation. The first prototype consisted of rectangular sensitive zones, made of conductive ink painted on an inner support and connected to a rigid PCB. The final device is made of a capacitive pressure sensor, a flexible PCB layer with circular taxels, wrapped around the inner support, and covered by layers of silicone foam and conductive rubber connected to the ground [Schmitz et al., 2010].

Hardware architecture for control

A set of DSP-based control cards, custom designed to fit the limited space available in the iCub, takes care of the low-level control loop. Each control board runs at 1 kHz, and is connected to the main relay CPU (a PC104, located in the robot head) via CAN bus (four lines in whole), which retrieves all motor-sensory information, handles synchronization and reformatting of the various data streams. More demanding computation can be performed on a PC cluster connected to the PC104 via a Gigabit Ethernet. Additional electronics have been designed to sample and digitalizes the numerous sensors: also in this case, all the signals converge to the PC104 by means of additional connections (e.g. serial, firewire). Moreover, the robot is equipped with an umbilical cord containing both an Ethernet cable and power supply line: with this solution, it can move freely in the space without being constrained to a specific position. A simple scheme describing the hardware/software architecture for control is shown in Figure 2.10.

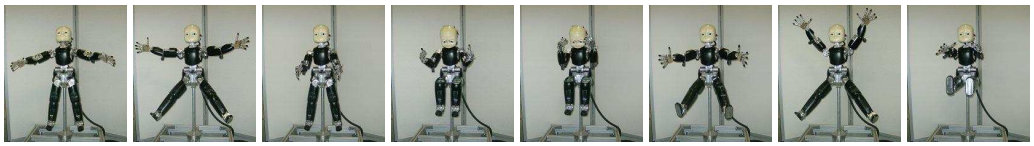
Software architecture

The core of the iCub software architecture is a set of modules developed on top of YARP [Metta et al., 2006, Fitzpatrick et al., 2008]. YARP is a set of cross-platform C++ libraries,

[2] Precisely, angular acceleration is found using an adaptive window filter.

which supports modularity, and provides universal interfaces with hardware and device modules. The philosophy is code reuse, which is to wrap each native device API, and provide a simple generic interface for any hardware device, so that changes in the hardware do not imply rewriting all modules, but only changing the API calls. Moreover, YARP is independent from both operating system and development environment, thanks to ACE and CMake: the first is an OS-independent library for inter-process communication, the second a cross-platform make-like tool to generate platform and IDE specific project files.

In the YARP framework, a suitable real-time layer is in charge of the low-level control of the robot, namely a set of processes running on the boards located through the robot body, interfacing sensors and actuators with the PC104. A pool of YARP modules defines a soft real-time communication layer, when multiple processes can coexist and exchange data through a series of universal ports, following the observer pattern by decoupling producers and consumers. This architecture is evidently suited for cluster computation: each module can be called or observed remotely from within the network. One evident drawback is that this architecture naturally excludes direct real-time control (i.e. a module directly sending commands to the joints), because of the many layers interposed in between the controller module (typically a module running on a cluster PC) and the robot hardware. Issues related to real-time performances must be addressed elsewhere.

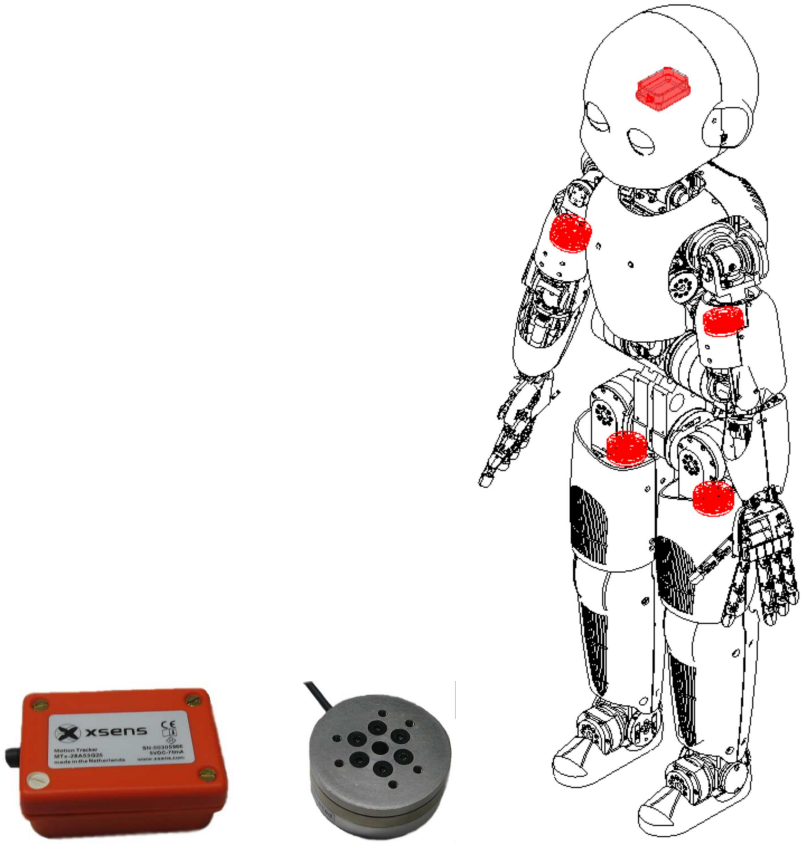


(a)

(b)

Figure 2.5: Top (2.5(a)): some snapshots of the "Yoga" demo, where iCub perform periodically a set of pre-programmed movements in the space. The base frame is fixed, being iCub supported by a metallic mainstay. **Bottom (2.5(b))**: some snapshots of iCub crawling on a carpet. Black straps are used to protect knees and wrists and simultaneously improve the friction of the plastic covers with the floor. Limbs motion is orchestrated by a controller based on central pattern generators [Degallier et al., 2008]. Self-body collision is prevented *a priori*. Interaction with the environment occurs on knees and wrists. The base frame is floating.

Figure 2.6: The iCub hand: in evidence, the embedded electronics (a MAIS board on the top), the tactile skin covering the palm and the tactile fingertips. More details can be found in [Schmitz et al., 2010].



(a) (b)

Figure 2.7: Right (**2.7(a)**): the force/torque and inertial sensors used in iCub. **Left** (**2.7(b)**): a mechanical scheme of the humanoid robot iCub: in evidence, the four proximal six-axes FTS (legs and arms) and the inertial sensor (head).

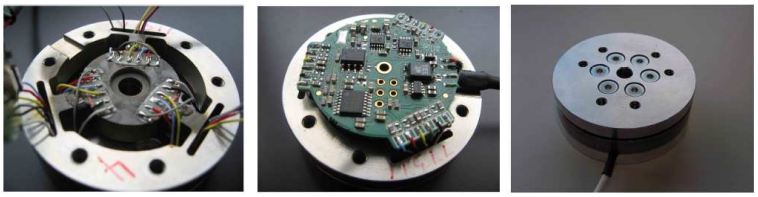


Figure 2.8: The custom FTS developed for iCub. Left: the sensing element. Center: the embedded electronics. Right: the assembled sensor. Notice the CAN line, going out from the sensor, which transmit sampled digital measures at $1ms$ rate (image from [Fumagalli et al., 2010b]).

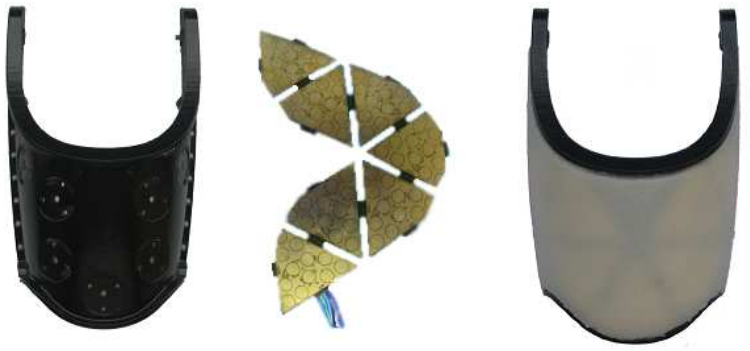


Figure 2.9: Distributed tactile elements constitute a sort of artificial "skin". The plastic cover, the elements and the final device for the fore-arm are shown. Details about skin fabrication and how iCub has been covered with it can be found in [Roboskin Project, www].

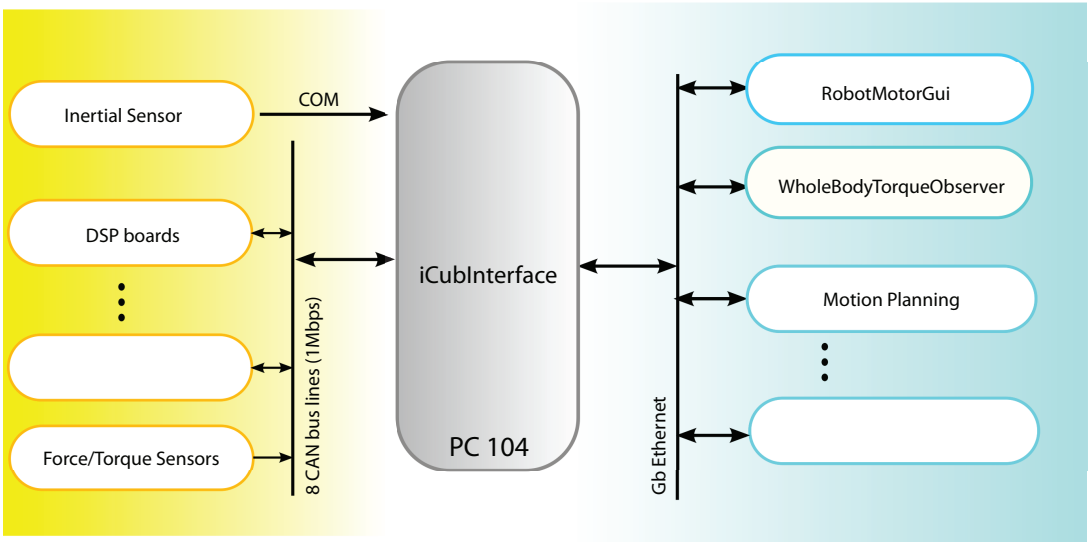


Figure 2.10: A scheme showing the networked control infrastructure. The PC104 collects joints measurements from the DSP boards, inertial measures from a specific COM port, and FTS measures via CAN bus. Each FTS is equipped with embedded electronics, with A/D converters. Through the iCubInterface, all measurements are replicated in the YARP local network, where a PC cluster is available, and multiple processes coexist. Among these, one is dedicated to the computation of "virtual" joint torques (as described in Section 5.4), one to the selection of the control modality (e.g. joint position, velocity or torque control). The estimated torques are sent back via the PC104 to the DSP boards to enable torque feedback. More details on this scheme can be found in the online documentation of the iCub: `http://eris.liralab.it/wiki/Force_Control`.

Chapter 3

Optimality: from humans to humanoids

Novel trends in computational neuroscience suggest that optimal control theory is crucial to understand the motor commands and the motor control that the human exerts during a task. Many researchers support the theory that the motions we observe in humans and animals are "optimal", because the sensorimotor system is the product of millions of years of evolution, but also because it constantly "evolves" being subject to continuous process such as learning, adaptation and training, which improve behavioral performance in terms of stability, accuracy and efficiency [Shadmehr and Wise, 2005, Franklin et al., 2008]. Even if the physical structure of the human body precludes certain motions, among a wide variety of possibilities, the CNS chooses to implement a selected set of planning strategies. We will consider these movements (that we can observe everyday) as "optimal", with the meaning that optimal control is a good modeling tool for human motor control. The idea that a motor controller is not only adaptive, but also optimal, suggests stating *the motor problem as a stochastic optimal control problem*.

It is a common assumption that the motions we observe in humans and animals are "optimal", because the sensorimotor system is the product of millions of years of evolution, but also because it constantly "evolves" being subject to continuous process such as learning, adaptation and training, which improve behavioral performance in terms of stability, accuracy and efficiency [Shadmehr and Wise, 2005, Franklin et al., 2008]. Even if the physical structure of the human body precludes certain motions, among a wide variety of possibilities, the CNS chooses to implement a selected set of planning strategies, leading to the "optimal" arm movements that we can observe every day.

Stochastic optimal control theory might provide the important link across the three levels of motor system: motor behavior, limb mechanics and neural control [Todorov, 2005, Todorov and Jordan, 2002, Todorov, 2004]. It naturally provides a mathematical framework to explain which are the controls generating the observed behavior, by providing or exploiting a cost function to describe the motion criteria.

Moreover [Scott, 2004], it might help to unravel how the primary motor cortex and other regions of the brain plan and control movement, providing valuable insights into the adaptive task-dependent control of movements. For this reason, neuroscientists and engineers cooperate

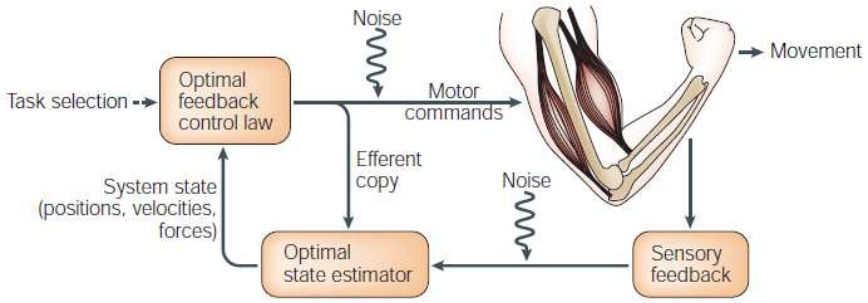


Figure 3.1: Optimal feedback control as the neural basis for motor control. The basic principle is that feedback gains are optimized on the basis of some index of performance. Such controllers correct variations (errors) if they influence the goal of the task; otherwise, they are ignored. Optimal state estimation is created by combining feedback signals and efferent copy of motor commands. The latter uses a forward internal model to convert motor commands to state variables (image from [Scott, 2004]).

for identifying the possible one-to-one correspondences between CNS and control schemes.

In general, all models agree on a certain control scheme, where the body (limbs, muscles, etc.) is the plant to control, the optimal controller provides feedback, sometimes feedforward terms, relying on an internal model of body and environment, while delayed signals are fed to a state estimator. The corresponding conceptual scheme, in one possible representation, is shown in Figure 3.1. A more detailed scheme is shown in Figure 3.2.

Concerning the optimal controller, two principal approaches can be used to select or identify a motion criteria. The first is to exploit *inverse optimal control theory* [Dupree et al., 2009, Krstic, 2009]: after recording experimental data, trying to infer the cost function to which the observed behaviors are optimal. The second, which is actually the most used, is to choose a priori a sound cost function and a mathematical model, and to verify its effectiveness in capturing the motion principles by comparing model predictions with experimental observations.

However, it is still uncertain how the CNS determines such optimal control policies. Motor control and learning explain the exceptional dexterity and rapid adaption to changes, which characterize human motor control, but do not provide an unique answer to the questions regarding the criteria which are at the basis of such plasticity and dexterity. So far, many different computational models have been proposed.

Existing optimization principles can be divided into two groups:

1. Deterministic approaches, where the cost is typically expressed as the integral of some deterministic function over the movement time. The problem is stated as the minimization of the cost subject to a set of dynamic constraints and boundary conditions.

    - Minimum jerk model [Flash and Hogan, 1985]
    - Minimum torque change model [Nakano et al., 1999]

2. Stochastic approaches, where random disturbances are included in the description of the model, thus the expected value of the cost function, subject to dynamic constraints and

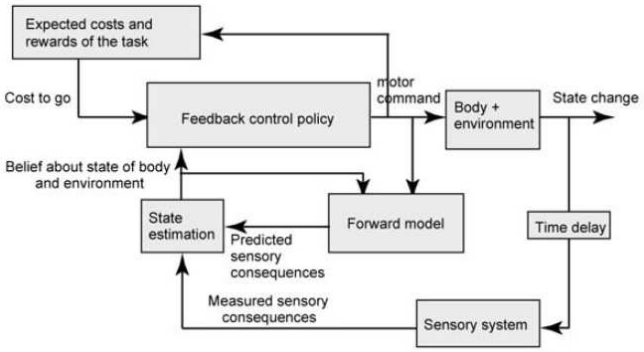


Figure 3.2: A schematic model for generating goal directed movements. With specific reference to the model proposed by Shadmehr and Krakauer, the cost/reward function refers to basal ganglia, the state estimation is performed by the parietal cortex, while the forward model can be put down to the cerebellum. The feedback controller is actually a combination of different modules which can be referred to the premotor and motor cortex (image from [Shadmehr and Wise, 2005]).

boundary conditions, can be minimized.

- Minimum variance model [Harris and Wolpert, 1998]
- Minimum intervention model [Todorov and Jordan, 2002]

In the following we will overview the main basic principles of human motor control which can be considered as "optimal", and focus on some computational models which have been developed to model goal directed movements. However, the question "which is the cost function?" will remain unanswered. We will then discuss our views on trying to transfer the concept of optimality from humans to humanoids, trying to identify the challenges and introducing the method we propose in this work, which will be object of the next chapter.

3.1 Optimality principles in human motor control

Human movements in adults show several prominent features. Multi-joint arm trajectories for discrete point-to-point planar movements, for example, have some spatiotemporal invariant features: roughly straight hand paths, bell-shaped velocity profiles and smooth acceleration [Morasso, 1983, Flash and Hogan, 1985, Kelso, 1982].

These invariant features can be seen as the result of an twofold optimization process: one coming from the CNS, evolved in time since the origin of our species, one from the personal ontogenesis process, where movements are refined. Even if the latter should yield in each person a certain variability to the motor trajectories, experimental results bring evidence that behaviors in human motor control are basically stereotyped.

Despite the huge number of possible combinations of agonistic and antagonistic muscle tensions that can generate the same torque, and the intrinsic noise of the motor system, observed hand trajectories are stereotyped, and the EMG signals show a typical triphasic pattern

activation. Clearly, the motor control system solves such infinite-dimensional problems according to some principles.

In the following, we will briefly describe the CNS, i.e. the biological structure in charge for motor control, and some insights on the learning and adaptation mechanisms which make the primate rough infant movements evolve to adult ones. Then, a review of the main computational motor control models, which can be of interest for robotic applications, is presented and discussed.

3.1.1 CNS and motor control

The CNS we study nowadays is the product of millions years of evolution, it is worth an analysis since it can provide some insights for a human inspired motor controller .

The CNS is a complex system which allows us to move, acquire skills, and adapt those skills to a variety of contexts. Each time we reach for an object, the CNS must solve a complex physical problem, to control and coordinate our limbs during interaction with a changing environment, subject to gravity [Shadmehr and Wise, 2005].

It consists of two major parts: the spinal cord and the brain, composed by the brainstem, divided into forebrain, midbrain and hindbrain (in its turn divided into medulla and pons). All levels of the CNS participate in motor control and motor learning, and in primates they all contribute to visually grounded reaching and pointing.

The forebrain, consisting of diencephalon and telencephalon, contains several structures that play a role in motor control. The hypothalamus, which is the main controller for the body functionalities, including homeostasis, reproduction and defense. The basal ganglia, receiving an efference copy of the motor commands, and providing an internal model of the body [1]. The thalamus, integrating the distributed brain systems (cerebellum, cerebral cortex, basal ganglia, globally named “loops”) into sub-cortical motor systems.

The spinal cord contains numerous types of neurons. Among them, motor neurons send command signals to muscles via motor nerves, while sensory nerves transmit sensory feedback to the CNS, i.e. information from skin, muscles and generally all body parts. Sensory nerve fibers terminate in the spinal cord, directly connected to the brain; special sets of neurons, called central pattern generators (CPG) in the spinal cord produce motor commands underlying rhythmic behaviors. The brainstem also has motor neurons and sensory nerves, and with additional neural networks can modulate the CPG activity. Moreover, it contains reticular formations regulating reflexes, and cerebellum, which is the principal region devoted to motor control.

The cerebellum is the largest component of the brainstem motor system: its main activity is the control of posture, gait, tone and ongoing activity in muscles, reaching and pointing movements; it also accounts for limb coordination, contributes to motor skills by reducing variability in the timing of movements and in force of muscle contractions. The connections between premotor cortex and cerebellum allows planning, integrating and executing smoothly

[1] Of course, the description of the functionalities of the brain is simplified, it is meant to provide a rough idea of the CNS structure. The interested reader can refer to [Shadmehr and Wise, 2005] for a more complete dissertation of the CNS particularly for motor control.

complex movements. Moreover, the cerebellum is involved in the motor learning and in the continuous process of re-learning, adjusting and refining reflex responses.[2]

The functional organization of the CNS inspired control schemes based on optimal feedback control. The parietal cortex integrates proprioceptive and visual outcomes, as well as sensory feedback, playing a role of state estimator. Premotor cortex and motor cortex transform predictions into sets of moto-neuronal discharges, encoding for force and direction of movement. The global control strategy takes into account the internal models built by the cerebellum to correct motor commands, while optimal control is yield by basal ganglia, facilitating the integration of different modules.

3.1.2 Learning, adaptation and re-optimization

Invariance, information, feedback and optimality play a role in the selection and adaptation of any movement through evolution and development.

Children show a constant developmental refinement of their motor control capabilities: starting from simple motor reflexes, the motor control systems evolves, until motions becomes stereotypic, typically in adult age, and a certain degree of kinematics and dynamics invariance is observable.

Recent findings indicate that these stereotyped arm kinematics patterns are not prewired or inborn, but the result of constant learning during ontogenesis: infants dramatically improve their kinematic performance during their first months, but the developmental process towards stereotypic joint kinematics continues. It is not known when this learning process finally leads to adult-like motor responses and what proximal joint configurations underlie the manifestation of stable endpoint kinematics [Konczak and Dichgans, 1997]. In Figure 3.3, the development of reaching trajectories in infants is shown. Trajectories straightens in time, although the unimodal velocity profiles and the inertial variability suggest that producing straight hand path may not be the first priority of the system during motion planning, and definitely not the most important criteria of the learning process.

During goal-directed movements such as reaching or pointing, the CNS overcomes the joint-level redundancy of the human motor system by applying coordinative constraints, leading to unique movement solution, e.g. straight hand paths with bell-shaped velocity profiles [Morasso, 1983].

Stochastic optimal control and optimal feedback control have been successful in modeling human reaching movements, as a minimization of motor commands during the movements and position error at the end of the movement.

What happens to the optimization process when learning a new dynamic environment?

Traditionally, adaptation (i.e. the mechanism which is triggered when facing a new dynamic environment) has been viewed as the process of canceling the effects of novel environment, on a noise rejection basis, so as to make the movements return to near baseline conditions (i.e. trajectory in unperturbed situation). For example in [Wolpert et al., 1995], perceived kinematic error played a role during adaptation, and subjects tended to maintain a visually straight

[2] Indeed, cerebellar lesions in particular cases can have transient effects as the system re-organizes and learns new ways to control motor activity [Konczak et al., 2010].

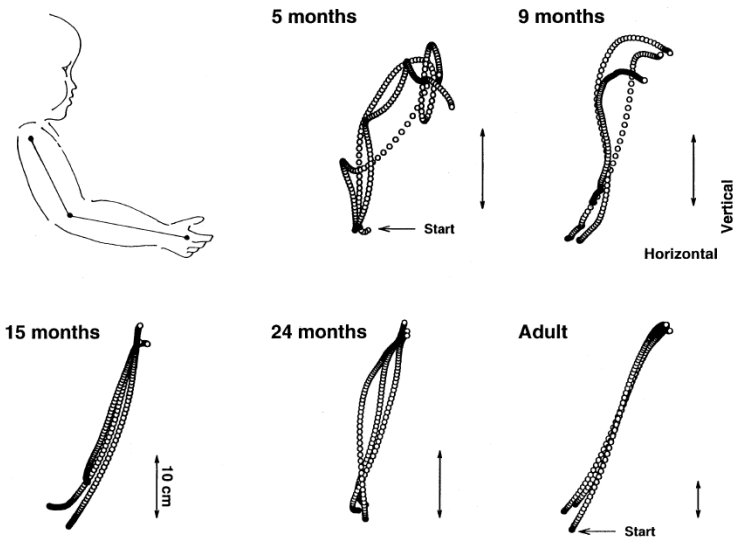


Figure 3.3: Evolution of reaching trajectories in infants. The picture shows sagittal hand paths of an infant at different developmental times, showing the progression toward the smoothing of the endpoint motion (image from [Konczak and Dichgans, 1997]).

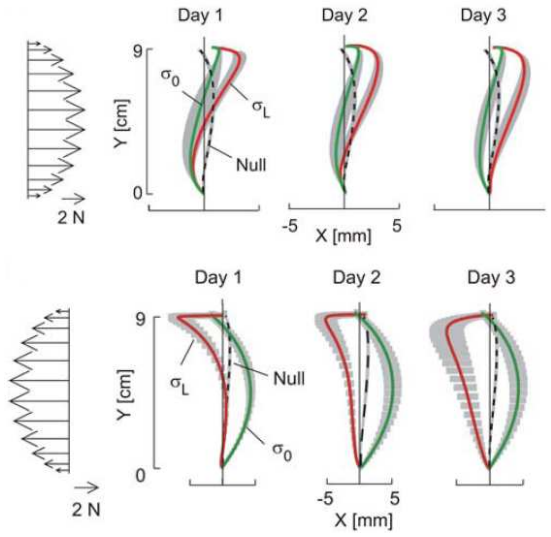


Figure 3.4: Re-optimization of reaching trajectories when adapting the arm to a new dynamic environment. The pictures show a comparison between the baseline trajectory (when the force field is null), and the ones learned when the external force field is deterministic (green line, signed σ_0) and stochastic (red line, signed σ_L), with clockwise and counterclockwise orientation (image from [Izawa et al., 2008]).

path in front of perturbations. However, in [Scheidt et al., 2000] it was later shown that kinematic errors are not necessary for adaptation, i.e. the internal kinematic and dynamic model is continuously adapted even in absence of visual kinematic feedback.

In [Mistry et al., 2008] it is shown that humans learn the new force field "dynamically" as opposed to solely rejecting the disturbances via increased stiffness and co-contraction. In detail, the authors suggested that when facing a novel dynamic environment, the CNS attempts to return to the baseline trajectory as a methodological strategy to learning an unknown dynamics. Subsequently, once the internal model is properly learned, the CNS can turn its efforts into re-optimizing the motor cost, altering the baseline trajectory if necessary. Similarly, [Izawa et al., 2008] suggested that adaptation entails accuracy and motor cost, and not the kinematic error from a desired baseline trajectory: thus, a re-optimization process computes a new optimal motor control trajectory whenever the external force field changes, as shown in Figure 3.4.

Goal-directed movements originate to acquire a rewarding state at a minimum cost, in a stochastic optimal control framework, then it is likely implausible that the brain computes a desired movement trajectory and that trajectory remains invariant with respect to environmental dynamics. Instead, when the environment changes, the learner performs at least two different computations: update the internal models (i.e. the mapping between the consequences of motor commands in terms of changes in the sensory states) and exploit the refined model to find re-optimize the trajectory planning strategy. As discussed in [Izawa et al., 2008], the cerebellum is the key structure for computing such models, since cerebellar damages produce impairments in the ability to adapt reaching to environmental changes. However, the mechanisms for the brain use this models to re-optimize movements are still uncertain.

In conclusion, motor adaptation entails both learning continuously accurate forward models, compensating environmental changes, and finding the optimal controllers that maximize rewards / minimize costs of planned movements. When facing unpredictable tasks, like picking a box without knowing its load, the CNS initially generates highly variable behaviors, but eventually converges to stereotyped patterns of adaptive responses, which can be explained by simple optimality principles [Braun et al., 2009].

3.1.3 Feedback and feedforward

Many theories of motor function are based on the concept of "optimality": they quantify task goals as cost functions, and apply the tools from optimal control theory to obtain detailed behavioral predictions, or to explain empirical phenomena. In the need of anticipating or responding optimally to trajectory perturbations, humans must combine feedback and feedforward signals.

Fast and coordinated limb movements cannot be executed under pure feedback control, because biological feedback loops are too slow (i.e. the delay in the sensory feedback cannot be neglected - it is about 60ms) and have small gains. Hence, coherently with recent theories in neuroscience [Diedrichsen et al., 2010], we believe that the CNS solves this and many other problems by combining multiple identification and control processes: precisely, exploiting integrated state estimators, internal models, and feedforward and feedback commands.

The most remarkable property of human movements is that they can accomplish complex

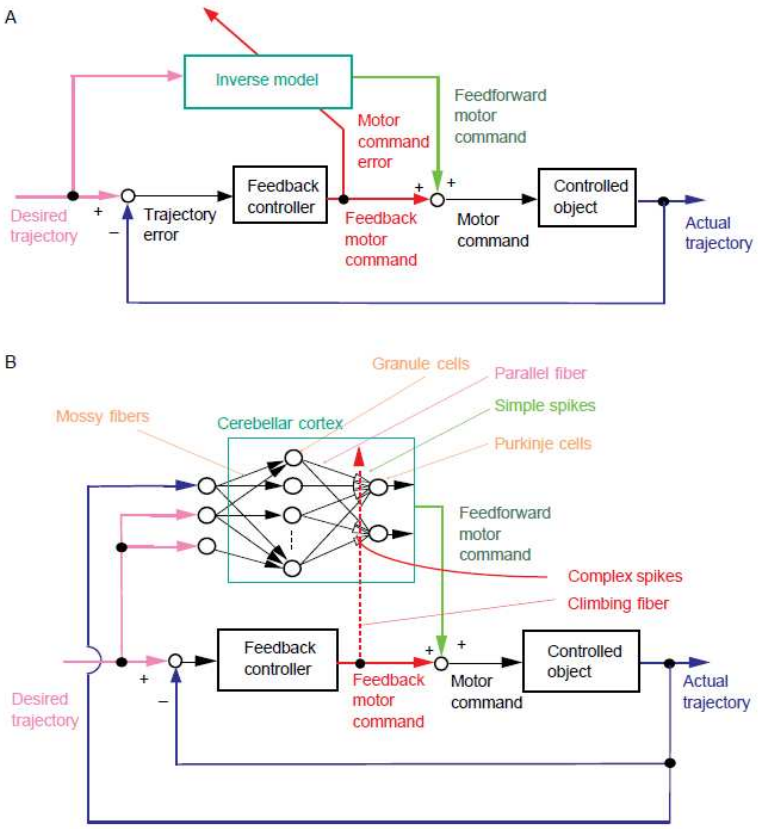


Figure 3.5: The cerebellar feedback error learning model. The "controlled object" block stands for a physical model of the limbs and body parts controlled by the CNS. The inverse model is a neural representation of the mapping between desired movement trajectory and corresponding motor commands required to attain the goal, thus it provides a feedforward command. The feedback command can be a simple PID or a more complex controller (image from [Wolpert et al., 1998]).

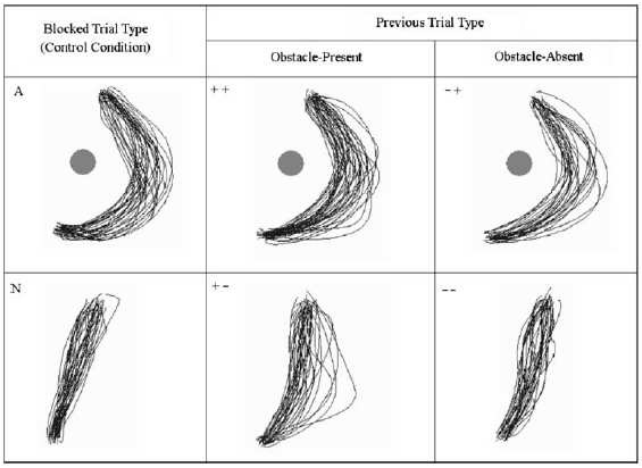


Figure 3.6: Reaching around an obstacle affects the subsequent trial when there is no obstacle. Left column shows the trajectories from the control group, reaching with or without an obstacle. The middle and right columns show data for a group where two consecutive movements were randomly with (+) or without (-) an obstacle (image from [Diedrichsen et al., 2010]).

high-level tasks in presence of disturbances, noise, delays, and unpredictable changes in the environment. Internal models support such plastic behavior, providing a fast prediction of the current system state to the motor controller (anticipating the feedback signals, which for structural and physical properties come with a certain delay both in humans and robots). But even with a quasi-perfect model of the body, open-loop approaches can only yield suboptimal performances in unstructured stochastic environments. Feedback is then necessary to explain the performance achieved by the system when adapting its strategies to tasks, environments, physical constraints, since it allows solving a control problem repeatedly rather than repeating its solution, thus affording remarkable efficiency and plasticity. Figure 3.6 reports the evidence of the continuous optimization process, which takes into account changes in the environment: when an obstacle impairs unconstrained reaching movements, the normally straight trajectory is modified to avoid the obstacle. If the obstacle is removed, the CNS does not "switch" to the control law found without obstacles, but rather adapt the previously found law to the new optimal one.

Another interesting feature of optimal feedback controllers is that desired trajectories do not need to be planned explicitly but simply fallout from the feedback control laws. This explains the trial-to-trial variability of trajectories performed by humans during repetitive tasks, like hand motion when subjects perform a goal-directed tasks: this variability cannot be explained by an optimal controller which purely executes trajectory tracking (i.e. if it tracks a pre-defined desired trajectory), but is captured by an optimal feedback controller that each time tries to minimize global task errors [Scott, 2004].

3.1.4 Internal models

Motor control must necessarily incorporate a constant adaption mechanism in order to cope with a changing environment [Shadmehr et al., 2010b]. Sensory feedback is noisy and delayed, which can make movements unstable or inaccurate, thus it is plausible that, together with feedback control relying on sensory measures, feedforward commands are employed to pre-compensate and internal models are used to predict the effect of actions on a changing body and its surrounding. Such models are usually called "forward models", as they build prediction of sensory consequences based on motor commands: as shown in Figure 3.6, they receive a copy of the motor commands, called "efferent copy", eventually access to the current state of system (even if a connection is not explicitly drawn) and produce a prediction of the sensory consequences of the action, which can be used "immediately" to refine control strategies, that is largely before the measured sensory feedback which is inevitably affected by delays. Predictions from internal models can be used to both calibrate continuously movements, and to improve the ability of the sensory system to estimate the state of the body and the environment. In particular, internal models have a learning dynamics such that prediction of the consequences of adopted controls is learned before they learn to control their actions in response to task or environmental changes [Flanagan et al., 2003]. Forward models remain calibrated through motor adaptation: that is, learning is driven by sensory prediction errors.

It is not yet perfectly clear whether the cerebellum contains an explicit representation of both forward and inverse models. While forward models seem necessary to compensate for the biological delays in the sensory apparatus, it is not equally evident if an inverse model (i.e. a model providing the neural commands necessary to achieve a desired trajectory) exists for all body parts and corresponding movements.

3.1.5 Optimality and movement duration

Optimal control theory has been recently proposed to explain the mechanisms for movement duration [Shadmehr et al., 2010a]. Human movements show several prominent features, the principal being described by [Viviani and Flash, 1995]:

- *isochrony principle*: movement duration is nearly independent of movement size
- *two-thirds power law*: instantaneous speed depends on movement curvature
- *movement compositionality*: complex movements are composed of simpler elements

Isochrony is strongly connected to the decomposition of complex movements into units of motor action. In [Viviani, 1986] it was suggested that the portion of trajectories with constant velocity gain factor may correspond to autonomous "chunks" of motion planning, i.e. elementary pieces of trajectories. The isochrony principle applies both globally to the entire trajectory (from the initial to the endpoint) and locally to the small motor actions units. An adequate theory capable of successfully accounting for all these principles, and explaining motor control features by means of motor primitives, is still under debate.

In particular, the mechanism underlying the selection of movement duration in the brain is still under investigation. During reaching, curvature path and angular velocity are closely

related, by the so called “two-thirds power law”:

$$V = KC^{2/3} = K(\frac{1}{R})^{2/3} \tag{3.1}$$

where V is the angular velocity, K a velocity gain factor, C the curvature and R the radius of curvature. This model, proposed by [Lacquaniti et al., 1983], was later extended and refined by [Viviani and Stucchi, 1992], and led to many further investigations on its relationship with the motion segmentation [Richardson and Flash, 2002] theory, based on the movement compositionality principle.

The average velocity of point-to-point movements increases with the amplitude of motion, while its duration is weakly dependent on the motion extent. It has been shown that the average velocity increase is due both by the velocity gain (depending on the amplitude of motion) and by the distribution of curvatures along the trajectory. These two factors contribute to the relative invariance of movement duration, and particularly the strong dependence of velocity gain to the amplitude of motion suggested that *isochrony* could best describe this experimental cue.

Most studies in movement duration are grounded on Fitts and Schmidt’ laws [Fitts, 1954, Schmidt et al., 1979], which relates the average movement duration with the amplitude of motion and the error tolerance when reaching the target, on the basis of either a logarithmic or a linear relationship of their ratio. According to Fitts’ psychomotor model, the time required to accomplish a movement T is a logarithmic function of the following type:

$$T = a + b\log_2\left(\frac{D}{W} + c\right) \tag{3.2}$$

where a, b, c are empirical constants [Beamish et al., 2006], D is the amplitude of movement (i.e., the distance between the initial and the endpoint position), and W is the target amplitude. Parameter D represents the Euclidean distance between the initial and target points in the Cartesian 3D space. The idea is that a certain amount of time is required to perform a movement, but the more the movement has to be precise (i.e. we want to touch a pin instead of a big ball) the more time is required to “adjust” the final position to the desired. Several analysis and extensions to this law have been done [MacKenzie, 1992], in particular for 2D tasks. These and other models, such as the minimum time principle [Tanaka et al., 2006], predict movement duration correctly, but only for point-to-point movements.

More recently, in [Bennequin et al., 2009], a theory of movement timing was proposed, and it was suggested that movement time is continuously selected by the brain based on the combination of different geometrical measures along curves. This hypothesis does not contravene the description of the whole trajectory by optimization criteria: on the contrary, invariance is compatible with different optimization principles such as the minimum-jerk [Flash and Hogan, 1985] or the minimum variance principles [Harris and Wolpert, 1998], and with optimal feedback control [Todorov and Jordan, 2002], and in general can be used together with optimization principles to solve redundancy problems at the task level, or to control the optimal selection of the relevant parameters which could enhance a trajectory description.

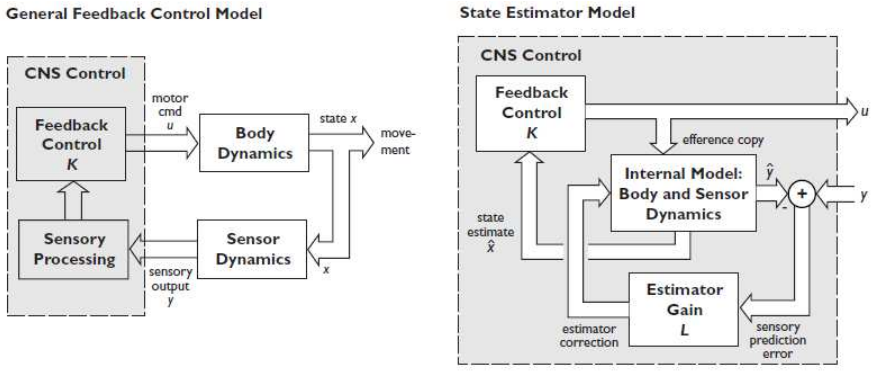


Figure 3.7: Schematics of the CNS controller proposed in [Kuo, 2005]. Right: A general feedback control model produces motor commands driving the body dynamics; sensory processing is used to compute the estimate of the body state. Left: detail of the state estimation, exploiting an internal model of the body and sensor dynamics, and the efference copy (i.e. the copy of the motor command). The integration of multiple sensors is computed optimally, in the sense that feedback gains are iteratively adjusted to minimize prediction error (image from [Kuo, 2005]).

3.1.6 Optimality and locomotion

Many researchers support the theory that optimization principles also explain the generation of gait and locomotors trajectories [Mombaur et al., 2008, Arechavaleta et al., 2008]. In humans the control of posture and goal oriented movement during locomotion is possible through a number of neural mechanisms, whose controls range to the head stabilization (creating a mobile reference frame) [Pozzo et al., 1990], to the exploitation of the vestibular system, to the generation of trajectories. Again, different models have been proposed.

In [Kuo, 2005] an optimal model for estimation and control of human postural balance is proposed. Assuming that the CNS estimates the postural "state" with a certain delay, and that this estimate is used to produce a feedback control to stabilize the system, as shown in Figure 3.7, the author propose a computation model based on stochastic optimal control. In detail, a linear quadratic controller is addressed, where the cost function is the sum of quadratic terms weighting the joints displacement from the equilibrium configuration and the neural effort, i.e. the amount of muscle activation used to stabilize the system; sensory noise is taken into account in the model, both as body model and transducer noise. Similarly, in [Lockhart and Ting, 2007] it is argued that an optimal control model with delayed feedback rule is at the basis of the neural effort produced by mammalians to keep the balance. Particularly, a feedback control law (a combination of the errors of position, velocity and acceleration of the COM of the body with respect the stable steady configuration) was optimized according to a quadratic cost function, weighting COM kinematic deviations and muscle effort (from EMG measurements).

3.2 Which is the correct "cost function"?

Stochastic optimal control theory provides an elegant mathematical framework for describing movements: by the notion of "motion criteria" transposed into "cost function" or "reward func-

tion" we can explain why a limb performs a certain trajectory among all possible options, while the solution of the optimal control problem yields the laws generating the observed behaviors.

The crucial point in this approach is the choice of the cost function to be minimized (or the reward function to maximize). Computational neuroscience does not provide a unique answer to this issue.

To tackle this problem, two are the main approaches which can be identified in literature.

The first is to try to identify the cost function by means of *inverse optimal control*, but the solution of such class of problems is very difficult to find in most situations, almost impossible when the search for a criteria is combined to systems with nonlinear dynamics. Closed form solutions exist, but in particular conditions such as in the well-known LQG formulation, where the system is linear, the cost is quadratic and the stochastic variables have Gaussian distributions. Recently, it was proposed as a promising approach to transfer biological motions into robots [Mombaur et al., 2010].[3]

The second approach consists in making some hypotheses on the structure of the cost, and trying to validate the model by comparing the predicted trajectories with experimental data. This is the most adopted choice, because the leading assumptions on the cost function are mainly inspired by cues emerging from human observed behaviors. For example, bell-shaped velocity profiles during point-to-point movements can be achieved by minimizing the jerk [Flash and Hogan, 1985] or high-order derivatives of the position; muscular inactivation can be explained by an absolute-like term in the cost function to be minimized [Berret et al., 2008]. However, it often happens that different models are suggested to explain certain behaviors, and that despite the variety of principles proposed in the models, it is difficult to confute the soundness of one model against the others: they usually provide solid arguments, and sometimes the model itself is so obvious that one may find it more appealing over the others just because of its implementation.

A clarifying example of the aforementioned arguments is given by the numerous models attempting to unveil point-to-point movements.

3.2.1 Minimum jerk

The experimental evidence is that goal-directed movements such as reaching or pointing result in straight hand paths with bell-shaped velocity profiles.

On this basis, in 1985 Flash and Hogan proposed the *Minimum Jerk Model* (MJM)[4] to describe the planar trajectories of the human arm while performing unconstrained point-to-point movements. The cost to be minimized is:

$$\mathcal{J} = \frac{1}{2}\int_0^T \left(\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2\right) dt \tag{3.3}$$

where T is the fixed duration of movement, while $x(t), y(t)$ define the time-varying hand position with respect to a fixed Cartesian coordinate system [Flash and Hogan, 1985]. In some

[3] In [Mombaur et al., 2010], all computations have been performed offline, without considering the physical robotics platform. However, the authors point out that the time to compute the optimal trajectory is lower than a standard delay of a humanoid before it starts walking. More specific details in this regard are not given.

[4] The jerk is the third derivative of the position, or the derivative of the acceleration.

peculiar conditions, closed-form solutions for $x(t), y(t)$ can be found. For example, given the boundary conditions $x(0) = x_0, y(0) = y_0$, with zero velocity and acceleration at the beginning and end of the trajectory, $\dot{x}(0) = \dot{y}(0) = \ddot{x}(0) = \ddot{y}(0) = 0$, $\dot{x}(T) = \dot{y}(T) = \ddot{x}(T) = \ddot{y}(T) = 0$, the explicit solution is:

$$\begin{aligned} x(t) &= x_0 + (x_0 - x_f)(15\tau^4 - 6\tau^5 - 10\tau^3) \\ y(t) &= y_0 + (y_0 - y_f)(15\tau^4 - 6\tau^5 - 10\tau^3) \end{aligned} \tag{3.4}$$

where $\tau = t/T \in [0, 1]$. Extensions to non-null boundary velocities, path segmentation through multiple "via points" easily follow.

It must be noted that trajectories are purely kinematics and are completely independent of the dynamics of the arm. However, the trajectories predicted by the MJM are straight-line Cartesian paths with bell-shaped velocity profiles, which is consistent with the experimental data for rapid human movements in absence of accuracy constraints.

The MJM is defined in an extrinsic-kinematic space (i.e. Cartesian space). An analogous model, defined in the arm joint space, was proposed in 1995 by Rosenbaum et al., where the function

$$\mathcal{J} = \frac{1}{2} \int_0^T \sum_{i=1}^{n} \left(\frac{d^3\theta_i}{dt^3} \right)^2 dt \tag{3.5}$$

where θ_i is the i-th joint angle. The model was called Minimum Angle Jerk Model (MAJM), always predicted straight paths in the joint space but in contrast to the MJM it allowed to represent trajectory curvatures.

3.2.2 Minimum torque change

The main defect of the MJM is that it always predict straight paths, so it does not fit to wide range movements and curved trajectories which occur for example during transverse movements, regardless of the influence of arm dynamics, arm posture, external forces, and movement duration.

Overcoming this issue, in 1989 Uno et al. proposed the *Minimum Torque Change Model* (MTCM), where trajectories were selected so as to minimize the rate of changes in torques, precisely:

$$\mathcal{J} = \frac{1}{2} \int_0^T \sum_{i=1}^{n} \left(\frac{d\tau_i}{dt} \right)^2 dt \tag{3.6}$$

where τ_i is the torque at the i-th joint of the chain [Uno et al., 1989]. The MTCM takes into account the arm dynamics, and is able to reproduce gradually curved trajectories. One controversial point in the MTCM is whether the CNS actually minimize torques, which seem to be difficult to estimate and integrate over a real trajectory (being dependent on the muscles dynamics). Computing an optimal trajectory with the MTCM is actually demanding.

Ten years later, Nakano et al. proposed a variant of MTCM, called The *Minimum Commanded Torque Change Model* (MCTCM) [Nakano et al., 1999], which provided a computable

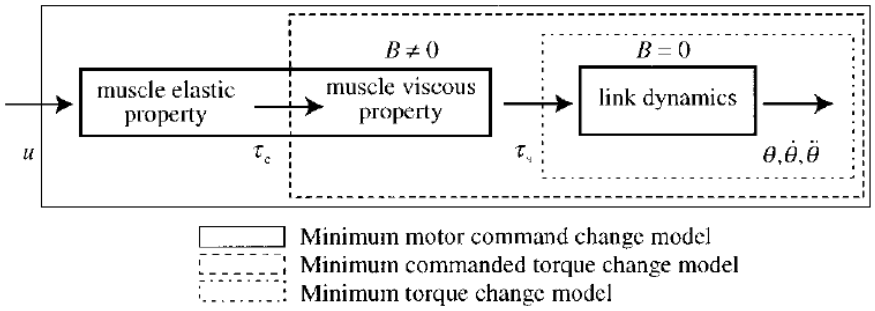


Figure 3.8: A schematic representation of the main difference between MTCM and MCTCM (image from [Nakano et al., 1999]).

approximation of the MTCM while taking into account both link and muscle dynamics. [5] A schematic representation of the two models is shown in Figure 3.8. The cost to be minimized is identical in both MTCM and MCTCM: the main difference between the two is the dynamical model of arm and muscles used to compute the torque commands. The MCTCM confirmed experimental observation in humans (see Figure 3.9), where other models such as MJM failed to reproduce trajectory curvatures depending on movement location and direction represented in intrinsic body coordinates. Their results indicated that CNS may plan optimally in intrinsic coordinates considering the arm muscles dynamics and using motor commands representations which include muscle tension.

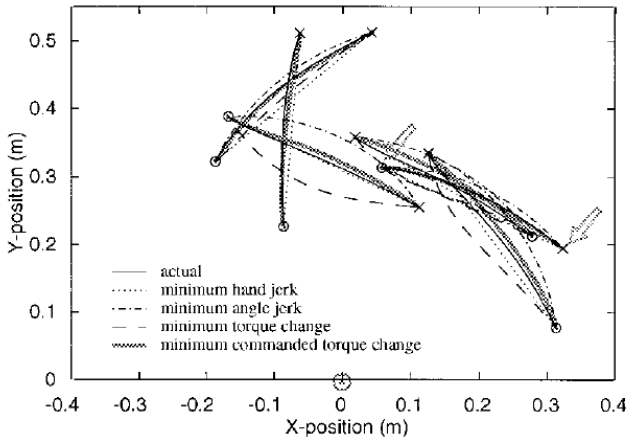
3.2.3 Minimum variance

In 1998 Harris and Wolpert observed that both eyes and arm movements were generated by neural controls corrupted by a signal-dependent noise, i.e. whose variance was proportional to the amount of control signal itself. Thus, rapid motions, requiring larger control signals, would deviate from the desired trajectory as an effect of the disturbed control, resulting at the end in unsuccessful or imprecise final positions [Harris and Wolpert, 1998]. Thus, they proposed the *Minimum Variance Theory* (MVT) which states that the accuracy in goal-directed movements is maximized by minimizing the variance of the final configuration. The MVT speed-accuracy tradeoff agrees with Fitts' law; moreover explains why repeated movements generally improve limb motion, as optimal trajectories can be learned during exercise.

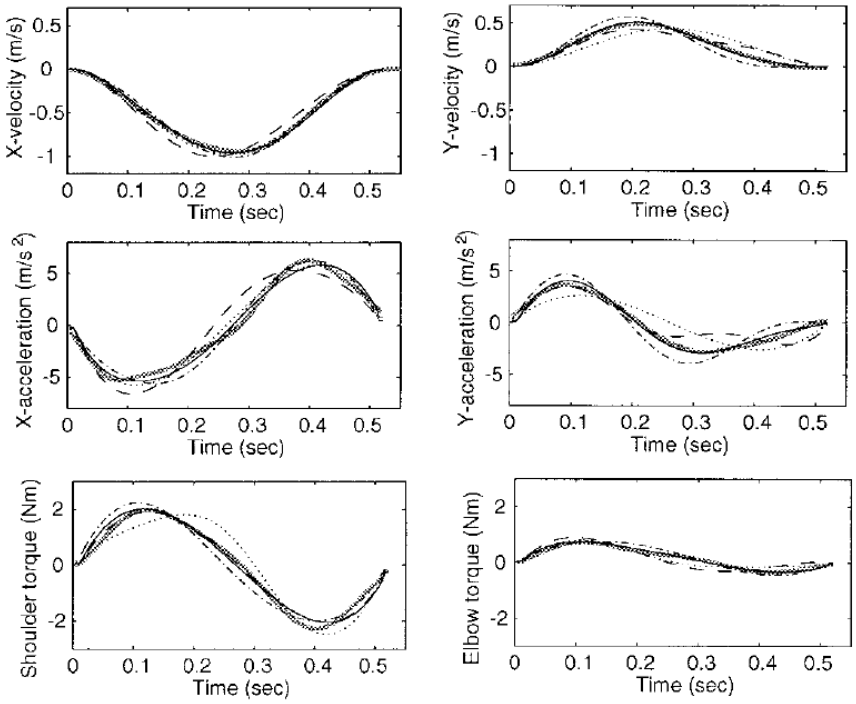
3.2.4 The Inactivation Principle

In 2008 Berret et al. proposed a cost including a term called "absolute work of forces", reflecting the mechanical energy effort of a motion. In contrast to previous models, this term is non-smooth and non-differentiable, being based on an absolute function, however it is reasonable since it is grounded on the Inactivation Principle [Berret et al., 2008]. According to this

[5] The MTCM assumes null viscosity in the arm model, while MCTCM uses a non-null viscosity matrix in calculating the joints torques, thus considering both link dynamics and muscles as controlled object in the model. For more detail, see [Nakano et al., 1999].



(a)



(b)

Figure 3.9: A comparison among different motor control models. **3.9(a)** Observed and predicted planar trajectories. **3.9(b)** observed and predicted velocity, acceleration and torque profiles during planar trajectories (images from [Nakano et al., 1999]).

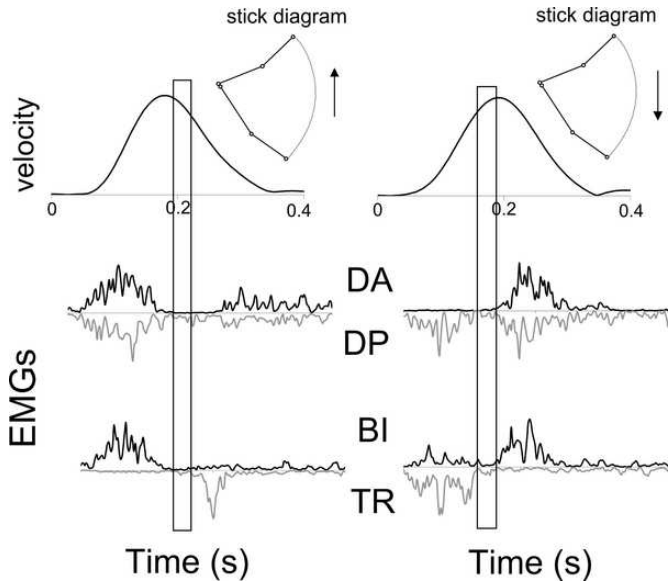


Figure 3.10: Velocity profiles of a pointing finger, and EMG recordings of the muscles during 1 DOF arm upward and downward movements. The speed profiles show muscular inactivations in proximity of velocity peaks (DA/DP=Deltoid Anterior/Posterior, BI=Biceps, TR=Triceps - image from [Berret et al., 2008]).

principle, supported by experimental observations from EMG signals (see Figure 3.10), minimizing absolute terms implies simultaneous inactivation of agonistic and antagonistic muscles acting on a single joint, near the time of peak velocity. In detail, the proposed cost is of the following type:

$$J = \int \sum_{i=1}^{n} \left|\tau_i \dot{\theta}_i\right| + \alpha_i \ddot{\theta}_i^2 \tag{3.7}$$

Notably, (3.7) accounts for a hybrid model, where both kinematics and dynamics variables are taken into account.

3.2.5 Which cost function?

Table 3.1 reports a summary of the aforementioned cost functions, and of many others, and still it is not fully comprehensive of all the models that have been suggested in literature. From the point of view of an engineer, it is difficult to choose which among the proposed models should be implemented on a robotic manipulator or sustained for future implementations.

One possible criteria to choose a model could be its explicit solution, for only those models with closed form solution could be implemented easily. This criteria rules out all models except the MJM, but explains why it is frequent to find comparisons between its predictions and experimental data from observed behaviors in humans. Overall, this model is quite appealing for roboticists too: since the analytical solution is provided (and it is also very simple), its implementation is straightforward. Furthermore, jerk minimization is beneficial if control

Criterion	Cost function $\mathcal{J}$	References
Hand jerk	$\int_0^T \dddot{x}^2 + \dddot{y}^2 dt$	[Flash and Hogan, 1985]
Angle jerk	$\int_0^T \dddot{\theta}_1^2 + \dddot{\theta}_2^2 dt$	[Wada et al., 2001]
Angle acceleration	$\int_0^T \ddot{\theta}_1^2 + \ddot{\theta}_2^2 dt$	[Ben-Itzhak and Karniel, 2008]
Torque change	$\int_0^T \dot{\tau}_1^2 + \dot{\tau}_2^2 dt$	[Uno et al., 1989]
Torque	$\int_0^T \tau_1^2 + \tau_2^2 dt$	[Nelson, 1983]
Geodesic	$\int_0^T [\dot{\theta}^\top \mathcal{M}(\theta)\dot{\theta}]^{1/2} dt$	[Biess et al., 2007]
Energy	$\int_0^T \left\lvert\dot{\theta}_1\tau_1\right\rvert + \left\lvert\dot{\theta}_2\tau_2\right\rvert dt$	[Berret et al., 2008]
Effort	$\int_0^T \mu_1^2 + \mu_2^2 dt$	[Guigon et al., 2008]

Table 3.1: Different cost functions (and related computational motor control models) for point-to-point movements.

strategies must be implemented on real devices: since the velocity and acceleration profiles are very smooth, the system is less "stressed".

From a robotics perspectives, one desirable feature is to produce controls which do not stress the mechanical structure of the manipulator or tend to minimize energy efforts. In this sense, MCTM or MCTC support this motivation, even if the biological counterpart seems a bit unclear.

It must be also pointed out that in humanoids robots motion trajectories can be controlled either with kinematics or dynamics loops, e.g. with joints velocity or joint torques. Thus, control variables must be taken into account as decision criteria.

The conclusion of this argument is twofold: first, in computational neuroscience there are many sounds models that can be used to study and describe human motions; second, many models are eligible for implementation in robotics, and the choice is basically application and robot dependent.

As if all this was not enough, recently some researchers debated the structure of this costs [Nagengast et al., 2011]. In particular, they suggested that humans not only optimize the average cost associated to a movement, but being risk-sensitive, while optimizing the mean payoff they also take into account the variability of the payoff itself. In other words, they minimize the average cost together with its mean variance. According to the authors, early experimental results suggested that the CNS acts like a risk-sensitive decision maker, which trades off the mean and the variance of movement effort. These claims suggest that the stochastic optimal control framework alone may not be sufficient to address the optimization problem behind motor control, and that other mathematical tools like decision theory and multi-objective optimization should be inevitably taken into account.

3.3 Optimality: from humans to humanoids

In the previous section an overview of the main features of human motor control was presented. Despite the incredible number of research studies and experiments, we still do not known precisely which are the correct mechanisms underlying the CNS controlling and learning complex

motor skills, except that most models assume optimal feedback control as a framework for motor control. All models are sound and could be advantageous for robotics, but require different implementations.

Nowadays, in robotics great attention is devoted to reproduce the behavior and the learning mechanisms of living creatures, developing systems that exhibit and replicate the control and learning abilities observed in animals or human beings. While modern mechatronics has reached great results (even within technology limitations), and humanoid robots exhibit shapes and structures which closely imitate humans, the great challenge is still to reproduce the learning processes and the interactions between the brain and the sensory and nervous system, that generate controls, emotions etc. and that could provide the robot with a real intelligent control: the challenge is cognitive control [Metta et al., 1999, Sandini et al., 2004].

Computational motor control models can provide useful guidance in the design of advanced control solutions for robots. Actually the two areas of research benefit from mutual achievements [Schaal and Schweighofer, 2005], because many problems faced by the primate brain in the control of movement have parallels in robotic motor control, while models and algorithms from automatic control and robotics research can bring useful inspiration, baseline performance, and sometimes direct analogs for neuroscience.

Indeed, it is our belief that only with the study of the human it is possible to build better controls for humanoid robots.

As engineers, we aim to a control policy $u(t) = \gamma(x(t), w)$, where u is a control vector, γ a control function or policy, x the system state vector and w a set of parameters which determine the policy. A policy can resort into direct control (i.e. direct generation of motor commands) or indirect control, that is the most common case in robotics, where for example desired planned trajectories are converted into motor commands (in this case, decoupling the system can notably simplify the control architecture).

The policy itself or its parameters (or both) can be fixed or adaptive. In the latter case, if the control problem is stated as an optimal control problem, an optimization routine can be set so that policy and parameters can be found, e.g.

$$\begin{aligned} &\gamma^\circ, w^\circ = \arg\min \mathcal{J}(x(t), u(t)) \\ &s.t. \qquad u(t) = \gamma(x(t), w) \end{aligned}$$

where $\mathcal{J}$ is the cost function defining the motion criteria.

In automatic controls there is a vivid research in the design of optimal control laws and the solution of such optimization problems, which sometimes can also have explicit solution (e.g. thanks to Riccati's equations, under the well-known LQG assumptions) [Vidyasagar, 1987, Bertsekas and Tsitsiklis, 1996, Sastry and Bodson, 1994]. Many application of optimal controls to robotics refer to classical approaches: for example in [Kim et al., 2000], where a robust control is found combining a stabilizing control (based on Riccati and Lyapunov equations) and a neural network accounting for unknown dynamics; in [Barambones and Etxebarria, 2002] a robust neural sliding mode controller is presented, while tracking controllers are discussed in [Sun et al., 2002, Braganza et al., 2005]. However, the high complexity and the desire to design an adaptive plastic system, as close as possible similar to the human, impose limitations on the use of classical controls, even if they provide a wide and assessed theory for stability,

convergence and optimality. The main limitation of such classical control schemes is that they are not suitable for cognitive functionalities. The biological inspiration we seek, force us to overcome the limits of traditional automatic controls, and choose new schemes to emulate the adaptation and learning capabilities of humans, combined with the optimality principles which have been observed experimentally.

Thus, we need more specific tools. In this perspective, we must design a suitable control scheme, which can guarantee all the good properties (such as stability, robustness, etc.) that traditional automatic control have; which can possibly benefit from decoupling and simplifying the systems and the control architecture, exploiting modularity; which is naturally arranged to be integrated with one or multiple learning mechanisms. Moreover, we need a mathematical tool which is able to compute trajectories and controls "optimally": it must be able to solve a general optimal control problem given the problem statement, the computational model and the cost function to be minimized.

In the following, we will briefly present the state of the art implementations of biologically inspired optimal control in robotics, particularly in humanoid robotics. Finally, we will introduce the key aspects of our proposed framework, to motivate the mathematical tool described in this work. Some insights of future developments will be also given.

3.3.1 Some implementations of optimal control models in robots

Reaching

Among the computational models presented in Section 3.2, only few have been actually implemented on real robotic platforms. An analytical solution exists (and can be easily implemented) for the MJM and the MAJM. An interesting implementation of a MJM based controller for the iCub robot can be found in [Pattacini et al., 2010]. An implementation of the MVT for a 2DOF arm was done in [Simmons and Demiris, 2005]. Models involving torques, such as the MCTCM, require the arm dynamics, thus a constrained nonlinear optimization problem must be solved, minimizing the cost function under some constraints (the nonlinear dynamics) and the boundary conditions (start and final configuration of the arm, physical limits). The solution of this class of problems is generally difficult, and depending on the problem statement there could be more than one method (or none) suited for its solution. For example, in [Kaneko et al., 2005] a solution to the MCTCM is found by means of a numerical optimization of the Euler-Poisson equation: though describing a general procedure, the authors admit the impossibility to guarantee the convergence of the routine, thus making the algorithm unsuitable for real-time planning or control in robotic applications. In [Shiller and Dubowsky, 1991] optimal control is used to compute time-optimal motions of a robotic manipulator, considering nonlinear dynamics, actuator constraints, joint limits, and obstacles. In [Zhao and Chen, 1996] an optimal motion planning is addressed to control a flexible space robot, in order to minimize the maneuvering time along with control and vibration energy. In [Mettin et al., 2010] an optimal control problem is used to find controls for ball pitching with an under-actuated 2DOF human-like arm, where in particular only the shoulder is actuated while the elbow is a passive spring with adaptive stiffness: the criteria is to maximize the ball velocity along a certain elevation angle. In [Matsui, 2008, Matsui et al., 2009] the authors propose an experimentally-

validated 3DOF model of the human arm during constrained and unconstrained reaching movements, where the criterion (i.e. the cost to be minimized) is based on energy and torque change, constrained by the hand-joint's freezing mechanism, explaining the experimental fact that the hand joint hardly changes its angle during reaching movements. Again optimal control theory is used to find the optimal trajectories of the hand during goal-directed motions.

Locomotion

The use of optimal control for humanoids has become recently popular to solve the gait and locomotion problems, and particularly both for stabilizing the robot and to plan optimally walking trajectories [Chevallereau and Aoustin, 2001, Kanoun et al., 2009]. In [Tlalolini et al., 2011] the authors suggest that human walking analysis could improve the current humanoid robots walks, an particularly to reduce the energy consumed during walking. In detail, they prove that a foot rotation subphase (specific during human fast walking) introduced in the gait contributes to the minimization of a torques-based cost, thus yielding optimal motions.
In [Arechavaleta et al., 2008] the authors investigate human goal-directed walking, with the underlying assumption that locomotors trajectories are chosen according to some optimization principle. With the attempt to identify the criteria which are optimized (duration, length, etc.), they found that the time derivative of the curvature is minimized, and that trajectories are well-approximated by the geodesics minimizing the L_2 norm of the control, shaped as clothoids[6]. In [Whitman and Atkeson, 2009] dynamic programming is used to optimize body motion, foot placement and step timing for a two link inverted pendulum model. In [Schultz and Mombaur, 2010] running is modeled as a multiphase periodic motion with discontinuities, based on multibody system models of the locomotors system with actuators and spring-damper elements at each joint; thus, running motions are generated as the solution of a an optimal control problem, based on energy criteria, solved by an efficient direct multiple shooting algorithm. In [Blair and Iwasaki, 2011] the authors suggest that the basic principle underlying animal locomotion is a mechanical rectification that converts periodic body movements to thrust force through interactions with the environment: thus, an optimal gait problem is formulated, where a quadratic cost function is minimized over a set of periodic functions subject to a velocity constraint, and the system is represented by a bilinear dynamic model, assuming small oscillations with respect to a nominal posture. In [He and Geng, 2007] optimal control is used for stable jumping of a one-legged hopping robot, with the goal to maximize energy efficiency of the motion. In [Lengagne et al., 2009] optimality is again exploited to make kicking motions more accurate, exploiting a combination of an off-line planning aimed basically at minimizing torques, with a fast re-planning process, which adapts the controls depending on the current target configuration. In detail, the cost function they try to minimize in the planning step is the sum of squared joints torques, which corresponds to the goal of improving the robot autonomy. As an optimization tool, the authors use IPOPT. The authors point out the limitations of their method, by admitting that the optimization of an instance of the problem takes about two hours CPU time (without more specific details).

[6]The clothoid or Cornu spiral is a curve, whose curvature grows with the distance from the origin.

3.3.2 Computational limits

Implementations of the aforementioned biologically inspired models on humanoid platforms face notable computational limits, since most optimal control problems incur into the COD, and even the solution of simplified problems (e.g. after strong hypotheses cutting the complexity of the model) cannot always guarantee the fulfillment of time constraints [Diehl et al., 2009]. Rather than searching for a generalized solution to the planning problem, whose computational limits make it unsuitable for online real-time control, many proposed approaches in literature usually focus on the optimization of single point-to-point movements [Simmons and Demiris, 2005, Matsui et al., 2006, Seki and Tadakuma, 2004, Tuan et al., 2008].

The corresponding optimal control problems are usually tackled via numerical methods and nonlinear programming algorithms, but the optimization process requires heavy computations and often prevents the application in real-time.

As an example, in [Tuan et al., 2008] a single movement generation is reported to take from 1 to 4 minutes, even with a fast optimizer as IPOPT [Wächter and Biegler, 2006]. In [Bauml et al., 2010], the optimization of a single trajectory for a 7DOF arm is performed in real-time, but under numerous assumptions regarding the system dynamics and kinematics, and most of all by a parallel computation on a cluster of 32 CPU cores, yielding 80% of success in the desired task.

Since closed-form solutions are utterly hard to find (impossible in many cases) approximate solutions can be addressed.

For example, Nonlinear Model Predictive Control (NMPC) methods can be used, but even the explicit precomputation of NMPC laws is prohibitive for state/parameters above $\mathbb{R}^{10}$. For example, in [Diehl et al., 2006] a NMPC with fast direct multiple shooting algorithm and several approximations were made to reduce a 20 CPU seconds computations on a 3GHz Pentium IV to $200ms$, for a 5 state $150ms$ trajectory. The reader should see [Diehl et al., 2009], where off-line precomputation, delay compensation and other techniques were surveyed, discussing reasonable compromises between computational time, convergence of the method, approximation performances and real-time guarantee.

One remarkable point is that optimal control schemes applied in robotics must take into account the platform constraints, and particularly the hardware and software limitations. For example, the generation of direct joint-level control must comply with their control loop rate (e.g. 1KHz in iCub and James); simultaneously, it requires a considerable amount of computations, both in term of time and resources. Thus, despite local computations should be preferred because they could fasten the control cycle, it may not be feasible to perform such processing on local boards (i.e. the boards directly connected to the joints) if they have limited processing capabilities. This, which is the case of iCub and James, implies that most computations must be performed by one or more PC in a cluster, which is remotely connected to the robot; in this configuration, real-time constraints cannot be a priori guaranteed, and in general the safety of this controls can be solved only up to a certain level.

3.3.3 A layered control scheme

The implementation of cognitive control for a complex humanoid robot is a challenge. An intelligent control system, modeled after biological systems and human cognitive capabilities, must possess learning, adaptation and classification capabilities, providing improved performances with respect to classical controls, but guaranteeing stability and adaptation in the presence of unknown disturbances, unmodeled dynamics (because the modeling is too difficult or because these dynamics have been neglected), and unstructured uncertainties [Metta et al., 1999, Sandini et al., 2004].

The control architecture we propose is particularly targeted for those systems where some computations can be only performed remotely. In detail, we support a layered architecture, where the task planning level is decoupled from the generation of low-level commands. The transformation between the two spaces can be performed by an intermediate level, which is basically constituted by an Inverse Kinematics (IK) module and eventually by a Forward Dynamics (FD) module, as shown in Figure 3.11.

Remark 1. *The layered architecture proposed hereinafter, reflects a traditional pattern of reaching, considered a two-stage process, where a planning phase is followed by an execution phase (and planner and controller can be two separate modules). This traditional view has been challenged by the dynamical system approach to movement control, claiming that there is no explicit trajectory planning, but rather an implicit set of trajectories which are generated by a dedicated dynamical system. An example is the VITE model [Hersch and Billard, 2006]. Here, we do not support such schemes, even if our technique is capable of coping with the dynamical systems theory.*

Figure 3.11: A conceptual scheme of a classic hierarchical control scheme for robotics. The task parameters, such as the control function to be minimized, the current status of the robot, the task goal etc. are fed to the optimal planner, which computes the optimal trajectory, typically in the operational space (e.g. Cartesian space). An intermediate layer is in charge with converting commands from operational to joint space. In this scheme, feedback loops are not voluntarily depicted.

The planning module is the core of this work. To describe the planning problem as an optimal control, the modeler has to specify:

- a family of admissible control laws
- a quantitative definition of task performance
- a compatible robot kinematics and dynamics model

The latter is usually known, since the kinematics and dynamics model of the robot is in general easy to find from the CAD specifications, thus the robot can be described by means of a set of

differential equations. In general, parameters are not perfectly known, and many dynamics cannot be fully modeled, so a supervised learning model could be used instead. The task is instead specified by a cost function: this usually comes with the computational motor control model adopted to plan the movement. The family of admissible control laws is difficult to choose, a priori, because it is strongly connected both to the robot model and to the task. Planning optimally a trajectory according to some principles, in a stochastic optimal control framework, is generally a tough problem. Without a priori hypotheses on the structure of the problem (cost, system model, constraints, etc.) it is impossible to state whether a given algorithm is guaranteed to yield a solution within a certain time. In Chapter 4 we will present a method to compute optimal trajectories, which concentrates the computation in an offline phase, but under suitable assumptions allow retrieving almost instantly the solution online, taking into account the feedback on the current status of the system. Moreover, given the particular structure of our controller, we will show that not only it is possible to learn the optimal solution online thorough an intensive learning phase, but it is possible to update the solution incrementally, combining control and adaptation, in case for example the system changes.

The IK and FD can be both learned from experimental data, or estimated if an accurate model of the robot kinematics and dynamics is available. In Chapter 5 we will discuss some possible methods that can be used, along with their advantages and disadvantages.

3.3.4 Orchestration in a control scheme: team theory

The optimal controllers inspired by the CNS and the many motor control models must be obviously integrated in the robot Cognitive Architecture (CA), that is the software system that implements the processes of the CNS [Vernon et al., 2007b] and constitute the effective "intelligence" of the robot. The CA usually consists of a set of independent modules, interconnected according to hierarchy, antagonism and cooperation, including all the relevant aspects for the modules across different application domains[7].

The primary requirement of a CA is to provide a complete perceptual representation of the robot state: the robot is a complex plant, subject to a continuous excitation of its sensory system, including all types of sensors, from proprioceptive (e.g. encoders) to visual (e.g. cameras) and tactile (e.g. skin).

There are many technological challenges in dealing with a great amount of sensor-collected data [Albers, 2002], in generating new actions and control strategies, and in the meantime in learning through data and interaction with environment, in self-organizing and adapting sensor strategies. Our belief is that motor control models, represented as stochastic optimal controllers, could be integrated and enriched in significance in the context of a team theory framework.

Team theory is an area of game theory (see [Radner, 1962, Ho and Chu, 1972]) that provides a mathematical framework which can easily describe the cooperation among devices, such as sensors and controllers, hereafter named as *Decision Makers* (DM), in highly complex systems. A team is a family of autonomous devices able to perform a task. It can often been

[7] Decision making, at any level; attentive system; prediction and internal models of self and environment; reasoning; autonomous exploration; memory and learning, etc.

viewed as a network in which each team member controls, observes, measures, gets different information, and decides to elaborate, share or transmit some personal information to the other team members, trying to maximize some common benefit or minimize a common cost.

The CNS can be modeled by a dynamic system S, resulting from the aggregation of N dynamic subsystems $S_1, S_2, \ldots, S_N$, connected among them, interacting, and evolving in a synchronous way. We conjecture that human "controllers", either performing high-level activities (e.g. learning, memory) or generating simple loops, can be represented as a family of different DM, each having a different task, processing capabilities and reacting to different stimuli. Each decision maker DM_i acts on one or more subsystems, as a controller or, when behaving as intelligent sensor, it generates the signals to be sent to the other decision makers. In all cases, DM_i influences the decisions and the behavior of the other team decision makers.

Whenever a dynamic system is controlled by a plurality of decision makers, a first problem insists in identifying the goals pursued by the decision makers. Within the human organism, it is reasonable to presume that controllers, sensors, and organs, though having different information, cooperate to the accomplishment of a common goal: wellbeing, growth (learning) and health of the living being. The existence of cooperation among the decision makers and the fact they possess "individual" information lead us to state the problem of their cooperation in the framework of team theory.

More specifically, if N decision makers $DM_i, i = 1, \ldots, N$, cooperate to the minimization of a common cost functional J, the optimization problem can be stated as follows:

$$\min_{\gamma_1,\ldots,\gamma_N} E\left\{\mathcal{J}\left[\gamma_1(I_1), \ldots, \gamma_N(I_N), \xi\right]\right\},$$

where ξ represents a set of exogenous random variables, γ_i is DM_i's decision or control functions, and I_i is its information vector. The expectation is evaluated with respect to ξ.

The solution of a team optimal control problem is a hard task, and can be solved, in general, only through approximation methods. However, there are many examples in literature of solutions exploiting functional approximators [Zoppoli et al., 2011, Baglietto et al., 2001b, Zoppoli et al., 2002, Baglietto et al., 2001a]. Even in this case these methods allow obtaining approximate solutions to such functional optimization problems, that benefit by the fundamental property of not incurring the COD phenomenon," i.e., the exponential growth of the number of parameters with the complexity (suitably measured) of the problem dealt with.

In this perspective, using ANN to approximate the optimal planning control functions seems a promising approach, which could be integrated in a more complex scenario where multiple controllers or more generally decision makers, cooperate for the achieving of a common goal.

Chapter 4

Optimal control by means of functional approximators

4.1 Planning "optimally" goal-directed movements

In robotics, the task of positioning the end effectors and to reach a goal is fundamental: whenever a robot has to move its arm to grasp an object, track a moving target, avoid collision with the environment or just explore, reaching is involved [Brock et al., 2008, Nori et al., 2007b]. Given the desired position it is common practice to plan a suitable trajectory in the Cartesian space using parameterized functions (e.g. polynomials or splines) and then to find the corresponding joint or torque commands analytically, exploiting traditional robotics schemes to perform the conversion from the operational space into joint space.

In humanoid robotics, the focus is not only on reaching the target, but on the way the target is reached, that is the criterion which the limbs accomplish while performing a movement. One of the main goals of humanoid robotics is indeed to exploit redundancy and constraints of the humanoid shape to achieve behaviors that are approximately as efficient as human movements, and to provide a testing platform for computational models, such as the ones presented in Chapter 3. In this perspective, a technique must be provided to the robot which allows finding optimal control commands for any given cost function or task, implementing different motion criteria.

Design of optimal control laws for robots is not new. In the area of automatic controls, there's a tradition in using simple and robust controls which yield to optimal control laws (i.e. solving a LQ problem by Riccati's equations). However, the biological inspiration force us to overcome the limits of traditional automatic controls, which are not sufficient to implement the adaptability of the control laws to new tasks, criteria and unknown dynamics.

The goal of the planning module is to find the optimal trajectory which makes the end-effector accomplish a certain task in an optimal fashion, i.e. minimizing a given cost functional. It must be quick, i.e. not computationally demanding in terms of time and resources, reactive to unpredictable target's changes, able to cope with the manipulator's physical limitations (singularities, joint limits, etc.) and control architecture.

A planning policy can resort into direct control or indirect control:

- direct control accounts for direct generation of motor commands, joint-level, of the following type:

$$\tau^* = \gamma(x) \quad \text{or} \quad \dot{q}^* = \gamma(x)$$

  where $\tau, \dot{q} \in \mathbb{R}^n$ are the vector of joint torques and joint velocities, respectively (thus referring to joint torque or joint velocity control schemes), and x the operational space configuration of the end-effector
- indirect control, that is the most common case in robotics, where for example desired planned trajectories are converted from the operational space into motor commands in the joint space, for example by means of Inverse Kinematics (IK) or Forward Dynamics (FD) modules:

$$\dot{x}^* = \gamma(x) \quad \text{then} \quad \dot{q}^* = \text{IK}(x^*, \dot{x}^*), \tau^* = \text{FD}(q^*, \dot{q}^*)$$

Sometimes it is better to segregate the trajectory planning from the trajectory execution, so that it is possible to tune both modules separately: in this case, indirect control is preferred.

Among different possible approaches, we decided to state the planning problem as a Finite Horizon (FH) or Receding Horizon (RH) problem, and to use functional approximation techniques in order to approximate numerically the global solution to the optimization problem, to pre-compute the optimal control laws. The RH approach becomes necessary whenever the duration of the movements cannot be predicted *a priori*. One argument is that the duration of motion can be found by computational models like the ones described in Section 3.1.5 (e.g. Fitts' law). Unfortunately, it is difficult to cast similar predictions for humanoid robots and to generalize these models for different tasks such as tracking or reaching. Nevertheless, the RH solution comes for free since it is immediately available once the generalized FH solution is found, for example by applying at each time instant only the first FH control law [Parisini and Zoppoli, 1995], as will be explained later on. Having both solutions available, we can tailor the solution depending on the task: e.g. a FH strategy is more suited for pure reaching movements, while a RH strategy for tracking targets moving indefinitely.

Trajectory planning finally consists in the computation of a time-invariant, feedback, stochastic optimal control law. In detail, a suitable sequence of neural networks is trained off line, so that they can approximate the optimal solutions of a stochastic FH control problem, which is generalized for every possible state configuration (i.e. every possible system and target states belonging to an opportune set of admissible states). The Extended RItz Method (ERIM) [Zoppoli et al., 2002] is chosen as a functional approximation technique, while the use of feedforward neural networks (thanks to their well-known approximation capabilities [Barron, 1993]) guarantees that the optimal solutions can be approximated at any desired degree of accuracy [Kurková and Sanguineti, 2005] by using a parsimonious number of parameters to be optimized.

In this way, the computation demand is concentrated in the off-line phase, while in the on-line phase only the computation of a single control law (corresponding to a neural network forward) is performed et each time. Thus the control action is generated with a very small computational effort. The feasibility of this approach has already been tested on the control of a thrusts-actuated nonholonomic robot [Ivaldi et al., 2008c]. Numerical results showing its effectiveness for different cost functions were presented in [Ivaldi et al., 2009b], for the motion

of a 2DOF manipulator. Experimental results were presented in [Ivaldi et al., 2010] for the control of the 4DOF arm of James.

If planning is performed in the operational space, an intermediate control loop is in charge with converting the desired trajectory into proper joint torque or velocity commands, taking into account the platform physical constraints. This mid-layer will be discussed in Chapter 5. Conversely, if planning is performed directly at joint-level, the system state model embedded in the problem formulation already takes into account the platform physical constraints.

While addressing the problem of finding the optimal control laws for the motion of the robot, many issues occur, in particular the problems of adapting the control laws to the time-variant, nonlinear dynamic system (as the robot is), and at the same time counteracting the disturbances due to unmodeled dynamics (friction, backlash, etc., which are in general difficult to model), delays, and the uncertainties in the model itself. To simplify the problem, some reasonable assumptions can be made.

The following are assumed to hold:

Assumption 1. *The robot's joints position, velocities and accelerations are perfectly measurable, without noise or delay.*

Assumption 2. *The robot's kinematics and dynamics are perfectly known (e.g., the Jacobian is known without errors).*

Assumption 1 is not verified on a real platform, and particularly in robots like iCub or James, where encoders provide a joint position measure only. The latter is very precise, and up to a certain degree can be treated as noiseless. In contrast, joint velocity and acceleration, which can be retrieved by double differentiation, are corrupted by a significant quantization noise, which must be filtered. Delay instead cannot be neglected. However, in this chapter we are going to address the method more formally, and these issues will be discussed with more detail in future sections. Assumption 2 holds if a fairly precise model of kinematics and dynamics exists. The kinematic model is relatively easier to write, since it relies on a Denavit-Hartenberg description of links and joints. The dynamics is fair more complicated, because of the many parameters which are more difficult to estimate, such as inertias. For example, iCub is described by a rigid-body dynamics model, whose parameters have been retrieved by the CAD model of the robot: hence, its forward dynamics is known. However, dynamics parameters are not always "fixed", but could be time-varying: e.g. mass or inertia may vary depending on the load which is applied to the manipulator.

If the robotic system is known, the following assumptions on the goal state that the target to reach (in case of goal oriented movements) is unpredictable and unknown, but can be measured at each time instant t.

Assumption 3. *The target Cartesian positions and velocities can be perfectly measured.*

Assumption 4. *The target kinematics or dynamics is unknown.*

Now let us consider the following scenario, where the robot is actively engaged to its environment: looks around trying to identify interesting objects and eventually attempts to reach them with the hands in order to grasp or manipulate; recognize people and engage in

cooperative tasks, thus driving the hand towards a desired tool to take it and deliver it to its companion.

The aforementioned goal-directed movements can be formalized as optimization problems: exploiting the models presented in Chapter 3, the goal is to find the control laws which make the robot move according to some "optimal" criteria i.e. driving the end-effector to a target position in a finite time while minimizing a certain cost function.

The following definitions are necessary:

- x_t the state vector containing the Cartesian coordinates and velocities of the end-effector, at time instant t;
- u_t the control vector, containing velocity commands in the Cartesian space;
- x^* the state vector representing the target/desired Cartesian positions and velocities in the Cartesian space; it can be fixed or time-varying, and in this case it is denoted by x_t^*;
- γ the optimal control function which steer the current state x_t to the desired.

Then the problems can be stated in the following way:

Problem 1 (*Reaching*)**.** *Find a sequence of optimal control laws $\gamma_0, \ldots, \gamma_{N-1}$ which drive the end-effector from the initial pose x_t towards the target x^*, supposed fixed, in N control instants, while minimizing a certain cost function $\mathcal{J}$.*

Problem 2 (*Tracking*)**.** *Find the optimal control law γ which at each time instant t drives the end-effector from the current pose x_t towards a target x_t^* moving unpredictably in the space, while minimizing a certain cost function $\mathcal{J}$.*

The statement of this problems is very generic, and there is no mention of models, disturbances acting on the system, or of the unpredictable actions that might change the current status of the problem. Generally speaking, we can consider the system to be modeled as

$$x_{t+1} = f(x_t, u_t, \eta_t)$$

where $x_t \in X_t \subseteq \mathbb{R}^n$ is the state vector, $u_t \in U_t \subseteq \mathbb{R}^m$ the control vector, $\eta_t \in N_t \subseteq \mathbb{R}^n$ a noise vector acting on the system; the control function can be written as

$$\mathcal{J} = \sum_{t=0}^{N-1} h_t(x_t, u_t) + h_N(x_N)$$

where h_t, h_N constitute the partial costs of the function $\mathcal{J}$ to be minimized.

Different approaches for the solution of such optimization problems have been presented in literature. A classical method is the well-known Dynamic Programming (DP) [Bertsekas, 1995], a generic global optimization procedure providing the optimal control laws $\gamma^\circ(x_t)$, by repeatedly solving the Bellman's recursive equation

$$J_t^\circ(x_t) = \min_{u_t}[h_t(x_t, u_t) + J_{t+1}^\circ(x_T)]$$

at sampled states x_t. One of the main advantages of DP is that there exists an explicit analytical solution to the control problem if it is stated under the known LQG hypotheses (linear system,

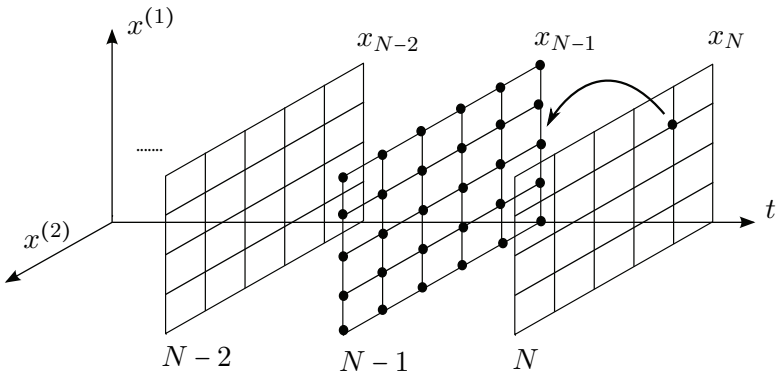


Figure 4.1: An example of bidimensional state "grid", used for Dynamic Programming. The state variable $x_t = col(x_t^{(1)}, x_t^{(2)})$.

quadratic cost function and mutually independent Gaussian stochastic variables). In the general case, one has to look for numerical approximations of the global solution.

This is usually done by sampling properly the state space and the controls, so that the functional equation that defined the DP procedure can be solved only in correspondence of a finite number of state values. Typically, states, controls, and value function are represented on a regular grid, for each control stage, and some interpolation is used to approximate these functions within each grid cell, as shown in Figure 4.1: in literature many methods can be found, from statistical-based sampling to neural approximations of the cost-to-go functions. The main drawback of DP is the computational complexity, i.e. the exponential dependence of space and computation resources needed on the dimensionality of the state, which limits its application in practice. If the sampling is uniform, and D samples are retained for each component of the state (e.g. D samples for each $x_i \in \mathbb{R}^n, i = 0, \ldots, N,$), then the number of samples grows with $(T+1)D^n$: such growth of the number of parameters restricts the application domain of DP to small dimensional problems, or require demanding resources. For example, in [Atkenson and Whitman, 2009] DP is used to find the optimal trajectories for biped walking: the authors performed the computations on a cluster of 100 nodes, each having 8 CPU cores, connected by a 16Gbit/s connection, where each grid cell was performing local optimization on a sub-sample of the state space. Also [Mandersloot et al., 2006] used DP to control velocity in bipedal walking: but despite the various simplifications introduced to avoid the COD (as the simplified model of locomotion, the disregard of dynamic effects such as swinging legs and footsteps), the authors could not go over a state of dimension $n = 3$.

Alternative solutions to DP range from the application of Pontryagin's Maximum Principle to the transformations of the functional optimization problem into a nonlinear optimizing one. Among the investigated solution, we can cite [Mitrovic et al., 2010], where iLQG was applied to compute optimal torques for the control of a planar arm, and [Diehl et al., 2006], where a multiple shooting Sequential Quadratic Programming method was used to find the minimum time control for a 5DOF simulated robotic arm.

In the following section, we will describe the theoretical tools at the base of our proposed method.

4.2 From functional optimization to nonlinear programming

Functional optimization problems deal with the minimization of functionals with respect to admissible functions, belonging to infinite-dimensional spaces. Under general hypotheses, these problems cannot be solved analytically, however it is possible to provide arbitrarily accurate suboptimal solutions by solving a suitable approximate nonlinear optimization problem. In the following one of this methods, the Extended RItz Method (ERIM), is described.

4.2.1 Stochastic functional optimization problems

The general formulation of a functional optimization problem is:

Problem 3. *Find*

$$\inf_{\gamma \in S} \mathcal{F}(\gamma) = \inf_{\gamma \in S} \underset{z}{E} \{\mathcal{J}[\gamma, z]\} \tag{4.1}$$

where $\gamma \in S$ are the admissible solutions to the problem, being S the subset of an infinite-dimensional real normed linear space H of functions $\gamma : B \subseteq \mathbb{R}^n \mapsto \mathbb{R}^m$; $\mathcal{F} : S \mapsto \mathbb{R}$ is a cost functional, while $\mathcal{J} : \mathbb{R}^m \times Z \mapsto \mathbb{R}$ is a given cost function; finally $z \in Z \subseteq \mathbb{R}^p$ is a random vector taking values from a known set Z with a known distribution, setting the problem in a stochastic context.

The target of Problem 3 is to find the optimal solution γ° among the admissible functions $\gamma \in S$, that minimizes the cost functional $\mathcal{F}(\gamma)$. The method described hereinafter is stated within a stochastic environment, however it can be applied also in deterministic situations. The stochastic formulation is more complex but is necessary in the presence of random variables acting on the system, which must be comprised in the problem formulation (e.g. noise, initial or final states with a probability distribution).

The solution of this class of problems is not easy. The analytical computation of the optimal solution of Problem 3 is feasible in few cases, again principally under LQG conditions. In all the other situations, since finding the analytical solution is hard, it is possible to use numerical techniques to approximate the desired optimal functions. The principal difficulty is that in functional problems the goal is to find one or more specific functions over an infinite dimension space: $\gamma^\circ = \arg\min \mathcal{F}(\cdot)$. Further complications arise in presence of functional dependencies: anticipating later discussions, if two or more functions must be found, e.g. $\gamma_1, \gamma_2 = \arg\min \mathcal{F}(\gamma_1, \gamma_2)$ and there exists a functional dependency, e.g. $\gamma_2 = \gamma_2(\gamma_1)$, then they must be found jointly, i.e. they cannot be decoupled. An usual approach to functional optimization is to constrain the solution to take on some structure (e.g. searching for linear solutions only), so that to obtain a suboptimal solution but that can be expressed in a simpler and closed form. Other approaches consists in giving up searching for global solutions and stop at local ones, after simplifying the problem, or aiming at the global solution via numerical approximations, using parameterized functions.

There are several ways to solve such optimization problems by means of functional approximation: the method proposed for this work constrains the functions to take on a fixed structure with a finite but sufficiently large number of free parameters. By substituting these parameterized functions in the cost functional and in the constraints expressed by the subset

S, nonlinear programming problem can be obtained, whose solution can be found by means of a proper descent algorithm. With the increase of the number of free parameters, the parameterized functions can "cover" the subset S, and the solution of the corresponding nonlinear programming problem approximates more accurately the optimal solution of the "original" functional problem.

The structure of the parameterized approximating functions is unlimited. The simplest is a linear combination of "fixed" basis functions[1]:

$$\hat{\gamma}(x) = \mathrm{col}\left(\sum_{i=1}^{\nu} c_{ij}\varphi_i(x),\ j = 1,\dots,m\right) \tag{4.2}$$

where the parameters are the coefficients $\{c_{ij}\}$ of the linear combination, and φ_i is a basis function (e.g. a sigmoid, a cosine, a Gaussian, *etc.*). Of course, $x \in \mathbb{R}^n$, and the notation $\mathrm{col}(\dots)$ accounts for all the elements of vector $y = \hat{\gamma}(x) \in \mathbb{R}^m$, such that:

$$y_j = \hat{\gamma}_j(x) = \sum_{i=1}^{\nu} c_{ij}\varphi_i(x) \tag{4.3}$$

where $\varphi_1(x),\dots,\varphi_\nu(x) \in H$ is a sequence of given basis function. Eq. 4.2 leads to the known Ritz method for calculus of variations [Ritz, 1909]. Traditionally, this method is not indicated for solving problems with a large number of variables, being subject to the well known COD issue [Bellman, 1957]: that is, the number of basis function (equivalent to the number of coefficients c_i in (4.2)) necessary to yield an approximation error (i.e. the maximum acceptable error in computing the approximation of the control functions) lower or equal than a certain ϵ may grow with n exponentially or in either way very rapidly, typically with a rate of order $O(1/\epsilon^n)$.

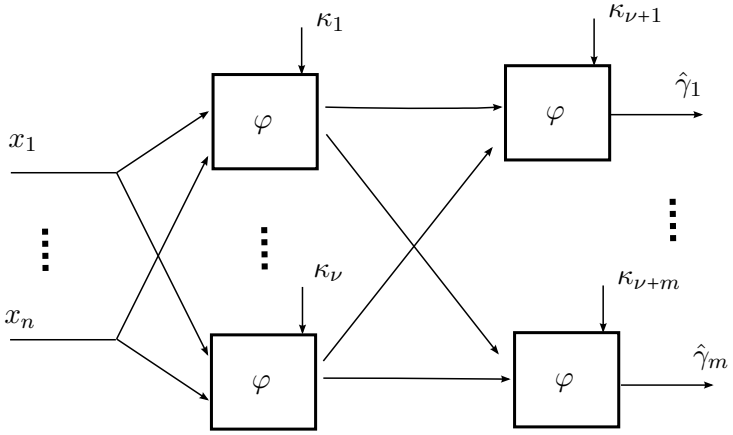


Figure 4.2: The fixed parametrized structure of an approximating function.

It is possible to lower this growth and indeed reduce the number of parameters choosing different approximating functions, in particular linear or nonlinear combinations of basis func-

[1] In the following, $\mathrm{col}(x_1,\dots,x_n) \triangleq [x_1 \dots x_n]^\top$.

tions containing free parameters (instead of fixed basis functions), in the form:

$$\hat{\gamma}(x, w) = \mathrm{col}\,(\gamma_j(x, w_j),\ j = 1, \ldots, m) \tag{4.4}$$

where

$$\hat{\gamma}_j(x, w_j) = \varphi_{\nu+j}(\varphi_1(x, \kappa_1), \ldots, \varphi_\nu(x, \kappa_\nu), \kappa_{\nu+j}),\ j = 1, \ldots, m \tag{4.5}$$

as shown in Figure 4.2.

Remark 2. *The following must be remarked:*

- *the hidden layer contains $\varphi_i, i = 1, \ldots, \nu$, while the output layer $\varphi_i, i = \nu + 1, \ldots, \nu + m$;*
- *the subscript i denoting each γ_i is used to disambiguate the basis functions, which in principle can be of different type; generally, all the functions in a layer adopt the same basis function homogeneously, for example, one can have a linear output layer and a sigmoidal hidden layer (in that case, it is simply said that the OHL-NN has sigmoidal basis functions);*
- *for a generic basis function $\varphi(\cdot, \kappa)$, $\kappa \in \mathbb{R}^K$ is the generic vector containing the free parameters; $K = \dim(\kappa)$ depends on the basis function type, for example if $\varphi(x, \kappa) = c^\top x + b$, then $\kappa = \mathrm{col}(c_1, \ldots, c_n, b) \in \mathbb{R}^{n+1}$;*
- *$w = \mathrm{col}(\kappa_i; i = 1, \ldots, \nu + m) \in \mathbb{R}^W$ is the big vector of parameters to be optimized, with $W = \sum_{i=1}^{\nu+m} \dim(\kappa_i)$;*
- *Eq. 4.4 is a two-layer NN, which is known to be a universal functional approximator [Hornik et al., 1989], i.e., there are conditions which state the existence of a sufficient number of neural units and of the corresponding optimal vector of parameters, given a desired accuracy in approximation. In general, continuous functions can be approximated to any degree of accuracy on a given compact set by feedforward NN based on sigmoidal functions, provided that the number ν of neural units is sufficiently large. Then if the function $\gamma^\circ(x)$ is unique, and is a continuous function $\mathcal{C}(X, \mathbb{R}^n)$, for every $\epsilon > 0$ there exist an integer ν and a weight vector w (and a corresponding "neural" function $\gamma_\nu(x, w)$) such that $\|\gamma^\circ(x) - \gamma_\nu(x, w)\| < \epsilon$, $\forall x \in X$. Of course, multi-layer NN can be used, guaranteeing further approximation capabilities but at the cost of a larger number of parameters.*

Common basis functions are parameterized splines, sigmoidal or radial basis functions. Using (4.4) instead of (4.2), the number of parameters to be optimized grows "moderately" (polynomially or even linearly) with n. The latter choice leads to the so-called *Extended RItz Method* (ERIM), which was first formalized in [Zoppoli et al., 2002], and refined until [Zoppoli et al., 2011]. The theoretical aspects discussing the fundamental property of the polynomial growth of W with respect to n and the approximating properties of the method were discussed in [Kurková and Sanguineti, 2005], as long as the concept of P-optimizing sequences, which will be introduced later on. The ERIM has proven to be effective in the solution of functional approximation problems in a variety of context and conditions, with stochastic constraints/costs, binary signals, linear and nonlinear systems.

4.2.2 The Extended RItz Method (ERIM)

The ERIM basically consists in constraining the admissible control function $\gamma(x) \in H$ from Problem 3 to take on a suitable parameterized but fixed structure $\hat{\gamma}(x, w) \in S$, as in Eq. 4.4, with a certain number of free coefficients $w \in \mathbb{R}^W$.

The most common structure with the ERIM is a *One-Hidden-Layer Neural Network* (OHL-NN), with the following form:

$$y = \hat{\gamma}(x, w) = \mathrm{col}\left(\sum_{i=1}^{\nu} c_{ij}\varphi(x, \kappa_i) + b_j,\ j = 1, \ldots, m\right) \tag{4.6}$$

where:

- $w \in \mathbb{R}^W$ collects all the parameters to be optimized: $w = \mathrm{col}\left(\{c_{ij}\}, \{\kappa_i\}, \{b_j\}\right)$; $i = 1, \ldots, \nu$; $j = 1, \ldots, m$; $c_{ij}, b_j \in \mathbb{R}, \kappa_i \in \mathbb{R}^K$; then $W = \nu K + m(\nu + 1)$;
- $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ so that $\hat{\gamma} : \mathbb{R}^n \times \mathbb{R}^W \mapsto \mathbb{R}^m$;
- $\varphi : \mathbb{R}^n \times \mathbb{R}^K \mapsto \mathbb{R}$ are the *parameterized basis function* (i.e. fixed structure, variable parameters)

Incidentally, $\hat{\gamma} : \mathbb{R}^n \times \mathbb{R}^W \mapsto \mathbb{R}^m$ describes a OHL-NN with linear output layer, where $\nu \in \mathbb{Z}^+$ is the number of "neurons" constituting the network (i.e. the number of "neural units" in the hidden layer) and $(\nu + m)$ the total number of "neural units". ν is also called *cardinality number of the OHL network*. W is the finite number of free parameters, and grows linearly with ν: $W = \nu K + m(\nu + 1)$. The main approximation properties of OHL-NN are discussed in [Hornik et al., 1989, Barron, 1993].

A common structure for the basis function φ is the *perceptron*, represented in Figure 4.3. A perceptron unit consists of a linear combination of x, where each variable element x_i is multiplied by a so-called "weight" (i.e. a varying coefficient), plus a bias coefficient, and their sum is processed by a so called "activation function".

Remark 3. *Except for the additional bias b_j, the main difference between (4.6) and (4.4) is that the sequence of basis functions $\varphi_0, \ldots, \varphi_\nu$ in the hidden layer (where subscript i meant that each could have a different structure) has been replaced with a set of ν parameterized basis function, each of the same type. The reason is that by increasing the cardinality ν we can avoid using different functions if the combination of the simple parameterized basis functions $\varphi(\cdot, \kappa_i) \in H, \forall \kappa_i \in \mathbb{R}^k$ has "sufficient" approximation properties. Choosing the most appropriate OHL networks is crucial: for example, when approximating the solution of Problem 3, generally nonlinear OHL networks require fewer parameters to be optimized with respect to linear ones (the approximating accuracy being equal).*

Sometimes (4.6) is enriched, for example it can have an activation function in the output layer different from a simple linear combination of the signals from the hidden layer with a

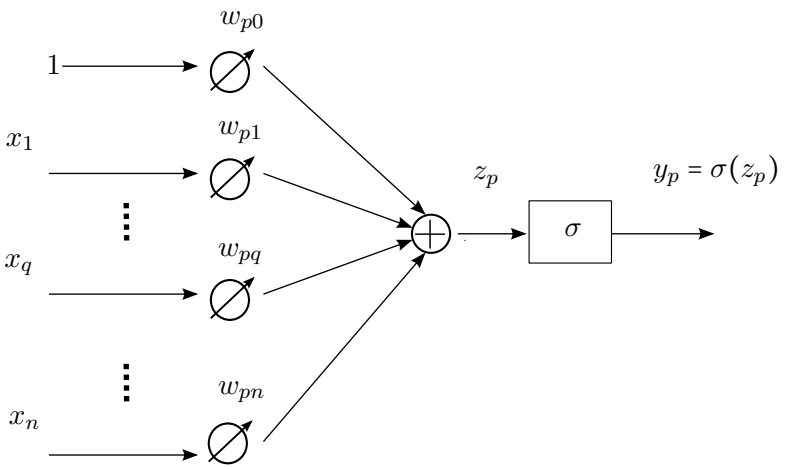


Figure 4.3: A perceptron neural unit, precisely the p-th in the hidden layer of a OHL-NN. Here, σ is the "activation function" of the unit (e.g. a sigmoid), w is a generic "weight", x is the input vector to NN, while y is the output of the hidden layer. A perceptron can be also the base unit for the output layer of a OHL-NN: in that case, $y_p = \hat{\gamma}_p(x, w_p)$, while x (in the figure) is the vector coming from the hidden layer.

bias, as anticipated in Remark 2[2].[3]

In practical implementations, Eq. 4.6 can be further slightly different. To keep the formulation as clear as possible, two important operations have been omitted from the equation (but will be discussed later on): the input and output normalization functions, i.e. the mappings $\mathcal{M}^{\mathrm{x}}, \mathcal{M}^{\mathrm{y}}$ for the range of the variables fed to the NN and exiting the NN is within the range $[-1, 1]$. Using sigmoidal as activation functions , $\hat{\gamma}$ is consequently adapted so that it complies with the constraints on its admissible values:

$$\hat{\gamma}(\tilde{x}, w) = \mathrm{col}\left(\tilde{\sigma}_j\left[\sum_{h=1}^{\nu} c_{hj}\sigma(\tilde{x}, \kappa_h) + b_j\right], j = 1, \ldots, m\right) \tag{4.7}$$

where the notations $\tilde{\sigma}$ and $\tilde{x}$ account for the output and input normalization: the input variables x are normalized from their original range to $[-1, 1]$, while the NN outputs are scaled from $[-1, 1]$ (the output range of a sigmoidal $\tanh$-based neural network) to the admissible range of $\gamma(x)$. A graphical representation of the OHL-NN in that case is shown in Figure 4.4. This operations are particularly useful when x and $y = \gamma(x)$ have a physical meaning, as will be clear in Chapter 5.

Common choices for the parameterized basis functions $\varphi(x, \kappa_i)$ are:

[2]Sometimes a sigmoidal output layer (instead of a classical linear output layer) is preferred since it naturally generates bounded values within a specific range, which can be made consistent with the real output ranges after data normalization. This choice allows removing signal constraints and not taking care of the possibility that the NN generates inconsistent values.

[3]It must be noted that the use of OHL-NN is particularly convenient, for its simplicity and for the theory assessing their approximation properties. However, NN with more than two layers can be used –the so called multi-layer NN – which in general may behave even better.

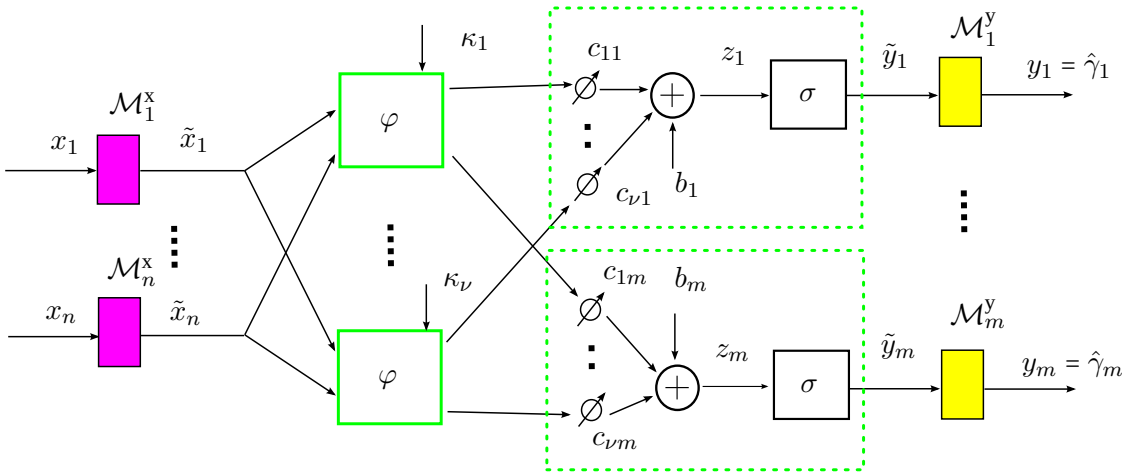


Figure 4.4: A OHL network with normalized input/output. The normalization blocks are put in evidence (magenta and yellow), and the normalized variables are put in evidence by the notation $\tilde{x}, \tilde{y}$. Single "neural" units (green boxes) as well as the parameters to be optimized (blue text) are also highlighted.

- radial constructions:
$$\varphi(x, \kappa_i) = h(\|x - \tau_i\|^2_{\Gamma_i}),$$
where $\|x\|^2_{\Gamma_i} = x^\top \Gamma_i x$, $\Gamma_i = \Gamma_i^\top, \Gamma_i > 0$, while $\kappa_i = col(\tau_i$, non-redundant elements of $\Gamma_i)$; an example is represented by Gaussian functions, like $e^{-\|x-\tau_i\|^2_{\Gamma_i}}$;
- ridge constructions:
$$\varphi(x, \kappa_i) = h(\alpha_i^\top x + \beta_i),$$
where $\kappa_i = col(\alpha_i, \beta_i)$, with $\beta_i \in \mathbb{R}, \alpha_i \in \mathbb{R}^n$; moreover, $h : \mathbb{R} \mapsto \mathbb{R}$ can be a linear function (such that, e.g., $h(x^\top \alpha_i + \beta_i) = x^\top \alpha_i + \beta_i$) or a nonlinear function, e.g. a sigmoidal one[4]. Feedforward neural networks with one hidden layer and linear activation functions are also ridge constructions.

Remark 4. *Sometimes to simply the output normalization ($\mathcal{M}^y$) a common choice is to choose a NN with sigmoidal output layer, for example using*

$$\varphi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}\,. \tag{4.8}$$

[4]Sigmoidal functions are continuous, differentiable, real-valued functions with a "S" shape and the following properties: $\lim_{z\to+\infty} \sigma(z) = 1$, $\lim_{z\to-\infty} \sigma(z) = 0/-1$. The most common "sigmoid" is the logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

defined within $[0, 1]$. Another sigmoidal function, frequently used in NN, is the hyperbolic tangent

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

defined within $[-1, 1]$.

which intrinsically generates bounded values within the range $[-1, 1]$*. Then, if* $\hat{\gamma}$ *must generate admissible values within the range* $[-U, U]$*, it is straightforward to multiply the output of the NN by* U *to obtain the desired.* [5].

Solution to nonlinear programming problems

By substituting $\gamma(x)$ with $\hat{\gamma}(x, w)$ in Eq. 4.1, the functional $\mathcal{F}(\gamma)$ becomes a function of a finite number (W) of parameters:

$$\mathcal{F}(w) \triangleq \mathcal{F}(\hat{\gamma}(\cdot, w)) \tag{4.9}$$

Therefore the original functional optimization problem is turned into a nonlinear programming one, which can be solved by means of some nonlinear programming descent algorithm. In particular, by substituting (4.6) with $\nu = 1, 2, \ldots$, a sequence of "approximating nonlinear problems" is obtained, each of them defining a nonlinear programming problem equivalent to Problem 3 and defined by the cardinality ν:

Problem 4. *Find*

$$\inf_{w \in \Psi} \mathcal{F}(w) = \inf_{w \in \Psi} \underset{z}{E} \{\mathcal{J}[w, z]\} \tag{4.10}$$

where $w \in \Psi \subseteq \mathbb{R}^W, W = W(\nu)$ *is the vector of admissible parameters, related to the constraints of* S*:* $\Psi \triangleq \{w : \hat{\gamma}(\cdot, w) \in A_\nu \cap S\}$.

In order to define the conditions for which the nonlinear programming problem can "approximate" the functional optimization one, i.e. the parameterized solution approximates the functional one, some properties of $\mathcal{F}$ and some conditions must hold.

Assumption 5. *An optimal solution* γ° *to Problem 3 exists, and the infimum and minimum are coincident:*

$$\gamma^\circ = \arg\min_{\gamma \in S} \mathcal{F}(\gamma), \ \mathcal{F}^\circ = \mathcal{F}(\gamma^\circ)$$

Assumption 6. *An optimal solution* w° *to Problem 4 exists, and the infimum and minimum of function* $\mathcal{F}(w)$ *are coincident and attained for* w°*:*

$$w^\circ = w_\nu^\circ = \arg\min_{w \in \Psi} \mathcal{F}(w_\nu), \ \mathcal{F}^\circ = \mathcal{F}_\nu^\circ = \mathcal{F}(w_\nu^\circ), \ \hat{\gamma}^\circ = \hat{\gamma}_\nu^\circ = \hat{\gamma}(\cdot, w_\nu^\circ)$$

Note that ν is explicit in $w^\circ = w_\nu^\circ$, $\mathcal{F}^\circ = \mathcal{F}_\nu^\circ$ and $\hat{\gamma}^\circ = \hat{\gamma}_\nu^\circ$ to make evidence of the dependence on a particular cardinality number ν; but the subscript ν can be dropped whenever the optimal parameterized solution is intended and it is not necessary to clarify which particular ν is used. Moreover, the following properties are required:

[5] *The guarantee of admissible outputs is fundamental when the method is exploited for generating controls for a real physical platform like a humanoid robot, where unpredictable or wrong controls could damage the system the environment, the robot itself or, the worst, people interacting with it.*

Assumption 7. *The sequence* $\{\hat{\gamma}_\nu^\circ\}_{\nu=1}^\infty$ *is such that* $\lim_{\nu\to\infty} F(\hat{\gamma}_\nu^\circ) = F^\circ$.

Assumption 8. *The sequence* $\{\hat{\gamma}_\nu^\circ\}_{\nu=1}^\infty$ *has a limit function* γ°*, i.e.* $\lim_{\nu\to\infty} \|\gamma_\nu^\circ - \gamma^\circ\| = 0$*, where* $\|\cdot\|$ *is the norm defined in the space* H.

If Assumption 5, 6, 7 and 8 hold, then formally *a sequence of problems like Problem 4, with increasing* ν*, approximates better and better Problem 3*. The sequence $\{\gamma_\nu^\circ\}_{\nu=1}^\infty$ is defined as *P-optimizing sequence*, and the corresponding OHL-NN is defined as *P-optimizing network*. It must be pointed out that the limits in Assumption 7 and 8 do not imply each other necessarily: precisely, if the *epigraphs* of the sequence of Problem 4 converge to the epigraph of Problem 3, then $\{\hat{\gamma}_\nu^\circ\}_{\nu=1}^\infty \to \gamma^\circ$ and $\{\mathcal{F}_\nu^\circ\}_{\nu=1}^\infty \to \mathcal{F}^\circ$.

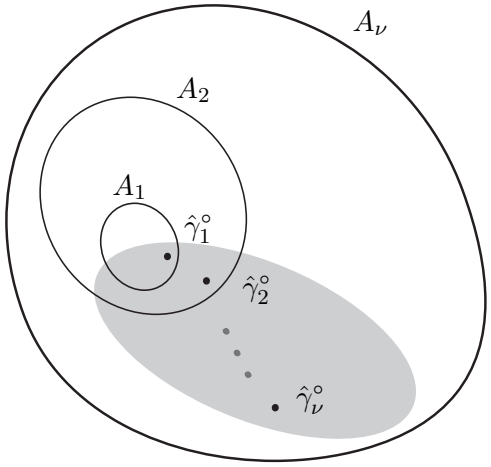


Figure 4.5: A schematic representation of $\{A_\nu\}$ and its relationship with the optimal solution $\hat{\gamma}_\nu^\circ$.

Remark 5. *Note that the aforementioned assumptions are related to the particular instance of the functional optimization problem: generally it is specified by the triple* $(H, S, \mathcal{F})$*, that is by the linear space* H*, the set of admissible solutions* S *and the cost functional* $\mathcal{F}$*. Conversely, the OHL-NN depend on the definition of the functional optimization problem through the requirements the function* $\hat{\gamma}$ *(or* $\hat{\gamma}_\nu$ *if* ν *is explicit) belong to* H*. Now the following definition are recalled from [Zoppoli et al., 2002]:*

Definition 1. A_ν *is the set containing all the functions (4.6) belonging to* H*, given a certain* ν*:* $A_\nu = \left\{\hat{\gamma}(x,w) \in H : w \in \mathbb{R}^W, W = W(\nu)\right\}, \nu = 1, 2, \ldots$*. The sequence* $\{A_\nu\}_{\nu=1}^\infty$ *has the infinite nested structure:* $A_1 \subset A_2 \subset \ldots \subset A_\nu \subset \ldots$ *(see Figure 4.5).*

Definition 2. *A sequence* $\{A_\nu\}_{\nu=1}^\infty$*, such that* $\bigcup_{\nu=1}^\infty A_\nu$ *is dense in* H*, is defined as* H-approximating sequence*, and the OHL-NN belonging each set* A_ν *are called* H-approximating networks*.*

The following assumption then relates the approximating networks of H *with the approximating functional problem:*

Assumption 9. *The decisional function $\gamma \in S$ benefits by certain regularity properties so that $\bigcup_{\nu=1}^{\infty} A_\nu$ is dense in S and the corresponding H-approximating sequence exists.*

Of course, this assumption is not sufficient to guarantee the existence of a P-optimizing sequence. Indeed, the existence of H-approximating sequences only ensures that an optimal solution to Problem 3, γ°, is an accumulation point for "some" sequence $\{\hat{\gamma}_\nu\}_{\nu=1}^{\infty}$, not necessarily for the $\{\hat{\gamma}_\nu^\circ\}_{\nu=1}^{\infty}$ determined by ERIM after solving a sequence of Problem 4.

The convergence properties of the P-optimizing sequences are fundamental to establish the complexity of the method. If the convergence speed of the sequences $\{\mathcal{F}(\gamma_\nu^\circ) - \mathcal{F}^\circ\}_{\nu=1}^{\infty}$ and $\{\hat{\gamma}_\nu^\circ) - \hat{\gamma}^\circ\}_{\nu=1}^{\infty}$, that is the speed at which the optimal solutions of a sequence of Problem 4, with $\nu = 1, 2, \ldots$ epi-converge to the optimal solutions of Problem 3, can be described by:

$$\exists p, q, p', q' \in \mathbb{R}^+ \; : \; \mathcal{F}(\hat{\gamma}_\nu^\circ) - \mathcal{F}^\circ \leq O\left(\frac{n^p}{\nu^q}\right), \; \|\hat{\gamma}_\nu^\circ - \gamma^\circ\| \leq O\left(\frac{n^{p'}}{\nu^{q'}}\right)$$

then given an approximation accuracy ϵ it is sufficient to choose a cardinality number ν satisfying:

$$\exists c, c' \geq 0 \; : \; \mathcal{F}(\gamma_\nu^\circ) - \mathcal{F}^\circ \leq c\frac{n^p}{\nu^q} \leq \epsilon, \; \|\gamma_\nu^\circ - \gamma^\circ\| \leq c'\frac{n^{p'}}{\nu^{q'}} \leq \epsilon$$

to have

$$\nu \geq \max\left[\left(\frac{c}{\epsilon}\right)^{\frac{1}{q}} n^{\frac{p}{q}}, \left(\frac{c'}{\epsilon}\right)^{\frac{1}{q'}} n^{\frac{p'}{q'}}\right]. \tag{4.11}$$

If a P-optimizing sequence verifies (4.11), then it is called *polynomially complex P-optimizing sequence*, as given the maximum approximation error ϵ (for which $\mathcal{F}^\circ, \gamma^\circ$ are approximated) the basis cardinality number ν grows at most as a power of n, which is the input dimension of the OHL-NN, hereby called *polynomially complex P-optimizing network*. The existence of a polynomially complex P-optimizing sequence makes the approximate solution of Problem 3 computationally feasible, and by (4.11) ensures that it is possible, by acting on ν, to obtain any desired degree of accuracy in the approximation by using networks containing a suitable but moderate number of parameterized basis functions.

Remark 6. *The polynomial growth of ν in the ERIM is a fundamental improvement with respect to the equivalent of the classical Ritz, where typically the growth is in the order of $O(1/\epsilon^n)$. In literature many limitations of the latter method are reported, the major being the inability to deal with the set S of admissible functions depending on a large number of variables n, and the consequent COD. Furthermore, the known error estimates (for the Ritz method) either refer to the case $n = 1$ or provide upper bounds which do not make explicit the dependence on n: hence, it is unclear to which extent it is possible to obtain arbitrarily accurate approximations by means of a "moderate" ν when the admissible decision functions γ depend on a large number n of variables.*

The upgrade of the approximating functions (from linear to nonlinear basis ones) is the key of the ERIM, which makes it avoid the COD. Several kinds of approximating networks behave

like "polynomially complex H- approximating networks", if the functions to be approximated attain some regularity conditions. In particular, for various triple $(H, S, \mathcal{F})$ it has been shown that a proper sequence of OHL-NN can be constructed by minimizing the functional over $S \cap A_\nu$: the resulting is polynomial complex P-optimizing sequence. To this extent, the functional $\mathcal{F}$ must exhibit the properties of continuity and convexity. More discussions on the convergence rates in this case can be found in [Kurková and Sanguineti, 2005, Krurková, 1997], where the upper bounds of the rates are found: the interesting property is that the error $\mathcal{F}(\gamma_\nu^\circ) - \mathcal{F}(\gamma^\circ)$ (both when the cost functional $\mathcal{F}$ is continuous only and continuous and uniformly convex) is bounded by a quantity which is at least inversely proportional to $\nu^{1/2}$, thus independently on n. Notwithstanding, a large ν is not sufficient alone to lower the approximation error, as its upper bound is also proportional to the variation norm of the optimal solution, $\|\gamma^\circ\|_{G_\varphi^n}$, being $G_\varphi^n \triangleq \varphi(\cdot, \kappa) : \kappa \in \mathbb{R}^k$ is the set of functions that can be obtained by varying the free parameters in the basis functions φ. That is, the norm is related to the basis functions φ in the OHL-NN and can be estimated if some a-priori knowledge on the admissible solutions is available. It is important to remark that the conditions which allow constructing the polynomially complex sequences, regard the functional $\mathcal{F}$ and the set of admissible function S, and are specified by geometric and regularity properties stated in a "static" context: a dynamic system evolving during the optimization process is not taken into account, or, if it exists, is implicitly considered and embedded in the properties of $\mathcal{F}$ and S.

4.2.3 A stochastic approximation technique

Problem 4 can be solved by means of a classical gradient technique, if the following fundamental assumption regarding $\mathcal{J}$ (see Eq. 4.10) is verified:

Assumption 10. *$\mathcal{J}(w, z)$ is a C^1 function with respect to w, $\forall z$.*

In the following, the various constraints on w, which define the set $\Psi : w \in \Psi$, are taken into account through penalty functions, so that Problem 4 is reduced to an unconstrained nonlinear programming problem. If "exact" constraints on w must be fulfilled, there exist specific stochastic approximation techniques that can handle them as required [Kusher and Yin, 1997]. If Assumption 10 is verified, then under some additional regularity hypotheses also $\mathcal{F}(w)$ is a C^1 function, thus its gradient can be computed, i.e. all the partial derivatives of $\mathcal{F}$ with respect to the parameters w. Then, it is possible to compute the set of optimal parameters w° by means of a classical nonlinear programming technique.

Among classical nonlinear programming algorithms, we focus our attention on gradient algorithms to introduce the concept of stochastic approximation in a simple and straightforward way. A general gradient descent algorithm is in the form

$$w(k+1) = w(k) + \alpha(k)s(k),\ k = 0, 1, \ldots \tag{4.12}$$

where $s(k)$ is a generic descent direction, and $\alpha(k)$ is a positive step-size. The idea of gradient methods is to exploit the derivative of the cost function $\mathcal{J}(w)$ to find its minimum, attained at w°, where $\nabla\mathcal{J}(w) = 0$. A descent direction $s(k)$ satisfies the condition $\nabla\mathcal{J}(w(k))^\top s(k) < 0$. The numerical procedure consists in starting from a guess or a random realization of the

parameters, $w(0)$, to generate a sequence of parameters values $w(1), w(2), \ldots, w(k), w(k+1)$ which satisfies $\mathcal{J}(w(k)) > \mathcal{J}(w(k+1))$. Typical examples of gradient descent algorithms are:

- Steepest Descent: $w(k+1) = w(k) - \alpha(k)\nabla\mathcal{J}(w(k))$
- Newton: $w(k+1) = w(k) - \alpha(k)[\nabla^2\mathcal{J}(w(k))]^{-1}\nabla\mathcal{J}(w(k))$
- Quasi-Newton: $w(k+1) = w(k) - \alpha(k)D(k)\nabla\mathcal{J}(w(k))$, where $D(k)$ is an approximation of the Hessian

Incidentally, it must be noticed that the aforementioned algorithms are designed for a numerical approximation based on a deterministic gradient, $\nabla\mathcal{J}$. In our case, the gradient is stochastic, because $\mathcal{F} = E\{\mathcal{J}\}$. In this case, the stochastic gradient (which is particularly tough to compute) can be substituted by its numerical approximation, and specifically by the gradient which is computed if a single realization of the stochastic variable occurs. Without going into further details of the algorithms, we will bring forward with our discussion using the steepest descent algorithm, that turns out to be the best choice for this problems [Spall, 2003].

The iterative steepest descent algorithm which allows solving Problem 4 is:

$$w(k+1) = w(k) - \alpha(k)\nabla_w \underset{z}{E}\,\mathcal{J}(w(k), z)\,,\ k = 0, 1, \ldots \tag{4.13}$$

Due to the general assumptions of Problem 4, it is practically impossible to compute analytically the gradient $\nabla_w \underset{z}{E}\,\mathcal{J}(w(k), z)$: indeed, at each iteration step k, the gradient of a complex function resulting from a multiple integral, related to the stochastic properties of z.

Stochastic approximation, applied to overcome such computational difficulties, consists in using $\nabla_w\mathcal{J}[w(k), z(k)]$ i.e. the gradient computed after a single realization of the stochastic variable z. Thus, instead of (4.13), the following updating algorithm is used:

$$w(k+1) = w(k) - \alpha(k)\nabla_w\mathcal{J}[w(k), z(k)]\,,\ k = 0, 1, \ldots \tag{4.14}$$

where the sequence $\{z(k)\}$ is generated randomly according to the known probability distribution of z. $\alpha(k)$ is a suitably decreasing positive step-size. The convergence of the *stochastic gradient* method is assured by a particular choice of the step size $\alpha(k)$, that must fulfill a set of conditions [Kushner and Yang, 1995].

Remark 7. *If $\mathcal{J}(w, z)$ is continuous and differentiable (see Assumption 10), under some regularity hypotheses also $\underset{z}{E}\,\mathcal{J}(w, z)$ is, and:*

$$\nabla_w \underset{z}{E}\,\mathcal{J}(w, z) = \underset{z}{E}\,\nabla_w[\mathcal{J}(w, z)]\ . \tag{4.15}$$

In practical situations, this formula can be exploited to have a certain "rough" approximation of the stochastic gradient to be used in place of the single realization $\nabla_w\mathcal{J}[w(k), z(k)]$, i.e.

$$\underset{z}{E}\,\nabla_w[\mathcal{J}(w, z)] \approx \frac{1}{Q}\sum_{q=1}^{Q}\nabla_w[\mathcal{J}(w, z(q))]\ . \tag{4.16}$$

By increasing suitably the number of realizations Q to consider at each iteration step k, it is possible to obtain a better approximation of the original gradient. This approach has been

used, for example, in [Ivaldi et al., 2008b], and similarly in [Ivaldi et al., 2009a] for the evaluation of a stochastic constraint. Of course the additional computational burden of such approximation with respect to the "single" stochastic approximation approach must be balanced with the improvement in the descent accuracy. This solution must be also balanced with the strategy for the adaptive step-size $\alpha(k)$, which is often "perturbed" as being the main object of heuristics like simulated annealing: since the gradient is already "perturbed" (even if averaged on Q realizations as in Eq. 4.16), sometimes it is difficult to assess if the gradient or the step-size are determinant for the convergence of the algorithm (4.13).

Eq. 4.14 is the simplest stochastic approximation method. Among the sufficient conditions for its convergence, which can be found in [Kusher and Yin, 1997], some concern the "shape" of the cost surface $\mathcal{F}(w)$ (but are very difficult to assess due to the characteristics of the surface itself), some the decreasing behavior of $\alpha(k)$. [Baglietto, 1998] the convergence requisites are discussed. In particular, if function $\mathcal{F}(w)$ and α verify the following assumptions:

- $\mathcal{F}(w) \geq 0, \forall w \in \mathbb{R}^W$;
- $\mathcal{F}(w)$ is continuous and differentiable, and a Lipschitz constant L exists such that $\|\nabla\mathcal{F}(w) - \nabla\mathcal{F}(w')\| \leq L\|w - w'\|, \ \forall w, w' \in \mathbb{R}^W$;
- $\exists c \in \mathbb{R}^+ \ : \ \nabla\mathcal{F}(w(k))^\top E\{[s(k)|I_s(k)]\} \geq c\|\nabla\mathcal{F}(w(k))\|^2, \ k = 0, 1, \ldots$, where $s(k)$ is a certain descent direction in (4.12) and $I_s(k) \triangleq [s(k-1), \ldots, s(0), w(k-1), \ldots, w(0)]$;
- $\exists K_1, K_2 \in \mathbb{R}^+ \ : \ E\left\{\left[\|s(k)\|^2 |I_s(k)\right]\right\} \leq K_1 + K_2\|\nabla\mathcal{F}(\omega(k))\|^2, \ k = 0, 1, \ldots$;
- the sequence of step-sizes $\alpha(k) > 0$ also satisfies:

$$\sum_{k=0}^{\infty} \alpha(k) = \infty \ , \ \ \sum_{k=0}^{\infty} \alpha^2(k) < \infty$$

then:

- the sequence $\mathcal{F}(w(k))$ converges;
- $\lim_{k\to\infty} \nabla_w\mathcal{F}(w(k)) = 0$;
- any limit point w° in sequence $\{w(k)\}$ is stationary, and $\nabla_w\mathcal{F}(w^\circ(k)) = 0$.

Of course, the properties of $\mathcal{F}(w(k))$ are related to the ones of $\mathcal{J}$. With regard to the step-size properties, it is requested that $\alpha(k)$ decreases towards zero with the increase of the iterations k: this is necessary since at the convergence point w° it could happen that $s(k) \neq 0$ (even if the exact gradient is null, as a consequence of the stochastic approximation). It is but necessary that the stepsize does not become too "small" too soon, to avoid the risk that the algorithm could be stucked in some local minima, and not be able to move out of it.

Example 1. *If $s(k) < b$ and $\sum_{k=0}^{\infty} \alpha(k) \leq B < \infty$, then $\|w(k) - w(0)\| \leq \sum_{i=0}^{k-1} \alpha(i)\|s(i)\| \leq bB, k = 1, 2, \ldots$, that is $w(k)$ is confined into a ball of radius bB and center $w(0)$: if the optimal solution was outside that ball, the algorithm would never reach that and eventually stop in a local minima. This motivates why the step-size can be neither constant nor too small.*

A step-size satisfying the aforementioned conditions (and widely used because of its simplicity) is:

$$\alpha(k) = \frac{c_1}{c_2 + k},\ c_1, c_2 > 0. \tag{4.17}$$

In literature different techniques have been suggested to accelerate the algorithm's convergence; in particular, *ad hoc* methods exists to optimize the neural networks' parameters. It is necessary to point out that heuristics frequently improve the optimization process, but their convergence and performance cannot be generally proved *a priori*.

4.2.4 Team functional optimization problems

Whenever a plurality of unknown functions is considered, Problem 3 must be extended. Let us consider a team with M agents or *Decision Makers* (DM), whose decisions are expressed by the functions $\gamma_0, \ldots, \gamma_{M-1}$. Assuming that each DM can measure x_i and use it to compute its decision $\gamma_i(x_i)$, and that the decisions of the team must minimize a certain global cost $\mathcal{J}$, the following problem can be stated.

Problem 5. *Find*

$$\inf_{\Gamma \in S_M} \mathcal{F}(\Gamma) = \inf_{\Gamma \in S_M} \underset{z}{E} \{\mathcal{J}[\gamma_0(x_0), \ldots, \gamma_{M-1}(x_{M-1}), z]\} \tag{4.18}$$

where $\Gamma \triangleq col\,(\gamma_0, \ldots, \gamma_{M-1})$ *is the set of functions* $\gamma_i(x_i) : B_i \in \mathbb{R}^{n_i} \mapsto \mathbb{R}^{m_i}$*, each belonging to an infinite-dimensional real normed linear space* H_i*,* $i = 0, \ldots, M-1$*;* $S_M \subseteq H_0 \times H_1 \times \ldots \times H_{M-1}$ *is the subset of admissible functions; each function* γ_i *has a specific argument* x_i*, which may depend on* z *as well as on the other decision functions* $\gamma_j, j \neq i$*;* $z \in Z \subseteq \mathbb{R}^p$ *is a random vector taking values from a known set* Z *with a known distribution; finally* $\mathcal{F} : S_M \mapsto \mathbb{R}$ *is the cost functional, and* $\mathcal{J} : \mathbb{R}^{n_0} \times \mathbb{R}^{n_1} \times \ldots \mathbb{R}^{n_{M-1}} \times Z \mapsto \mathbb{R}$*.*

Examples of Problem 5 are:

- *team functional optimization problems*, when several decision makers, each provided a "personal" information vector I_i (where $I_i = x_i$ or generically $I_i = g(x_0, \ldots, x_{M-1})$), cooperate to minimize a common cost functional
- *T-stage stochastic single-person optimal decision problems*, where M is the number of control/decision instants T, and random disturbances z act on the controlled system
- *Finite Horizon control problems*, where M is the number of control instants, random disturbances z represents the noise vector (e.g. the noise acting on the system), while x_i is the generic i-th state vector of the system to be controlled: in this case the system model is known and embedded in the problem formulation (transparently in (4.18))

The latter is of particular interest, since it would be the object of our further investigations.

Remark 8. *It must be remarked that the arguments* x_i *may depend on* z *and other functions* $\gamma_j, j \neq i$*, through known mappings. In fact,* x_i *is often written as* I_i *and called "information vector", denoting the aggregation of all the possible information in input to the decision function* γ_i*, representing a so-called* decision maker *(DM), or decisional entity. In this cases, it is*

useful to have a graphical representation of the information flux among the functions γ_i, as a fundamental role is played by the "partial nesting" or not of the information structure of each DM [Baglietto et al., 2001b].

The theoretical properties holding true for Problem 3 can be fully extended to Problem 5.

4.2.5 Some notes on the optimization phase

The proposed method enables NN to approximate the optimal feedback solutions, and is supposed to overcome the COD of DP-based solutions. Certain aspects deserve some attention:

- The optimal solution are approximated numerically, and it is not possible to distinguish a local from the (or a) global solution when running the optimization algorithms; for the same reason, the stability of the weight update cannot be guaranteed, since a precise analytical condition is missing. The so called "training phase" requires consistent computations and a large number of patterns for the training, depending on the problem statement. Computations are so demanding (in terms of time), that they sometimes prevent the use of the solution in real-time applications. In Section 4.5.4 a more detailed discussion on such limits is reported.

- NNs have been chosen for their approximating properties. Barron proved that neural networks are universal approximators for continuous functions, more efficient than traditional functional approximators (polynomials, splines, trigonometric expansions, etc.), even though there exists a fundamental bound on the functional reconstruction error [Barron, 1993], which is condensed in Maurey-Jones-Barrons bound. The mean integrated squared error between the approximating neural network and the target function f is bounded by

$$O\left(\frac{C_f^2}{\nu}\right) + O\left(\frac{\nu n}{L}\ln L\right) \tag{4.19}$$

  where ν is the number of neural units, n is the input dimension of the function, L is the number of training observations, and C_f is the first absolute moment of the Fourier magnitude distribution of the target function f [Barron, 1994]. In particular, with $\nu \approx C_f(L/(n\ln L))^{1/2}$ neural units, the order of the bound on the mean integrated squared error is optimized to be $O(C_f((n/L)\ln L)^{1/2})$. In [Niyogi and Girosi, 1996], similar results are shown for Gaussian radial basis functions (RBF) networks, in particular the generalization error is bounded by

$$O\left(\frac{1}{\nu}\right) + O\left(\left[\frac{\nu n \ln(\nu L) - \ln\delta}{L}\right]^{1/2}\right) \tag{4.20}$$

  where n is the number of inputs, ν the number of neural units, L is the set of training observations, and δ is a positive real number, with $0 < \delta < 1$. A frequent heuristic to determine an approximation of the optimal number of neural units ν for a given number of training observations L is also given by $\nu \propto L^{1/3}$. This result takes into account the

compromise between the minimization of the generalization error (which would require high numbers of ν) and of the estimation error (which would require a low number of L). As a rule of thumb, in other works (for example [Barambones and Etxebarria, 2002]) reasonably good results are usually obtained if the number of neural units is roughly two-three times the order of the system. Since in our framework the functions to approximate are the unknown, it is not possible to make any guess, in particular on C_f (Eq. 4.19). The underlying assumption is that an optimal set of parameters exists, given a desired approximation error. In practical terms, multiple trainings are usually performed, and the number of neurons is "manually" adjusted .

- A two-layer (or One-Hidden-Layer) neural network can be used to approximate any non-linear function, with a suitable number of neural units. In [Nguyen and Widrow, 1990] a method is proposed for the initialization of the weights in order to reduce the training time. The basic idea is that picking initial weights so that the hidden units are scattered in the input space substantially improves learning speed of networks with multiple inputs. In the following formulation, the elements of the input vector (x_t for example) take values from the range $[-1, 1]$ (the so called *input normalization*). Considering the output y of a OHL neural networks with sigmoidal activation function, $y = \sum_{i=0}^{\nu-1} v_i \sigma(x w_i + b_i)$ where ν is the number of neurons in the hidden layer, and N is the dimension of the input vector x (so $x \in \mathbb{R}^n$), then calling $rand(a, b)$ the operation which extracts a uniform random number within a certain range $[a, b]$, weights and biases are randomly initialized in the following way:

$$|w_i| = \nu^{\frac{1}{n}} \ , \ b_i = rand(-|w_i|, |w_i|)$$

  The authors in [Nguyen and Widrow, 1990] also suggest that a certain overlap between the intervals must be provided, by setting the magnitude of w_i to $0.7\nu^{\frac{1}{n}}$. Nguyen's formulation was proposed for neural networks approximating SISO (Single-Input-Single-Output) and MISO (Multiple-Input-Single-Output) functions. In particular, the exact formula is provided only for the first case, while the latter is "solved" by simply scaling the range of admissible values with the magnitude. Extensions to the MIMO (Multiple-Input-Multiple-Output) can be easily found. For example, a possible adaptation to the multiple dimension case is to choose weight w and bias b as:

$$b = urand(0.7\nu^{(1/n)}) \ , \ w = urand((0.7/n)\nu^{(1/n)}) \tag{4.21}$$

  where ν, n are the number of neurons and the number of inputs of the NN, respectively, and $urand(a)$ is a function extracting a random value within the range $[-a, a]$.

- The choice of the approximating functions is critical. Alternatively to NN, one could use other functional approximators, like Gaussian Mixture Models (GMM) or Support Vector Machines (SVM). However, there is no trivial way into applying SVM in our current approach. Firstly, SVM is meant as a supervised learning method, i.e. it tries to find a function $f(x, \alpha)$ (with alpha being it's parameters) that approximates best a set of labeled samples S, where $S = (x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the i-th input vector and y_i the i-th label (desired output). SVM basically tries to find a solution that

minimizes the error between the predicted $\hat{y}$ and the actual y, by means of a quadratic optimization problem, which is convex (hence it yields a single optimal solution). This is incompatible with our approach, where there is no such thing as a desired output and therefore neither an error; there is only the cost functional $\mathcal{F}$. Moreover, the canonical SVM expects to be trained on a batch of labeled samples, whereas in our method samples of the stochastic variables are fed into the 'learning' algorithm one-by-one, after which the parameters are updated. A possible setting in which SVM could easily be applied, is if a set of optimal control commands u_t would be known for a given set of input vectors x_t, so that a certain training set could be created, and the SVM could then be trained on this set of labeled samples.

4.3 Finite and Receding Horizon control problems

4.3.1 Applying the ERIM to solve a T-stage stochastic optimal control problem

In the following the ERIM is applied to the solution of general T-stage stochastic optimal control problem, i.e. a problem where a sequence of T optimal control functions minimizing a certain cost functional has to be found. Typical problems are the control of teams of cooperating agents, decisional problems where multiple entities play a role, but more frequently to the Finite Time or Finite Horizon control.

Here we state a T-stage stochastic optimal control problem, where the goal is to find the T optimal control laws that steer the dynamic system from an initial known state x_0^* to a final desired one x_T^*, by minimizing a suitable cost function $\mathcal{J}$.[6]

Problem 6. *Given known boundary conditions, i.e. fixed initial state $x_0 = x_0^*$ and final state x_T^* to reach in T stages, for the system:*

$$x_{t+1} = f_t(x_t, u_t, \eta_t) \ , \ t = 0, 1, \ldots, T-1 \tag{4.22}$$

where $x_t \in X_t \subseteq \mathbb{R}^n$, η_t is a stochastic variable with known distribution, and the controls are subject to the following

$$u_t^\circ = \gamma_t^\circ(x_t) \in U_t(x_t) \subseteq \mathbb{R}^m \tag{4.23}$$

find a sequence of optimal control functions $\gamma_0^\circ(x_0), \ldots, \gamma_{T-1}^\circ(x_{T-1})$ minimizing the cost functional

$$\mathcal{F} = \underset{\eta_0,\ldots,\eta_{T-1}}{E}\{\mathcal{J}\} = \underset{\eta_0,\ldots,\eta_{T-1}}{E}\left\{\sum_{t=0}^{T-1} h_t(x_t, \gamma_t^\circ(x_t)) + h_T(x_T)\right\} \tag{4.24}$$

Eq. 4.22 is a set of equations describing a discrete-time stochastic dynamic system (in general, nonlinear), where at the time instant t, x_t is the state vector, which may be taking values from a finite set $X_t \subseteq \mathbb{R}^n$, starting from a known initial state $x_0 = x_0^*$; u_t is an admissible control vector, constrained to take values from a finite set $U_t(x_t) \subseteq \mathbb{R}^m$; η_t is an exogenous

[6] The following is a fundamental problem, addressing the main aspects of a FH stochastic optimal control problem. The boundary conditions (i.e. fixed initial and desired state) but make it unsuitable for RH controls, as will be explained later on.

variables vector. T is a known positive integer. It must be remarked that the constraints (4.23) can be expressed by means of additional penalty functions to be added to the main cost function, that is turning a hard constraint into a soft one. $\eta_0, \eta_1, \ldots, \eta_{T-1}$ are random disturbances, and if Dynamic Programming (DP) was applied to solve the problem later on, the mutual independence of each random vector would be required. If the ERIM is applied, then this assumption can be removed.

The state vector x_t, at time instant t, is perfectly known or measurable, thus it can be used to design a feedback control law $u_t = \gamma(x_t)$. We remark that a feedback control law is necessary whenever noise and generally disturbances act on the system, or if we want to counteract to unknown or unmodeled dynamics in the system itself, which could drive it into undesired states or even compromise its stability. The cost function $\mathcal{J}$ is generally of the type:

$$\mathcal{J} = \sum_{t=0}^{T-1} h_t(x_t, u_t) + h_T(x_T) \tag{4.25}$$

where the final term $h_T(\cdot)$ usually weights the disparity between the system state at the end of the maneuver and the desired state; and a sum of terms $h_t(\cdot)$, which can weigh the disparity between the desired and the system state during the maneuver, the trajectory shape, or the effects of controls, their consumption etc. Note that because of (4.23), $\mathcal{J}$ is a "function of functions", and precisely:

$$\mathcal{J} = \mathcal{J}(\gamma_0, \ldots, \gamma_{T-1}) \tag{4.26}$$

Eq. 4.25 has a stochastic nature, because the system is affected by noise, so the formulation of the problem takes into account the minimization of the expected value of $\mathcal{J}$, i.e. the cost functional $\mathcal{F} = E\{\mathcal{J}\}$.

Remark 9. *It is important to point out that in Problem 6 the initial system state x_0 is fixed to x_0^*, because the goal was to find a sequence of controls for specific boundary conditions. The final state x_T is not constrained to take on a desired value, but a desired value x_T^* is specified. Typically, the desired state (i.e. the state to reach) is expressed by penalty functions in the cost function to be minimized, such as:*

$$h_T(x_T) = V_T(x_T - x_T^*)^2$$

where $V_T \in \mathbb{R}^{n \times n}$, $V_T = V_T^\top > 0$ is a gain matrix weighting each component differently. In the following, we will not explicitly state the dependency of system trajectories, cost function, etc. to the desired final state, to keep all formulas lighter.

It is well known that Problem 6 can be solved "analytically" through the DP only if suitable conditions hold, typically the known LQG hypotheses (linear system, quadratic cost function and mutually independent Gaussian stochastic variables). In the general case, one has to look for approximate solutions. This is usually done by discretizing or sampling properly the state space and the controls, so that the functional equation that defined the DP procedure can be solved only in correspondence of a finite number of state values: for each control stage, a uniformly (or not) sampled state space is found (an example was shown in Figure 4.1).

Alternatively, the ERIM can be used to approximate the global control laws, exploiting the recursive formulation of the problem. For a better comprehension of the procedure described hereinafter, it is useful to refer to the graphical representation of the evolution of the trajectory x_t in time, making explicit the links between dynamic system and control laws (which will be substituted with neural controllers, following the procedure described in Section 4.2). At time instant t, the combination of Eq. 4.22 and 4.23 can be represented by the system and control blocks shown in Figure 4.6. If we unfold system and controls in time, and replicate the basic couple of blocks for each time instant $t = 0, \ldots, T$, a "chain" is obtained, as shown in Figure 4.7. Note that the first control function $u_0 = \gamma_0(x_0)$ can be simply denoted by u_0, since the initial state is fixed in Problem 6, $x_0 = x_0^*$: thus, u_0 can be determined exactly, and the problem actually concerns only $\gamma_1, \ldots, \gamma_{T-1}$.[7]

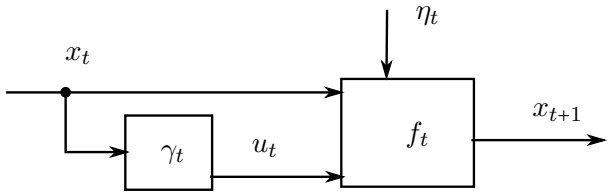


Figure 4.6: The t-th elements couple, when the control task is "unfolded" in time. The system and the control blocks , f_t and γ_t respectively, refer to Eq. 4.22 and 4.23.

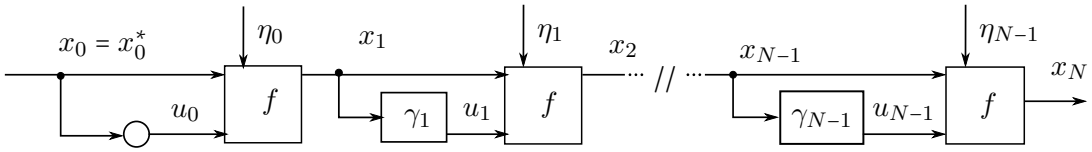


Figure 4.7: The "chain" of state and control blocks, unfolded in time, for Problem 6. Note that γ_0 is not indicated, since x_0 is fixed, $x_0 = x_0^*$. Thus, $u_0 = \gamma_0(x_0)$ is not effectively a control law spanned over X_0, but a simple fixed vector $u_0 \in \mathbb{R}^m$, being $u_0 = \gamma_0(x_0^*)$.

Applying the ERIM to solve Problem 6 consists basically in constraining the admissible control functions $\gamma_0(x_0), \ldots, \gamma_{T-1}(x_{T-1})$ to take on the fixed parameterized structure of OHL-NN (see Eq. 4.6): $\hat{\gamma}_{\nu_0}(x_0, w_0), \ldots, \hat{\gamma}_{\nu_{T-1}}(x_{T-1}, w_{T-1})$. Since the OHL-NN have the same structure at each control stage t, and each one is completely specified by the vector of parameters w_t. It must be noted that $\dim(w_t)$ is related to the cardinality number ν_t; the dependence of ν_t on t is due to the fact that the regularity properties of the function ν_t to be approximated may be time-varying. So it can be possible that $\nu_j \neq \nu_i, j \neq i$, but generally the contrary holds

[7] In the following we will still use γ_0 to keep the formulation universal: in a general context, x_0 can be unconstrained, or taking values from a variable set.

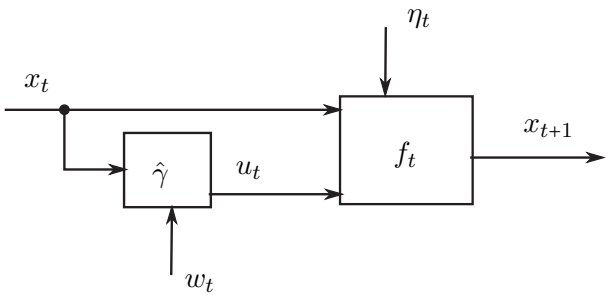


Figure 4.8: The t-th elements couple of Figure 4.6, after the application of the ERIM. Note that γ_t is being approximated by $\hat{\gamma}(x_t, w_t)$. The structure of $\hat{\gamma}$ is fixed, while the control law is completely specified by w_t.

true, so it is possible to drop the subscript ν_t from $\hat{\gamma}$ and let w_t specify the t-th control function:

$$u_t = \hat{\gamma}(x_t, w_t),\ t = 0, \dots, T-1 \tag{4.27}$$

as shown in Figure 4.8 for the t-th element of the chain. Note again that the first control function $u_0 = \gamma_0(x_0)$ does not need to be replaced with the OHL-NN, since the initial state, in this context, is a fixed vector $x_0 = x_0^*$, but we will still mention it to keep a more general formulation. Indeed, this is valid only if the initial state is fixed: if x_0 is a stochastic variable with its own probability density, then $\gamma_0(x_0)$ must be replaced with $u_0 = \hat{\gamma}(x_0, w_0)$ as the other functions (because u_0 obviously depends on x_0). This case will be object of interest later on.

By substituting (4.27) in (4.24), the general cost function $\mathcal{J}$ (the "function of functions" as in (4.26)) is turned into a function which is only dependent on a finite number of real variables:

$$\mathcal{J}(\tilde{w}) = \mathcal{J}(u_0, w_1, \dots, w_{T-1}) \tag{4.28}$$

where $\tilde{w}$ contains all the parameters to be optimized:

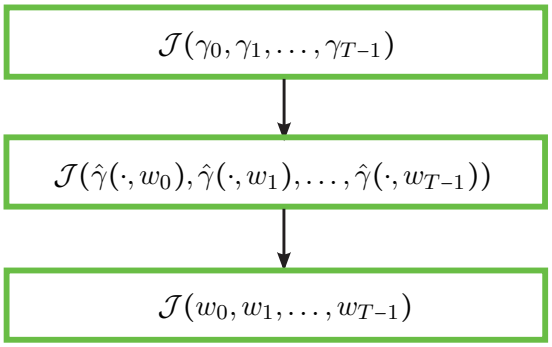
$$\tilde{w} \triangleq [u_0, w_1, \dots, w_{T-1}] \in \mathbb{R}^W\ , \tag{4.29}$$

with $W = m + \sum_{i=1}^{T-1} W_i$ (being $u_0 \in \mathbb{R}^m$ and $w_i \in \mathbb{R}^{W_i}$).

The solution of Problem 6 by means of the ERIM can be then summarized by these three main steps:

- write the cost functional $\mathcal{J}(u_0, \gamma_1, \dots, \gamma_{T-1})$
- substitute the control functions with the OHL-NN: $\mathcal{J}(u_0, \hat{\gamma}(\cdot, w_1), \dots, \hat{\gamma}(\cdot, w_{T-1}))$
- write the cost function $\mathcal{J}(u_0, w_1, \dots, w_{T-1})$

If the assumption on the known initial state $x_0 = x_0^*$ does not hold, the general formulation is:



Problem 6 is then turned into the following:

Problem 7. *Given known boundary conditions, i.e. fixed initial state $x_0 = x_0^*$ and final state x_T^* to reach in T stages, for the system (4.22), where the controls are subject to (4.27) find the vectors of optimal parameters $w_0^\circ, \ldots, w_{T-1}^\circ$, i.e. $\tilde{w}^\circ$ (see Eq. 4.29), which minimize the cost function*

$$\mathcal{F} = \underset{\eta_0,\ldots,\eta_{T-1}}{E}\{\mathcal{J}(\tilde{w})\} = \underset{\eta_0,\ldots,\eta_{T-1}}{E}\left\{\sum_{t=0}^{T-1} h_t(x_t, \hat{\gamma}(x_t, w_t)) + h_T(x_T)\right\} \tag{4.30}$$

under the constraints given by the state equation.

Gradient algorithms and stochastic approximation

As done for Problem 3- 4, it is possible to approximate progressively better Problem 6, by suitably increasing the cardinality number in the T OHL-NNs, with a sequence of unconstrained nonlinear programming problems. In analogy to the solution proposed for Problem 4, the optimal parameters of Problem 7 can be found through a nonlinear programming algorithm and a stochastic approximation technique:

$$\tilde{w}(k+1) = \tilde{w}(k) - \alpha(k)\nabla_{\tilde{w}} \underset{\tilde{\eta}}{E}\mathcal{J}(\tilde{w}),\ k = 0, 1, \ldots \tag{4.31}$$

where $\tilde{\eta} \triangleq \mathrm{col}(\eta_0, \ldots, \eta_{T-1})$. As already discussed in Section 4.2.3, it is impossible to calculate exactly all the gradient components, because of the stochastic nature of $\tilde{\eta}$. A stochastic approximation technique is then applied and the update equation becomes:

$$\tilde{w}(k+1) = \tilde{w}(k) - \alpha(k)\nabla_{\tilde{w}(k)}\mathcal{J}(\tilde{w}, \tilde{\eta}(k))\ k = 0, 1, \ldots \tag{4.32}$$

where the sequence $\{\tilde{\eta}(0), \ldots, \tilde{\eta}(k), \tilde{\eta}(k+1), \ldots\}$ is generated randomly according to the known probability density function of each $\eta_i, \forall i$. Of course, it must be assumed that function $J(\tilde{w}, \tilde{\eta}(k))$ is C^1 with respect to $\tilde{w}$ for all $\eta_0(k), \ldots, \eta_{T-1}(k)$.

In order to update the value of each parameter $w_t^i(k)$ (the i-th parameter of the vector w_t at the iteration step k), the partial derivatives of $\mathcal{J}$ with respect to $\tilde{w}(k)$ must be computed, i.e.

$$\frac{\partial \mathcal{J}}{\partial u_0(k)}, \frac{\partial \mathcal{J}}{\partial w_t(k)} = \mathrm{col}\left(\frac{\partial \mathcal{J}}{\partial w_t^i(k)}\right),\ t = 1, \ldots, T-1;\ i = 1, \ldots, W_t \tag{4.33}$$

where $w_t^i(k)$ is the i-th component of vector $w_t(k) \in \mathbb{R}^{W_t}$, and $W_t = \dim(w_t(k))$ (e.g. $W_t = (n+1)\nu + (\nu+1)m$ if the OHL-NN has ν "neurons", and each perceptron unit has the bias). The update equation for a single parameter $w_t^i(k)$ is:

$$w_t^i(k+1) = w_t^i(k) - \alpha(k)\frac{\partial \mathcal{J}}{\partial w_t^i(k)},\ i = 1,\dots,W_t\ ;\ t = 1,\dots,T-1\ ;\ k = 0,1,\dots \tag{4.34}$$

More frequently, when using NN with the ERIM, the following update equation is preferred:

$$w_t^i(k+1) = w_t^i(k) - \alpha(k)\frac{\partial \mathcal{J}}{\partial w_t^i(k)} + \eta(w_t^i(k) - w_t^i(k-1)) \tag{4.35}$$

where a regularization term is added, weighted by $\eta \in [0,1]$, as it is usually done when training neural networks.

Given (4.27), it is quite straightforward to compute the partial derivatives in (4.33):

$$\frac{\partial \mathcal{J}}{\partial w_t^i(k)} = \frac{\partial \mathcal{J}}{\partial u_t}\frac{\partial \hat{\gamma}(x_t, w_t(k))}{\partial w_t^i(k)}\ ,\ i = 1,\dots,W_t\ ;\ t = 1,\dots,T-1 \tag{4.36}$$

since $\dfrac{\partial \hat{\gamma}(x_t, w_t(k))}{\partial w_t^i(k)}$ can be easily retrieved from the known structure of the OHL-NN.

The tough part is computing $\dfrac{\partial \mathcal{J}}{\partial u_t}$: the procedure consists in a two-steps algorithm, with a *forward* and a *backward* phase. The pseudo-code is shown in Algorithm 1.

In detail, the "chain rule" is applied again, so that system and control blocks are unfolded in time, as previously done for Figure 4.7. The generic control functions γ_t are then substituted with their OHL-NN counterpart $\hat{\gamma}(\cdot, w_t(k))$: thus each control element as in Figure 4.8 is replaced with the one of Figure 4.8. Starting from $x_0 = x_0^*$ and following the connections between the blocks, one can easily compute all the trajectories of state x_t and controls $u_t(k)$, given a realization $\tilde{\eta}(k)$ of the stochastic variables. These computations make the *forward phase*. In detail, the feedback between system and neural controllers is made explicit through unfolding in time their blocks. Given the initial state x_0 the trajectory of system state and controls is:

$$u_0, u_t = \hat{\mu}(x_t, w_t(k))\ ,\ x_0, x_{t+1} = f(x_t, \hat{\gamma}(x_t, w_t(k)), \eta_t(k))\ ,\ t = 1,\dots,T-1$$

Then all the partial costs $h_t(x_t, u_t)$, $h_T(x_T)$ and the cost function $\mathcal{J}(k)$ are computed.

A *backward phase* follows, where the gradient components needed for the update algorithm (4.32) are computed. A graphical representation is shown in Figure 4.9.

To compute the gradient, the following cost-to-go function is defined, for $t = 1,\dots,T-1$:

$$\mathcal{J}_t(x_t^T, w_t^{T-1}(k), \eta_t^{T-1}(k)) = h_t(x_t, \hat{\gamma}(x_t, w_t(k))) + \sum_{i=t+1}^{T-1} h_i(x_i, \hat{\gamma}(x_i, w_i(k))) + h_T(x_T) \tag{4.37}$$

where

$$\begin{aligned} x_t^T &= \mathrm{col}(x_t, x_{t+1}, \dots, x_T) \\ w_t^{T-1}(k) &= \mathrm{col}(w_t(k), w_{t+1}(k), \dots, w_{T-1}(k)) \\ \eta_t^{T-1}(k) &= \mathrm{col}(\eta_t(k), \eta_{t+1}(k), \dots, \eta_{T-1}(k)) \end{aligned}$$

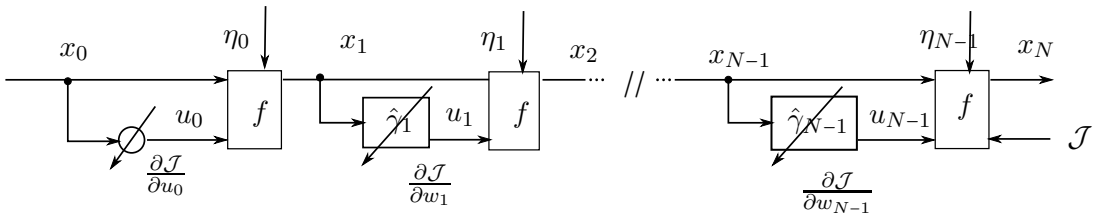


Figure 4.9: Backward phase: the partial derivatives of the cost function with respect to the outputs are back-propagated through the OHL-NNs. Inside each block $\hat{\gamma}$, the partial derivatives with respect to the parameters to be optimized are computed and used for the gradient descent. Note that to simplify the notation and the scheme, $\hat{\gamma}_t = \hat{\gamma}(\cdot, w_t)$.

The partial derivatives with respect to the parameters are then, for $t = 1, \ldots, T-1$:

$$\frac{\partial \mathcal{J}_t}{\partial w_t(k)} = \frac{\partial h_t(x_t, u_t)}{\partial u_t} \frac{\partial \hat{\gamma}(x_t, w_t(k))}{\partial w_t(k)} + \\ + \frac{\partial \mathcal{J}_{t+1}(x_{t+1}^T, u_{t+1}^{T-1}, \eta_{t+1}^{T-1}(k))}{\partial x_{t+1}} \frac{\partial f_t(x_t, u_t, \eta_t(k))}{\partial u_t} \frac{\partial \hat{\gamma}(x_t, w_t(k))}{\partial w_t(k)} \tag{4.38}$$

whereas for the first control, fixed, being $u_0 = \gamma(x_0^*)$:

$$\frac{\partial \mathcal{J}_0}{\partial u_0} = \frac{\partial h_0(x_0, u_0)}{\partial u_0} + \frac{\partial \mathcal{J}_1(x_1^T, u_1^{T-1}, \eta_1^{T-1}(k))}{\partial x_1} \frac{\partial f_0(x_0, u_0, \eta_0(k))}{\partial u_0} \tag{4.39}$$

Exploiting the chain rule, it is possible to compute the partial derivatives needed by the algorithm, by following the trajectories of u_t, x_t across the chain backward, i.e. $t = T, T-1, \ldots, 1, 0$; this operation leads to the following recursive equations:

$$\frac{\partial \mathcal{J}_t}{\partial x_t} = \frac{\partial h_t(x_t, u_t)}{\partial x_t} + \frac{\partial \mathcal{J}_{t+1}}{\partial x_{t+1}} \frac{\partial f_t(x_t, u_t, \eta_t(k))}{\partial x_t} + \frac{\partial \mathcal{J}_t}{\partial u_t} \frac{\partial \hat{\gamma}(x_t, w_t(k))}{\partial x_t} \tag{4.40}$$

$$\frac{\partial \mathcal{J}_t}{\partial u_t} = \frac{\partial h_t(x_t, u_t)}{\partial u_t} + \frac{\partial \mathcal{J}_{t+1}}{\partial x_{t+1}} \frac{\partial f_t(x_t, u_t, \eta_t(k))}{\partial u_t} \tag{4.41}$$

initialized by

$$\frac{\partial \mathcal{J}_T}{\partial x_T} = \frac{\partial h_T(x_T)}{\partial x_T} \tag{4.42}$$

It must be pointed out that in the equations above, the dependency on k has been made explicit only for the stochastic variables $\tilde{\eta}(k)$, which are randomly generated according to their distribution, and the parameters $\tilde{w}(k) = \mathrm{col}(u_0(k), w_1(k), \ldots, w_{T-1}(k))$ because the latter are the ones iteratively changed at each step k. The initial state x_0 is fixed to x_0^*, and so is the desired final state x_T^*. However, the state trajectory $x_t, t = 0, 1, \ldots, T$ changes at each iteration step k, as a consequence of the change in the parameters and consequently of the control laws generating the sequence of controls $u_t, t = 0, 1, \ldots, T-1$, and so does the cost-to-go. Therefore, it would have been more correct to write $x_t(k)$ and $\mathcal{J}_t(k)$: in fact, the dependence on k has been dropped to keep the equations clearer.

Algorithm 1 Find $\tilde{w}^\circ = [u_0^\circ, w_1^\circ, \ldots, w_{T-1}^\circ]$ minimizing (4.30) in Problem 7.

Require: T
Ensure: $\tilde{w}^\circ = u_0^\circ, w_1^\circ, \ldots, w_{T-1}^\circ$
$k = 0$
Initialize $u_0(k), w_1(k), \ldots, w_{T-1}(k)$ randomly or according to some specific technique
repeat
  Generate $\eta_0(k), \ldots, \eta_{T-1}(k)$ according to their probability density
  Forward
  $x_1(k) = f_0[x_0^*, u_0(k), \eta_0(k)]$ according to (4.22)
  for $t = 1 : T-1$ **do**
    Compute $u_t(k) = \hat{\gamma}(x_t(k), w_t(k))$
    Compute $x_{t+1}(k) = f_t(x_t(k), u_t(k), \eta_t(k))$ according to (4.22)
    Compute $\dfrac{\partial h_t}{\partial x_t(k)}, \dfrac{\partial h_t}{\partial u_t(k)}, \dfrac{\partial f_t}{\partial x_t(k)}, \dfrac{\partial f_t}{\partial u_t(k)}$
  end for
  Backward
  Compute $\dfrac{\partial \mathcal{J}}{\partial x_T(k)}$ according to (4.42)
  for $t = T-1 : 0$ **do**
    Compute $\dfrac{\partial \mathcal{J}}{\partial u_t(k)}$ according to (4.41)
    Compute $\dfrac{\partial \mathcal{J}}{\partial x_t(k)}$ according to (4.40)
    for $i = 1 : W_t$ **do**
      Compute $\dfrac{\partial \mathcal{J}}{\partial w_t^i(k)}$ according to (4.36)
      Update weight $w_t^i(k)$ according to (4.34)
    end for
  end for
until Convergence condition is met
return $\tilde{w}^\circ = \tilde{w}(k)$

Remark 10. *The forward-backward technique is naturally decentralized, since each neural control block can "autonomously" be updated, if the proper signal connections between consequent blocks are set up. It is interesting to observe that they provide a time-varying parameterized feedback control law.*

In Algorithm 1, the convergence conditions are not specified, since it is possible to design different stopping rules, and the convergence condition could change according to the specific instance of Problem 6- 7. In general, it is difficult to guarantee convergence of the algorithm to a global minimum. The particular choice of OHL-NN also affects the multi-dimensional surface $\mathcal{J}(\tilde{w})$, which is also stochastic with respect to $\tilde{\eta}$': so, even if the theoretical convergence of the method is assured by a suitable choice of the step size $\alpha(k)$ (e.g. monotonically decreasing - the strictness is not required), in practice it is frequent to find hard to "descent" the cost function because of local minima and "flat" region where the cost is practically constant.

The same procedure described to solve Problem 6 by turning it into Problem 7 can be generalized to the case of team control problems, where multiple cooperating Decision Makers act for the accomplishment of a common goal, if a certain "order" is defined among the team agents. In a T-stage control problem, the order is naturally induced by the timing relationship among the blocks, that comes after the feedback is made explicit. In general team control problems a causality order can be induced by the flux of information from one agent to another, e.g. if decision u_1 taken by DM_1 influences DM_2 in generating u_2 and so on. An example can be found in [Ivaldi et al., 2009a].

4.3.2 Variations in Finite Horizon problems

Hereinafter variations to Problem 6 are discussed, when different conditions occur.

- If the initial state x_0 is not fixed to x_0^*, but can take values from a certain set

$$x_0 \in X_0 \subseteq \mathbb{R}^n \tag{4.43}$$

  then x_0 is a stochastic variable (whose probability density properties are assumed to be known) which must be taken into account in the expectation of the functional cost to be minimized, such that (4.24) becomes

$$\mathcal{F} = \underset{x_0,\eta_0,\ldots,\eta_{T-1}}{E}\{\mathcal{J}\} = \underset{x_0,\eta_0,\ldots,\eta_{T-1}}{E}\left\{\sum_{t=0}^{T-1} h_t(x_t,\gamma_t(x_t)) + h_T(x_T)\right\} \tag{4.44}$$

  Moreover, (4.43) is a new constraint. The solution to the problem is practically the same as previously described. The first control u_0 is not anymore determined straightforward: in Problem 7 $u_0 = \hat{\gamma}(x_0^*, w_0)$ collapses in u_0 simply, i.e. does not require a NN, whereas if x_0 is stochastic, then $u_0 = \hat{\gamma}(x_0, w_0)$ and the NN is necessary. The corresponding forward and backward phases are shown in Figure 4.10(a) and 4.10(b).

- The final state specified in the statement of Problem 6, but as already explained in Remark 9 its dependence was not made explicit in the cost function $\mathcal{J}$. To specify a desired final state x_T^*, two possible approaches are possible: first, the use of a so-called *soft*

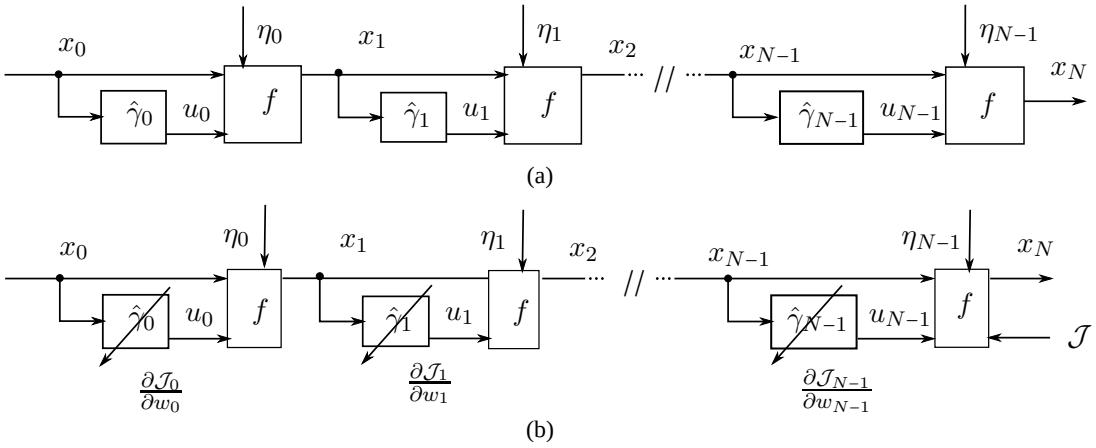


Figure 4.10: The "chain" of states and controls, unfolded in time, when (4.43) holds. Note that $\hat{\gamma}_t$ in the control blocks means $\hat{\gamma}(\cdot, w_t)$. **4.10(a)** and **4.10(b)** show the forward and backward phase respectively.

constraint, i.e. a penalty function to be added to the cost function J, usually convex with a single minimum in x_T^*, which behaves like an attractor for the system state x_T to x_T^*; second, a *hard-constraint*, requiring that $x_T = x_T^*$, which but may not be satisfied if reachability in T stages is weak. As a consequence of the stochastic vector acting on the system, but also on the nature of the cost and state functions themselves, it might not be possible to satisfy the constraint exactly. The interesting, here, is the first approach (also because satisfying hard constraints in our context could require notable efforts). In (4.24) a term accounting for the final state is already considered: $h_T(x_T)$, where, for example, $h_T(x_T, x_T^*) = \|x_T - x_T^*\|^2$.

In general, if the desired final state x_T^* is made explicit, the functional cost (4.24) must be written as:

$$\mathcal{F} = \underset{\eta_0,\ldots,\eta_{T-1}}{E} \left\{ \sum_{t=0}^{T-1} h_t(x_t, \gamma_t(x_t)) + h_T(x_T, x_T^*) \right\} \tag{4.45}$$

and if the desired state is not fixed a priori, but is time-varying (i.e. there exists $x_t^*, t = 0, \ldots, T$) the cost function becomes:

$$\mathcal{F} = \underset{\eta_0,\ldots,\eta_{T-1}}{E} \left\{ \sum_{t=0}^{T-1} h_t(x_t, \gamma_t(x_t), x_t^*) + h_T(x_T, x_T^*) \right\}$$

- With a little complication with respect to the previous case, if x_T^* is not fixed, but again can take values from a finite set

$$x_T^* \in X_T^* \subseteq \mathbb{R}^n \tag{4.46}$$

then (4.24) becomes

$$\mathcal{F} = \underset{x_T^*,\eta_0,\ldots,\eta_{T-1}}{E}\left\{\sum_{t=0}^{T-1} h_t(x_t,\gamma_t(x_t)) + h_T(x_T,x_T^*)\right\} \tag{4.47}$$

and (4.46) is again a new constraint. In this case, the control functions γ_t must also take into account the desired final state, as they could "change" depending on the desired final state. Then it is correct to write:

$$u_t = \gamma(x_t, x_T^*),\ t = 0,\ldots,T-1 \tag{4.48}$$

to make the feed term x_T^* explicit. Of course the "chain" and the equations used for the forward-backward algorithm must be modified to accomplish to the new cost and control function.

One could also define x_T^* as the initial state of a constant system, i.e.

$$\zeta_{t+1} = g_t(\zeta_t) = \zeta_t,\ t = 0,\ldots,T-1$$

where $\zeta_0 = x_T^*$ (with the same probability properties). Then an aggregate system could be designed, where $\xi_t = \mathrm{col}(x_t,\zeta_t) \in \mathbb{R}^{2n}$ would be the new state vector, $F = \mathrm{col}(f_t, g_t)$ the new system of equations, $u_t = \gamma(\xi_t)$ the new control function.

- If both x_0 and x_T^* are not fixed, then

$$\mathcal{F} = \underset{x_0,x_T^*,\eta_0,\ldots,\eta_{T-1}}{E}\left\{\sum_{t=0}^{T-1} h_t(x_t,\gamma_t(x_t),\xi_t) + h_T(x_T,x_T^*)\right\} \tag{4.49}$$

and Problem 6 becomes a finite horizon problem with stochastic boundary value constraints.

- A desired trajectory might also be specified for the system state, thus outlining a *tracking problem* (see Figure 4.11). The goal is to find the set of optimal controls that minimize $\mathcal{F}$ while making the system state x_t track a desired x_t^*. The target state can be treated differently if some information about the target system is known or not. If the target can be modeled, i.e. a differential equation can be written as

$$x_{t+1}^* = f_t^*(x_t^*, u_t^*, \eta_t^*) \tag{4.50}$$

where the simplest equation is

$$x_{t+1}^* = x_t^* \tag{4.51}$$

with a suitable initialization vector with known probability properties, then it is possible to gather both systems (4.22) and (4.50) into an aggregated system.

Then an augmented system vector is designed, $\xi_t = \mathrm{col}(x_t, x_t^*) \in \mathbb{R}^{2n}$, and $F = \mathrm{col}(f_t, f_t^*)$ and $u_t = \gamma(\xi_t)$ will be the new system of equations and control functions.

The latter example raises a practical point. Notwithstanding the polynomial complexity properties of the ERIM, in practical situations one wants to keep the input space to the

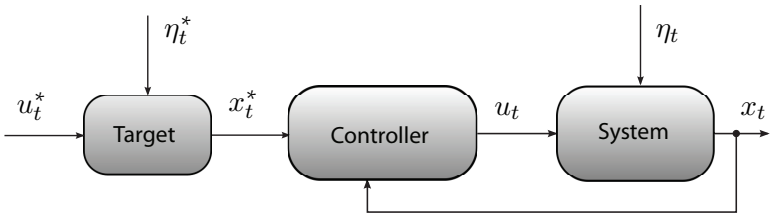


Figure 4.11: A tracking problem.

OHL-NN as small as possible. To this purpose, one can exploit the linearity of the system equation to state the tracking problem as a differential regulation problem, where the system state is $\xi_t = x_t - x_t^*, \xi_t \in \mathbb{R}^n$ and the goal is to bring the new system state to zero. The corresponding control function $u_t = \gamma(\xi_t)$ halves the input space, being $\mu : \mathbb{R}^n \mapsto \mathbb{R}^m$. The impact on the complexity of the problem once the ERIM is applied is notable: if using OHL-NN, the total number of parameters from $W = N[(2n+1)\nu + (\nu+1)m]$ reduces to $W = N[(n+1)\nu + (\nu+1)m]$ ($Nn\nu$ parameters less). A graphical representation of the effect of both solutions on the basic couple of blocks is shown in Figure 4.12.

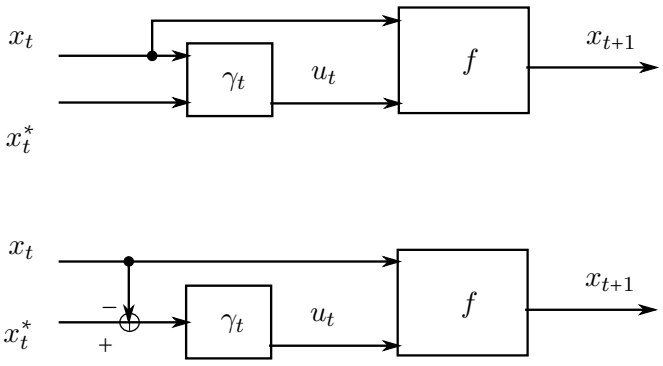


Figure 4.12: The t-th element couple, when the control task is "unfolded" in time and a desired value x_t^* is considered. In the first case, x_t^* is another input to the neural controller. In order to reduce the computational complexity by halving the inputs to the neural controller, the second solution is proposed, where the input is the difference between the desired and the current state. The two solutions are "identical" only if f is linear.

- The stochastic vector η_t acting on the system has been considered to address the most general problem, where noise and disturbances may act on the system. Under suitable assumptions one can neglect the contribution of η_t, or a problem can be stated without

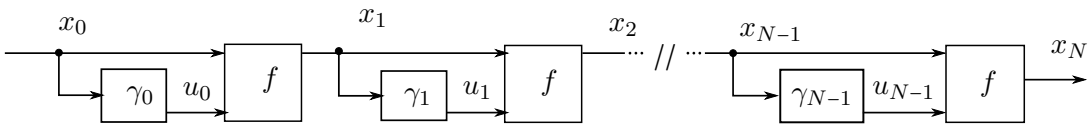


Figure 4.13: Forward phase, when the system is (4.52).

any η_t since the beginning.[8] In that case, the system equation is

$$x_{t+1} = f_t(x_t, u_t),\ t = 0, 1, \ldots, T-1\ ; \tag{4.52}$$

the control function $u_t = \gamma_t(x_t)$ does not change, while the cost function does, according to the statement of the problem. If, for example, x_0 is stochastic, then the cost function to minimize is

$$\mathcal{F} = \underset{x_0}{E}\left\{\sum_{t=0}^{T-1} h_t(x_t, \gamma_t(x_t)) + h_T(x_T)\right\}\ . \tag{4.53}$$

A graphical representation of the chain of system and control blocks, unfolded in time, is shown in Figure 4.13.

4.3.3 A Receding Horizon technique

There are two main limitations to FH control. First, the horizon T must be known and fixed a priori, but often, feedback control systems must run for sufficiently long periods, and the control horizon can be hardly "predicted". This is the case of a robotic motion controller in particular tasks, such as when a the target to reach is time-varying. In these ongoing processes, FH optimal control cannot be adopted: this issue is usually solved by seeking for Infinite Horizon (IH) controls (for example, optimal LQG regulators have both FH and IH formulations).

Second, variable or stochastic final states can be taken into account if the ERIM is used, but in the general statement of a T-stage problem the final state is fixed, and the sequence of optimal controls is open-loop. In other words, in Problem 6 no changes are admitted, and if something in the problem statement changes, for example the target state, the control sequence is no longer optimal. In addition, Receding Horizon (RH) control can be used [Kwon and Han, 2005]. The main advantage of RH control is that it naturally yields closed-loop controls due to the repeated computation and implementation of only the first control of an optimal sequence: this is substantially different from FH control, where the initial and finite state to reach are fixed. The basic concept of RH is as follows.

At the current time t a sequence of T optimal controls, minimizing a T-stage cost function, are derived, basically solving a FH problem like Problem 6. Precisely, the finite fixed horizon taken into account is $[t, t+T]$, and controls are denoted by

$$u^{\mathrm{FH}}_{0|t}, u^{\mathrm{FH}}_{1|t}, \ldots, u^{\mathrm{FH}}_{T-1|t} \tag{4.54}$$

[8] Sometimes it is difficult to assess the effect of exogenous variables on the system, because these terms are not identifiable or modeling is hard. Then either one includes in the problem a generic random vector η_t or neglect the noise.

where the suffix $i|t$ refers to the i-th control in the FH sequence generated at time t. Among the sequence of optimal controls computed on the fixed horizon $[t, t+T]$, only the first one is adopted as the current control law, hence

$$u_t = u_t^{\mathrm{RH}}(x_t) = u_{0|t}^{\mathrm{FH}} \tag{4.55}$$

T defines the so called "Finite Horizon Sliding Window". At stage $t+1$, the same procedure is repeated: the time interval $[t+1, t+1+T]$ is considered, and a sequence of T optimal controls is computed

$$u_{0|t+1}^{\mathrm{FH}}, u_{1|t+1}^{\mathrm{FH}}, \ldots, u_{T-1|t+1}^{\mathrm{FH}}$$

and then only the first is applied to the system. The procedure is repeated up to infinity, for $t = t+2, t+3, \ldots$, where the corresponding time intervals are $[t+2, T+2+T], [t+3, T+3+T], \ldots$. A graphical representation of the concept of RH is shown in Figure 4.15: notice that the term "receding" is indeed introduced since the horizon recedes as time proceeds. We point out that the above receding horizon procedure implicitly defines a time-invariant control policy $u_t : X_t \mapsto U_t$ of the form $u_t^{\mathrm{RH}}(x_t)$, as in Eq. 4.55, which is intrinsically closed-loop, as shown in Figure 4.14.

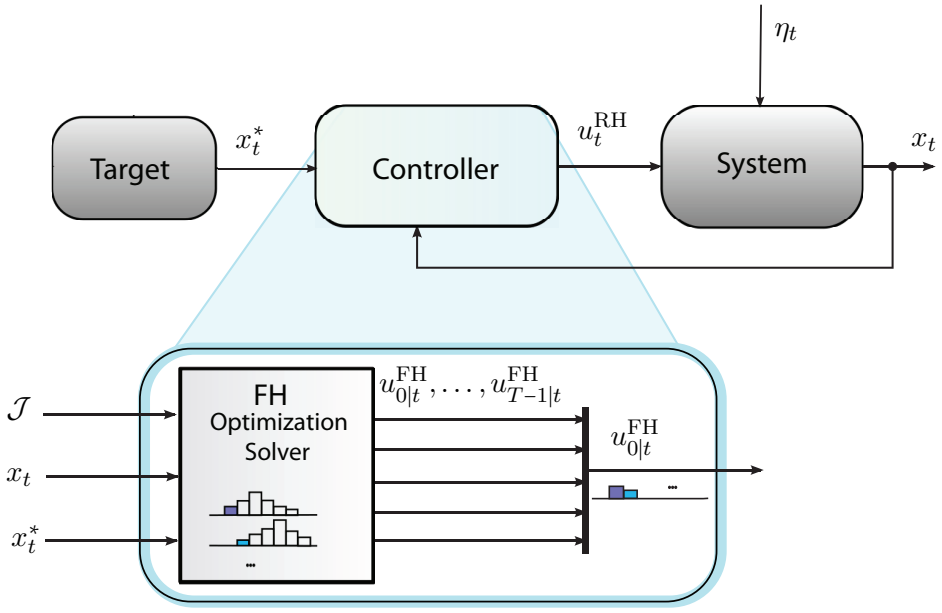


Figure 4.14: Using RH for closed loop control. The mechanism for selecting the first of a sequence of FH optimal controls is reported with more clarity in Figure 4.15.

Stabilizing properties of RH control have been established for different problem statements, for example under LQ assumptions [Kwon and Paearson, 1978, Kwon et al., 1983] and for nonlinear systems [Mayne and Michalska, 1990]. The solution was first provided using the terminal equality constraints, $x_{t+T} = 0$, [Keerthi and Gilbert, 1988, Mayne and Michalska, 1990]; such hard constraint was relaxed in [Michalska and Mayne, 1993], where the regulator was simply required to drive the system to a neighborhood of the origin, where the control switched

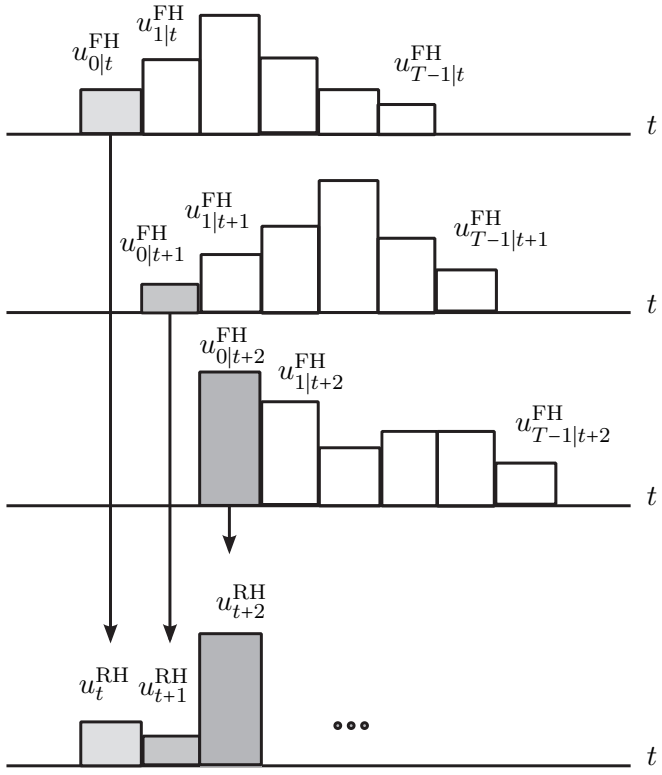


Figure 4.15: The concept of a Receding Horizon controller. At time instant $t = 0$, the target value x_0^* is measured, and a sequence of optimal controls $u^{\mathrm{FH}}_{0|0}, \ldots, u^{\mathrm{FH}}_{T-1|0}$ is computed, consequently with a FH trajectory $x^{\mathrm{FH}}_{0|0}, \ldots, x^{\mathrm{FH}}_{T-1|0}$. Then only the first control is retained, $u^\circ_0 = u^{\mathrm{FH}}_{0|0}$. At time $t = 1, 2, \ldots$, the same procedure is repeated. Three consecutive "instants" are shown, for $T = 4$.

to a linear regulator designed to stabilize the nonlinear system, steering the state to its origin. In [Parisini and Zoppoli, 1995] the attractiveness of the origin was imposed by means of a penalty function in the cost function.

A general RH control problem can be stated in the following way:

Problem 8. *Given the fixed initial state $x_0 = x_0^*$ and a desired state x_t^* to reach, for the system:*

$$x_{t+1} = f_t\left(x_t, u_t, \eta_t\right) \ , \ t = 0, 1, \ldots, \infty$$

where $x_t \in X_t \subseteq \mathbb{R}^n$, η_t is a stochastic variable with known distribution; find the optimal controls $u_t^\circ \in U_t$ where u_t° is the first of a sequence of T optimal controls $u^\circ_{0|t}, \ldots, u^\circ_{T-1|t}$, having the form

$$u^\circ_{0|t} = \gamma^\circ_0(x_t) \in U_t(x_t) \subseteq \mathbb{R}^m$$

which minimize at each time instant t the cost functional

$$\mathcal{F} = \underset{\eta_{0|t},\ldots,\eta_{T-1|t}}{E}\{\mathcal{J}\} = \underset{\eta_{0|t},\ldots,\eta_{T-1|t}}{E}\left\{\sum_{i=0}^{T-1} h_i(x_{i|t}, \gamma_i^\circ(x_{i|t})) + h_T(x_{T|t})\right\} \tag{4.56}$$

where $x^*_{0|t} = x_t$ and $x^*_{T|t} = x^*_{\bar{t}}$.

Notice that the control functions $\gamma_i, i = 0,\ldots,T-1$ are time-invariant, i.e. $\gamma_{i|t} = \gamma_i, \forall t$. As previously pointed out for Problem 6, in (4.56) there is not an explicit reference of the desired state in the cost function $\mathcal{J}$. It is straightforward to notice the relationship between Problem 6 and Problem 8: the RH control u_t° is the first of the optimal controls which are effectively computed once a specific instance of Problem 6, where the initial state x_0^* of the FH problem is the current state x_t in the RH problem ($x^*_{0|t} = x_t$) while the desired state x_T^* is set to the same of the RH problem ($x^*_{T|t} = x^*_{\bar{t}}$).

Incidentally, one may notice that if both initial and desired state to reach $x^*_{\bar{t}}$ are known a priori, exploiting the reachability properties one may think of directly implementing a FH control. However, due to the stochastic noise, the reachability in T stages may not be satisfied or guaranteed, thus a RH control law would be more indicated.

However, the main motivation for using a RH controller is that it can easily deal with a time-varying x_t^* to reach. In particular, in this case a certain principle must be assumed.

Certainty Equivalence

Whenever the controlled system has to track a desired target, or its decisions depend upon the evolution of a certain system, the RH can be easily applied if combined with the so called *Certainty Equivalence* Principle (CE/CEP).

The CE is fundamental when the target is time-varying, x_t^*, and its dynamics and statistical properties are unknown or unpredictable, but it is perfectly measurable at time instant t. In this case, every t the RH controller is designed as if the stochastic quantity x_t^* would remain unchanged in the future, for $t, t+1, \ldots, \infty$. For example, if the target at instant $\bar{t}$ is in the state $x^*_{\bar{t}}$, the controller would assume $x_t^* = x^*_{\bar{t}}, \forall t > \bar{t}$. At time $t+1$, a new measure $x^*_{\bar{t}+1}$ of the target is provided, so the controller would assume $x_t^* = x^*_{\bar{t}+1}, \forall t > \bar{t}+1$. The procedure is repeated iteratively. The corresponding RH control problem is:

Problem 9. *Given the fixed initial state $x_0 = x_0^*$ and a time-varying desired state x_t^* to reach, for the system:*

$$x_{t+1} = f_t(x_t, u_t, \eta_t)\ ,\ t = 0, 1, \ldots, \bar{t} < \infty$$

where $x_t \in X_t \subseteq \mathbb{R}^n$, η_t is a stochastic variable with known distribution; find the optimal controls $u_t^\circ \in U_t \subseteq \mathbb{R}^m$ where u_t° is the first of a sequence of T optimal controls $u^\circ_{0|t}, \ldots, u^\circ_{T-1|t}$, having the form

$$u^\circ_{i|t} = \gamma_i^\circ(x_t) \in U_t(x_t) \subseteq \mathbb{R}^m\ ,\ i = 0,\ldots,T-1$$

which minimize at each time instant t the cost functional

$$\mathcal{F} = \underset{\eta_{0|t},\ldots,\eta_{T-1|t}}{E}\{\mathcal{J}\} = \underset{\eta_{0|t},\ldots,\eta_{T-1|t}}{E}\left\{\sum_{i=0}^{T-1} h_i(x_{i|t}, \gamma_i(x_{i|t})) + h_T(x_{T|t})\right\}$$

where $x^*_{0|t} = x_t$ *and* $x^*_{T|t} = x^*_t$.

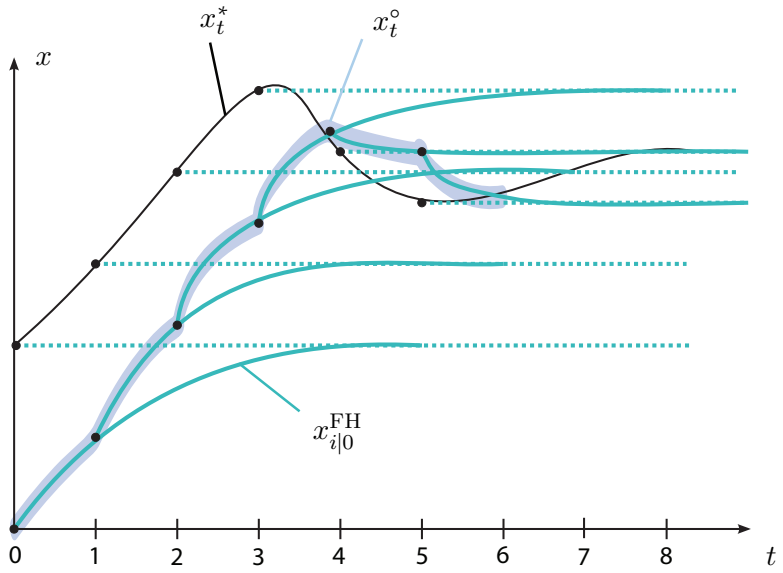


Figure 4.16: An example of tracking when a RH controller is used. A time-varying target x^* is tracked by a system, where a RH controller is used to find the optimal controls u° and the corresponding "optimal" state trajectory x°. At each time instant, the CEP is applied, and a T stages - FH problem is solved. Here, the finite horizon is $T = 6$. At time instant $t = 0$, the target value x^*_0 is measured, a sequence of optimal controls $u^{\mathrm{FH}}_{0|0}, \ldots, u^{\mathrm{FH}}_{5|0}$ is computed, consequently with a FH trajectory $x^{\mathrm{FH}}_{0|0}, \ldots, x^{\mathrm{FH}}_{5|0}$. Then only the first control is retained, $u^\circ_0 = u^{\mathrm{FH}}_{0|0}$, and the system state $x^\circ_1 = x^{\mathrm{FH}}_{1|0}$ is found. The procedure is repeated iteratively for $t = 1, 2, \ldots$. In figure, the FH trajectories $x^{\mathrm{FH}}_{i|t}$, $i = 0, \ldots, 5$, $\forall t$ are shown, as well as the final RH trajectory x°_t.

We remark that in Problem 9 there are no assumptions or models of the target behavior. This means that the RH controls u°_t are "locally" optimal, with respect to the current measure of the target. Combining the RH framework with the CEP, means that at each stage t, the T optimal controls are derived, with the stochastic quantity remaining constant: that is, in the case of tracking, considering the target trajectory to hold, as shown in Figure 4.16.

It is now evident that the main advantage of RH control is that it yields closed-loop controls which can counteract to disturbances and time-varying parameters, due to the repeated computation and implementation of only the first control of a sequence which is "optimal" given the current problem statement: so at each time instant t, the initial and desired state of a new instance of FH problem are set, after measuring the current system and target state. This is substantially different from FH control, where the initial and finite state to reach are fixed *a priori*. In particular, as already observed for Problem 6, variable or stochastic final states can be taken into account if the ERIM is used, but in the general statement of a T-stage problem the final state is fixed, and the sequence of optimal controls is computed in "open-loop" (and is a sequence of numerical values, not functions).

ERIM and RH control

The classical RH technique assumes that a FH optimization routine computes at each control instant t a sequence of optimal controls: basically the control vectors are generated after the solution of a nonlinear programming problem at each time instant t. This procedure can be implemented in real applications only if the process dynamics is sufficiently "slow", i.e. the time between two consequent controls is enough to solve a FH control problem. This assumption is unrealistic in the case of humanoid robotics, as the robot (and sometimes also the target) dynamics is fast and the complexity of the problem generally increases with the number of DOF to control. In order to solve the optimization problem on-line, with the guarantee of satisfying the temporal constraint, a proper hardware and software are required: at least, one should be provided with a real-time processing unit supporting fast and highly precise computations, directly connected to the robot sensing and actuation devices (to avoid delays). Unfortunately, different multi-level control architectures often do not support this control scheme. Both iCub and James, for example, cannot support it, as shown in Figure 4.14, because the PC104 "controlling" the robot is not more than an interface to the cluster (where demanding computations are performed), and cannot bear such computations. This fact motivates the use of the ERIM in this problem: since it allows concentrating in an off-line phase all the computational burden required to approximate the optimal control functions, in the on-line phase the control actions can be promptly generated with a small computational effort [Ivaldi et al., 2008a, Ivaldi et al., 2008b].

Remark 11. *The ERIM can be generalized also for Infinite Horizon (IH) control. In practical applications, the impossibility of applying the ERIM to the IH case is basically due to the necessity of using a chain containing an infinite number of neural networks, which leads to a controversy in the initialization of the backward phase when finding the optimal parameters of the functional approximators [Pianosi and Soncini-Sessa, 2008]. A workaround is to approximate and "truncate" the cost, for example assuming that at stage t the IH for the backward phase "starts" at $\bar{t}$, where $t \ll \bar{t} < t_\infty$. However, the complexity of the solution in this case is considerably high, and justified only for offline precomputation of the optimal control laws, or in applications where the dynamics of the controlled system is particularly "slow", such as in management of water reservoirs [Pianosi, 2008].*

Applying the ERIM to a RH control problem is very easy. Following the same procedure described for turning Problem 6 into Problem 7, applying the ERIM consists in constraining the finite sequence of control functions, generated at each time instant t, to take on a fixed structure with a certain number of parameters to be optimized. More precisely, Problem 9 becomes:

Problem 10. *Given the fixed initial state $x_0 = x_0^\star$ and a time-varying desired state $x_t^\star$ to reach, for the system:*

$$x_{t+1} = f_t\left(x_t, u_t, \eta_t\right) \ , \ t = 0, 1, \ldots, \infty$$

where $x_t \in X_t \subseteq \mathbb{R}^n$, η_t is a stochastic variable with known distribution; find the vectors of optimal parameters $w_t^\circ, \forall t$ corresponding to the optimal controls $u_t^\circ \in U_t \subseteq \mathbb{R}^m$ with the following structure:

$$u_t^\circ = \hat{\gamma}(x_t, w^\circ)$$

where w° is the first of a sequence of T vectors of optimal parameters[9] $w^\circ_{0|t}, \ldots, w^\circ_{T-1|t}$*, corresponding to the optimal controls $u^\circ_{0|t}, \ldots, u^\circ_{T-1|t}$, having the form*

$$u^\circ_{i|t} = \hat{\gamma}(x_{i|t}, w^\circ_{i|t}) \in U_t(x_{i|t}) \subseteq \mathbb{R}^m \ , \ i = 0, \ldots, T-1$$

which minimize at each time instant t the cost functional

$$\mathcal{F} = \underset{\eta_{0|t},\ldots,\eta_{T-1|t}}{E}\{\mathcal{J}\} = \underset{\eta_{0|t},\ldots,\eta_{T-1|t}}{E}\left\{\sum_{i=0}^{T-1} h_i(x_{i|t}, \hat{\gamma}(x_{i|t}, w_{i|t})) + h_T(x_{T|t})\right\}$$

where $x^*_{0|t} = x_t$ and $x^*_{T|t} = x^*_t$.

If $\hat{\gamma}$ are OHL-NNs, the approximating properties of neural RH controller can be found in [Parisini and Zoppoli, 1995]. The same arguments hold, concerning the existence of a sufficient number of neural units and of the corresponding optimal vector of parameters given a desired accuracy in approximation [Hornik et al., 1989], as previously discussed in Section 4.2. In particular, if the first control function $\gamma^\circ(x_t) = \gamma^\circ_{0|t}(x_t)$ is unique, and is a continuous function $\mathcal{C}(X_t, \mathbb{R}^n)$, for every $\epsilon > 0$ there exists an integer ν, a weight vector w°_t and a corresponding neural RH control $\hat{\gamma}^\circ_\nu(x_t, w^\circ_t)$, such that $\|\gamma^\circ(x_t) - \hat{\gamma}^\circ_\nu(x_t, w^\circ_t)\| < \epsilon$, $\forall x_t \in X_t$.

Problem 10 outlines an iterative procedure for finding the optimal parameters. At time instant t, a FH optimal control problem as Problem 7 is stated, for the time interval $[t, t+T]$, where the initial and desired state to reach are set as previously discussed, measuring the current system and target state, and applying the CEP. The solution of the FH problem with ERIM, yields a sequence of optimal parameters

$$w^\circ_{0|t}, \ldots, w^\circ_{T-1|t} \ .$$

Only the first is retained for the RH control, such that

$$w^\circ_t = w^\circ_{0|t} \ , \ u^\circ_t = \hat{\gamma}(x_t, w^\circ_t) \ .$$

At the next time $t+1$ another FH neural problem is stated, for the time interval $[t+1, t+1+T]$, with different initial and target state, a new sequence of controls is computed $w^\circ_{0|t+1}$, …, $w^\circ_{T-1|t+1}$, then the first is used to compute the $t+1$-th control law $u^\circ_{t+1} = \hat{\gamma}(x_{t+1}, w^\circ_{t+1})$. For $t+2, t+3, \ldots$ the procedure is repeated iteratively.

Within this formulation, at each time a new FH problem must be solved. However, we can exploit the time invariance of the problem and the functional approximating properties of the NN, and solve a "general" FH problem only once for all (under suitable assumptions).

Indeed, we can state a FH control problem like Problem 7, where the initial and final state, x^*_0, x^*_T are not fixed, but can take values from a certain set:

$$x^*_0 \in X^*_0 \subseteq \mathbb{R}^n \ , \ x^*_T \in X^*_T \subseteq \mathbb{R}^n$$

[9] *With a slight abuse of notation, we specify the control functions and their parameters with the suffix $i|t$, meaning the i-th control computed by solving the optimization problem at time instant t. In fact, it is not necessary to use t, since the controls are time-invariant, thus $w_{i|t} = w_i, \forall t$, meaning that a unique control function (precisely, a unique vector of parameters) must be computed. However, to enhance the advantage of the complete precomputation of the control laws, we will keep both indexes.*

in that case, the cost functional must include them in the expectation of $\mathcal{J}$, i.e.

$$\mathcal{F} = \underset{x_0^*, x_T^*, \eta_0, \ldots, \eta_{T-1}}{E} \{\mathcal{J}\}$$

and the same for the control laws, whose outputs necessarily depend on the boundary values, in the most general form: $u_t^\circ = \hat{\gamma}(x_t, x_0^*, x_T^*, w^\circ), \forall x_t \in X_t \subseteq \mathbb{R}^n$, where the approximated optimal controls are generated for any possible instance of a FH control problem, given the known stochastic boundary conditions. In that case, Problem 10 becomes:

Problem 11. *Given the fixed initial state $x_0 = x_0^*$ and a time-varying desired state x_t^* to reach, for the system:*

$$x_{t+1} = f_t(x_t, u_t, \eta_t) \ , \ t = 0, 1, \ldots, \infty$$

where $x_t \in X_t \subseteq \mathbb{R}^n$, η_t is a stochastic variable with known distribution; find the vector of optimal parameters w° corresponding to the optimal controls $u_t^\circ \in U_t \subseteq \mathbb{R}^m$ with the following structure:

$$u_t^\circ = \hat{\gamma}(x_t, x_t^*, w^\circ)$$

where w° is the first of a sequence of T vectors of optimal parameters $w_0, \ldots, w_{T-1}$, corresponding to the optimal control functions having the form

$$u_i = \gamma_i^\circ(x_i, x_0^*, x_T^*, w_i) \in U_i(x_i) \subseteq \mathbb{R}^m \ , \ i = 0, \ldots, T-1$$

where $x_0^* = x_t, x_T^* = x_t^*$, which minimize at each time instant t the cost functional

$$\mathcal{F} = \underset{x_0^*, x_T^*, \eta_0, \ldots, \eta_{T-1}}{E} \{\mathcal{J}\} = \underset{x_0^*, x_T^*, \eta_0, \ldots, \eta_{T-1}}{E} \left\{ \sum_{i=0}^{T-1} h_i(x_i, \hat{\gamma}(x_i, w_i)) + h_T(x_T) \right\}$$

where $x_0^* \in X_0^* \subseteq \mathbb{R}^n$ and $x_T^* \in X_T^* \subseteq \mathbb{R}^n$.

The fundamental difference between Problem 10 and Problem 11 is that while the first requires the execution of the optimization procedure at each time instant t (i.e. the solution of a specific FH problem), the latter requires the solution of a more generalized (yet more complex) FH control problem only once. The main advantage of this approach, is that it is possible to pre-compute explicitly, upon a desired approximation accuracy, the optimal RH control law, completely in an off-line phase. The information of the optimal RH control law lies on the vector of optimal parameters, w°, that is on a finite number of real values. Thus, once the control law is computed, it can be easily implemented and embedded in real-time control loops: at any time instant t, given the target and system measurements x_t^*, x_t, the computation of the associated optimal control u° is almost instantaneous, being a simple forward of a NN (a finite number of sums and products). A scheme illustrating the benefits of this solution when applied to closed loop control is shown in Figure 4.17.

The pseudo-code for the solution of Problem 11 is shown in Algorithm 2.

Algorithm 2 Receding-Horizon procedure combined with ERIM, for the solution of Problem 11.

"Online" RH

Ensure: $u_t^\circ,\ \forall t$

Pre-compute w° = Offline FH(T, X_0^*, X_T^*)
loop
  Measure x_t^*, x_t
  Apply $u_t^\circ = \hat{\gamma}(x_t, x_t^*, w^\circ)$
end loop

"Offline" FH

Require: T,X_0^*,X_T^*

Ensure: $\tilde{w}^\circ = w_0^\circ, \ldots, w_{T-1}^\circ$

$k = 0$
Initialize $\tilde{w}(k)$ randomly or according to some specific technique
repeat
  Generate $\eta_0(k), \ldots, \eta_{T-1}(k)$ according to their probability density
  Generate $x_0^*(k), x_T^*(k)$ according to their distribution over X_0^*,X_T^*
  Initialize $x_0(k) = x_0^*(k)$
  Forward
  for $t = 0 : T-1$ **do**
    Compute $u_t(k) = \hat{\gamma}(x_t(k), x_T^*, w_t(k))$
    Compute $x_{t+1}(k) = f_t(x_t(k), u_t(k), \eta_t(k))$
    Compute $\dfrac{\partial h_t}{\partial x_t(k)}, \dfrac{\partial h_t}{\partial u_t(k)}, \dfrac{\partial f_t}{\partial x_t(k)}, \dfrac{\partial f_t}{\partial u_t(k)}$
  end for
  Backward
  Compute $\dfrac{\partial \mathcal{J}}{\partial x_T(k)}$
  for $t = T-1 : 0$ **do**
    Compute $\dfrac{\partial \mathcal{J}}{\partial u_t(k)}$
    Compute $\dfrac{\partial \mathcal{J}}{\partial x_t(k)}$
    for $i = 0 : W_t$ **do**
      Compute $\dfrac{\partial \mathcal{J}}{\partial w_t^i(k)}$
      Update weight $w_t^i(k)$
    end for
  end for
until Convergence condition is met
return $\tilde{w}^\circ = \tilde{w}(k)$

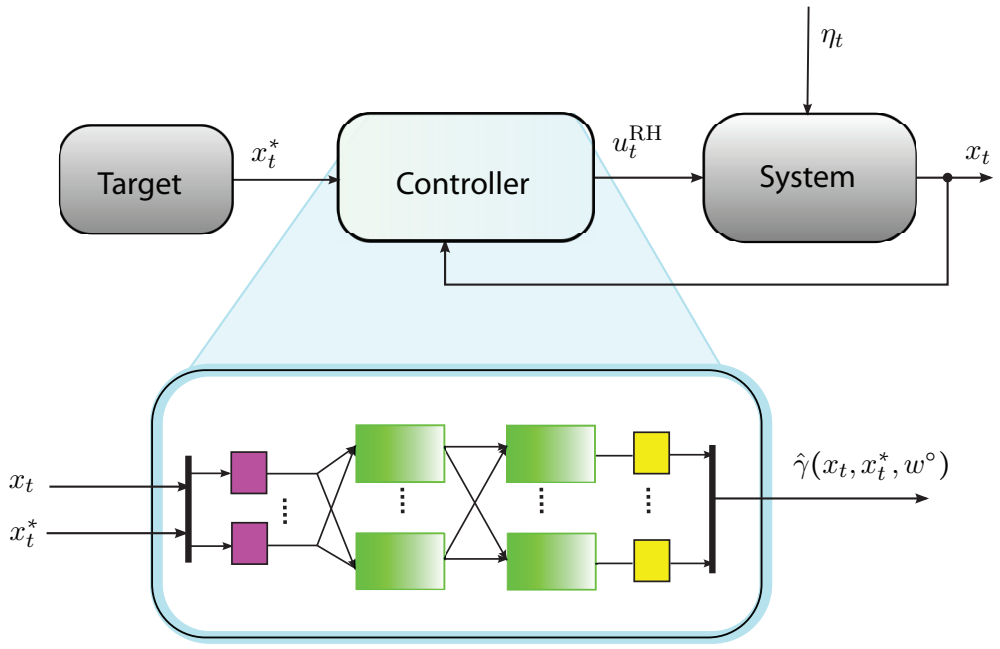


Figure 4.17: Using a neural approximation of a RH control law, as described by Problem 11. In contrast to Figure 4.14, in the online phase the optimization procedure required to find the controls is no longer needed, since the NN can be pre-trained offline. Thus, $u_t^\circ = u_t^{\mathrm{RH}} = \hat{\gamma}(x_t, x_t^*, w^\circ)$. The OHL-NN represented in the box is the same as in Figure 4.4.

4.4 Neural Finite and Receding Horizon regulators for reaching and tracking

In the following we will state two fundamental motion control problems, which must be implemented in a robot in order to provide reaching and tracking skills.

Let us consider the following robotic scenarios:

1. A humanoid robot is sitting in front of a desk, and looking at some objects: the task is to choose some objects, pick up them and put them in a box. When it recognizes an interesting item (for example, because its attentive system labels this item as interesting), it moves quickly its hand towards the object, thus performing a point-to-point, goal-directed movement; grasps the object, and with another point-to-point fast movement put the object inside the box, finally releasing the hand.
2. A humanoid robot is playing in a interaction scenario with a child: the baby wants the robot to catch his object, so he moves the hand almost randomly in the space, to catch the robot's attention. The robot, once recognized the target, moves its hand towards the moving target, until it catches the target.

The first scenario addresses a *reaching* problem: the initial position of the end-effector and the desired target position of the object to reach are known. If the target is in the reachable

space of the robot's workspace, and a certain movement duration can be estimated, it is possible to state the reaching problem as a FH optimal control problem.

The second scenario addresses instead a *tracking* problem: the initial position of the end-effector is known, however the target position is known only instantly (at each measurement), since it changes rather unpredictably (although smoothly). Moreover, it is hard to cast a prediction on the movement duration: in this case, it is possible to state the tracking problem as an IH or RH optimal control problem.

In the previous Sections, it was shown how the ERIM can approximate numerically the optimal solution to such problems. Another advantage of this approach, is that once the structure of the control functions is fixed (i.e. OHL-NN with ν "neurons"), it is possible to switch from one solution to the other by simply loading a new set of parameters, which is merely a vector of real numbers. Thus, one can build a "neutral" NN control function on the main board connected to the robot (i.e. the PC104 for iCub), and then simply load the right set of parameters for any desired control task. Furthermore, not only it is possible to switch from FH to RH controllers, but also among different FH controllers, where the cost function to minimize $\mathcal{J}$ has been chosen to implement different computational motor control models, as the ones described in Chapter 3. That is one can switch from reaching with the MJM (Section 3.2.1), to reaching with the MTCM (Section 3.2.2), to tracking minimizing a simple quadratic cost function. Switching can be done online, because the optimal control functions are pre-computed in an offline phase, where the NNs are "trained" with iterative procedures like the ones in Algorithm 1 and Algorithm 2.

Let us now state the problems more specifically.

Let us denote by x^r the Cartesian coordinates of the end-effector of the robot, with respect to a fixed reference frame, by q, τ the corresponding vectors of the manipulator joints coordinates and torques, respectively. Then the forward kinematics is $x^r = f_{\text{arm}}(q) : \mathbb{R}^{n_q} \mapsto \mathbb{R}^{n_x}$, and can be easily retrieved by means of the Denavit-Hartenberg description of the robot kinematics [Sciavicco and Siciliano, 2005]. The forward dynamics can be instead described by the rigid-body dynamics model. We remark that:

- $q \in \mathbb{R}^{n_q}$ is the vector of joints coordinates, and can be different (conceptually) from the vector of controlled joints;
- $x^r \in \mathbb{R}^{n_x}$ is usually a vector with $\dim(x^r) = 3, 6, 7$ depending on what is controlled among the Cartesian coordinates: position only, or position and orientation

The generic Cartesian coordinates of the target are denoted by $x^g \in \mathbb{R}^{n_x}$. At time instant t, the robot and target state vectors are x^r_t, x^g_t.

It is assumed that the following compact model can be used to describe the evolution of the end-effector with respect to the target, if the control u_t acts on the robot:

$$\xi_{t+1} = f\left(\xi_t, u_t\right) \ , \ t = 0, 1, \ldots \tag{4.57}$$

where at time instant t, ξ_t is the big state vector, taking values from a finite set $\Xi \subseteq \mathbb{R}^n$, and u_t is the control vector, constrained to take values from a finite set $U \subseteq \mathbb{R}^m$.

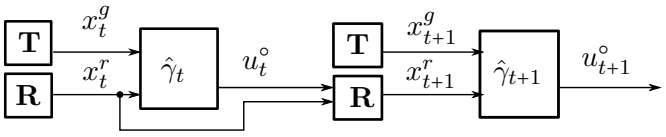


Figure 4.18: A conceptual scheme, representing the interaction among robot, target, and neural controls, unfolded in time. $\mathbf{T}$ accounts for the "target", while $\mathbf{R}$ for the "robot". $\hat{\gamma}_t$ can be either a FH control law, $\hat{\gamma}(\cdot, w_t^\circ)$, where $w_t^\circ = w_t^{\mathrm{FH}}$, or a RH control law $\hat{\gamma}(\cdot, w^\circ)$, where $w^\circ = w^{\mathrm{RH}}$.

Remark 12. *Here, we keep the problems statement as generic as possible: indeed, we do not specify neither the nature of u_t nor the one of ξ_t. For example, the control vector u_t can be a velocity joint command, or a joint torque command; ξ_t can be generically a vector describing the difference between the current end-effector coordinates and the target ones, $\xi_t \triangleq [x_t^g - x_t^r]$, or a more detailed vector containing position, velocities, acceleration errors, etc. (e.g. $\xi_t \triangleq [x_t^g - x_t^r, \dot{x}_t^g - \dot{x}_t^r]$). Consequently, we do not specify the structure of f, that could be a pure kinematics or dynamics model, or a combination of both.*

Assumption 11. *The robot kinematics and dynamics model is perfectly known.*

Thanks to Assumption 11, it is possible to write Eq. 4.57, where f is perfectly known. Moreover, thanks to the time invariance of the system f (and of the cost function), $t = 0$ can be considered as a generic time instant. The goal of the reaching control problem is to find, at time instant t, a sequence of N optimal controls $u_0^\circ, \ldots, u_{N-1}^\circ$ that minimize a suitable cost function $\mathcal{J}$, which is chosen so as to characterize the trajectory of the system state ξ_i, which is steered from an initial state ξ_0 towards a desired ξ^*.

$\mathcal{J}$ takes generally the following form

$$\mathcal{J} = \sum_{i=0}^{N-1} h_i(\xi_i, \xi^*, u_i) + h_N(\xi_N) \tag{4.58}$$

where h_N, h_i are partial cost terms. A typical cost structure in automatic controls is:

$$\mathcal{J} = \sum_{i=0}^{N-1} P_i \left\| u_i \right\|^2 + V_i \left\| \xi_i \right\|^2 + V_N \left\| \xi_N \right\|^2 \tag{4.59}$$

where P_i, V_i, V_N are suitable positive definite weight matrices. If $\xi_i \triangleq x^g(t) - x^r(t)$ then (4.59) penalizes generically the error between the Cartesian coordinates of end-effector and of the target to reach, at the end and during the whole trajectory (see V_N and V_i), along with a penalty on the amount of "energy" spent for the controls. Some examples of cost functions for computational motor control models have been presented in Chapter 3.

Then, the neural reaching problem can be stated as follows, as a FH optimization problem:

Problem 12 (Neural Reaching)**.** *Given the system of Eq. 4.57, find the vectors of optimal parameters $w_0^\circ, \ldots, w_{N-1}^\circ$ that minimize the cost function*

$$\mathcal{F} = \underset{\xi_0 \in \Xi, \xi^* \in \Xi^*}{E} \left\{ \sum_{i=0}^{N-1} h_i(\xi_i, \xi^*, \hat{\gamma}(\xi_i, \xi^*, w_i)) + h_N(\xi_N, \xi^*) \right\}$$

under the constraints given by the state equation, where controls take on the structure $u_i = \hat{\gamma}(\xi_i, \xi^*, w_i) \in U \subseteq \mathbb{R}^m$.

It is remarked that the problem is generalized for any possible initial state $\xi_0 \in \Xi$ and desired state $\xi^* \in \Xi^*$ (i.e. for every possible robot and target's coordinates), in virtue of the expectation operator E. The solution of Problem 12 follows the aforementioned iterative procedure for the solution of T-stage FH problems. It is important to point out that once the problem is solved, the set of vectors $w_0^\circ, \ldots, w_{N-1}^\circ$ explicitly carry the information about the optimal control laws.

Incidentally, the solution of Problem 12 allow solving without effort the corresponding RH problem. Indeed, by making the assumption that at time instant t the target state $\xi_t^* = 0$ will hold for N stages, thus applying a CEP, it is possible to exploit the first FH control law and apply it as a RH control law, as previously discussed. This method particularly fits to the tracking unknown/unpredictable targets: the robot assumes the target to retain the current position whenever a new measurement is available, and moves toward it as if the planned trajectory would assume it fixed for N stages. A change in the target's position does not affect the tracking policy, since at each time instant the target's coordinates are measured and the proper control corresponding to a newer trajectory is performed.

Then, the neural tracking problem can be stated as follows, as a RH optimization problem:

Problem 13 (Neural Tracking)**.** *Given the system of Eq. 4.57, find the vector of optimal parameters* w°*, being the first among the set* $w_0^\circ, \ldots, w_{N-1}^\circ$ *that minimize the cost function*

$$\mathcal{F} = \underset{\xi_0 \in \Xi, \xi^* \in \Xi^*}{E} \left\{ \sum_{i=0}^{N-1} h_i(\xi_i, \xi^*, \hat{\gamma}(\xi_i, \xi^*, w_i)) + h_N(\xi_N, \xi^*) \right\}$$

under the constraints given by the state equation, where controls take on the structure $u_i = \hat{\gamma}(\xi_i, \xi^*, w_i) \in U \subseteq \mathbb{R}^m$.

Once $w^\circ = w_0^\circ$ is found, the generic tracking control law is:

$$u_t^\circ = \hat{\gamma}(\xi_t, \xi_t^*, w^\circ)$$

that is a generic, time-invariant closed loop control function.

Since the RH problem exploits the solution of the corresponding FH problem, only the latter must be solved. The solution of the aforementioned problems is based on the forward-backward algorithm: Figure 4.18 shows a simple scheme with the "chain" of NN, target and system blocks, unfolded in time, which are used for computing the optimal parameters (see also [Ivaldi et al., 2010]).

4.5 Numerical results

In the following, we will discuss some numerical results, where two peculiar systems have been tested in a variety of problems settings, both in reaching and tracking mode: a 2DOF robotic manipulator and a 3DOF nonholonomic mobile robot. The aim was to analyze the performances of the proposed methods on relatively simple nonlinear systems with respect to

the growing complexity of the problem. In particular, the idea to use a simple nonlinear system was inspired by the desire to apply the method directly at joint level control, thus combining the optimal planning with the low-level task to joint space conversion modules.

Numerical simulations show interesting results: however, even if the proposed methods show some evident advantages, there are many practical issues which have to be taken into account, and which we believe could set limitations on the application domain of the methods. First, the method can manage only theoretically a significant number of parameters without incurring in the COD, i.e. the exponential growth of the number of parameters with the complexity of the problem, because their growth is actually polynomial [Zoppoli et al., 2001]. But the number of parameters is still very high, due to the fact that the number of controls N must be large enough to provide a plausible sequence of controls, and similarly each NN can provide sufficient approximation capability for wide class of functions only with large number of parameters. The second issue is the training complexity, which is a combination of the effects of the number of parameters, the back-propagation algorithm, and the stochastic gradient. The third issue is related to the stochastic formulation and the consequent amount of training data, and the time required to train the chain. Finally, the initialization of the OHL-NN can play a role, as a randomized initialization is correct from a theoretical point of view, and is the only plausible way to initialize the neural network if there is no a priori hypotheses on the structure of the control laws. If any information on their "shape" is available, it could be useful to initialize the networks differently, for example with a classical least-squares algorithm. A more detailed analysis of these defects is reported in Section 4.5.4.

4.5.1 A two DOF manipulator in a planar space

In the following preliminary numerical results concerning the control of an anthropomorphic arm are presented. For the sake of simplicity, a two DOF arm is used: the main reason is the possibility to easily compute both Cartesian and joint coordinates easily, as the forward and inverse kinematics are known, and as $n = m = 2$ the manipulator is not redundant. The forward kinematics of the planar arm is:

$$\begin{aligned} x^r &= l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ y^r &= l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \ . \end{aligned}$$

where q_1, q_2 are the joints angles, while x^r, y^r the Cartesian coordinates of the end-effector in the $X - Y$ planar space. A representation is shown in Figure 4.19.

Point-to-point movement with the MJM

With the following numerical example, we want to verify the capability of a chain of neural networks to approximate a desired control function minimizing a certain cost, upon a desired accuracy. Here, the "neural controllers" must drive the end-effector from an initial to a desired position in the robot's workspace, with initial and final null velocities.

The "neural controllers" must drive the end-effector of a manipulator from any initial position $\xi_0 = [x_0, y_0] \in X \subseteq \mathbb{R}^2$ to a final desired $\xi^* = [x^*, y^*] \in X \subseteq \mathbb{R}^2$. The initial and final

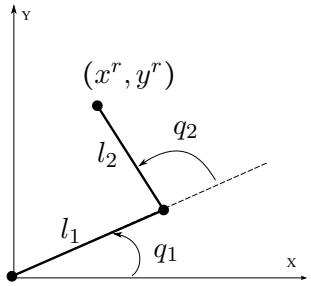


Figure 4.19: A simple model of a two DOF planar arm.

velocity must be equal to zero. The MJM has been chosen as a computational model describing the motion trajectory, and the following *minimum jerk* is the cost criterion:

$$\mathcal{J} = \int_0^T \left[\left(\frac{d^3 x^r}{dt^3} \right)^2 + \left(\frac{d^3 y^r}{dt^3} \right)^2 \right] dt \tag{4.60}$$

As shown in Figure 4.20, the MJM produces bell-shaped, smooth trajectories in the joint space, while the planar trajectory is almost straight. The known analytical solution has been compared with the "neural" trajectory: with $N = 60$, OHL-NN with sigmoidal activation function and linear output layer, and $\nu = 40$, the accuracy in the approximation was very high.

Tracking a point moving unpredictably

An instance of Problem 13 has been stated with the following cost function:

$$\mathcal{J} = \sum_{i=0}^{N-1} c(u_i) + \xi_{i+1}^T V_{i+1} \xi_{i+1} \tag{4.61}$$

where $\xi_i = [x_i^r - x_i^g, y_i^r - y_i^g] \in X \subseteq \mathbb{R}^2$, $\xi^\star = 0$, $N = 30, \nu = 40$, OHL-NNs with sigmoidal activation function. f was basically a double integrator. Approximately 10^9 samples were used for the off-line training of the NNs, considering any possible position of both target and end-effector in the reachable space. The criterion for the task accomplishment in (4.61) is a tradeoff between the minimization of the energy consumption and the "best" end-effector proximity to the target during and at the end of the maneuver (it could not be able to reach it perfectly, as a consequence of the unpredictable behavior of the target or the robot's intrinsic physical limits). Weight matrices V_i were chosen such as to obtain a reasonable compromise between the attractiveness of the target and the energy consumption, whereas $c(u_{j,t}), j = x, y$ is a nonlinear but convex function [Ivaldi et al., 2008c]:

$$c(u_{j,t}) = \alpha \left[\frac{1}{\beta} \ln(2 + e^{\beta u_{j,t}} + e^{-\beta u_{j,t}}) - \frac{1}{\beta} \ln(4) \right], j = x, y \tag{4.62}$$

which, for large values of β approximates the ideal but non differentiable cost $\alpha |u_{j,t}|$, as shown in Figure 4.22. Of course, $u_{j,t}$ indicates the j-th component of the control at time instant t, i.e.

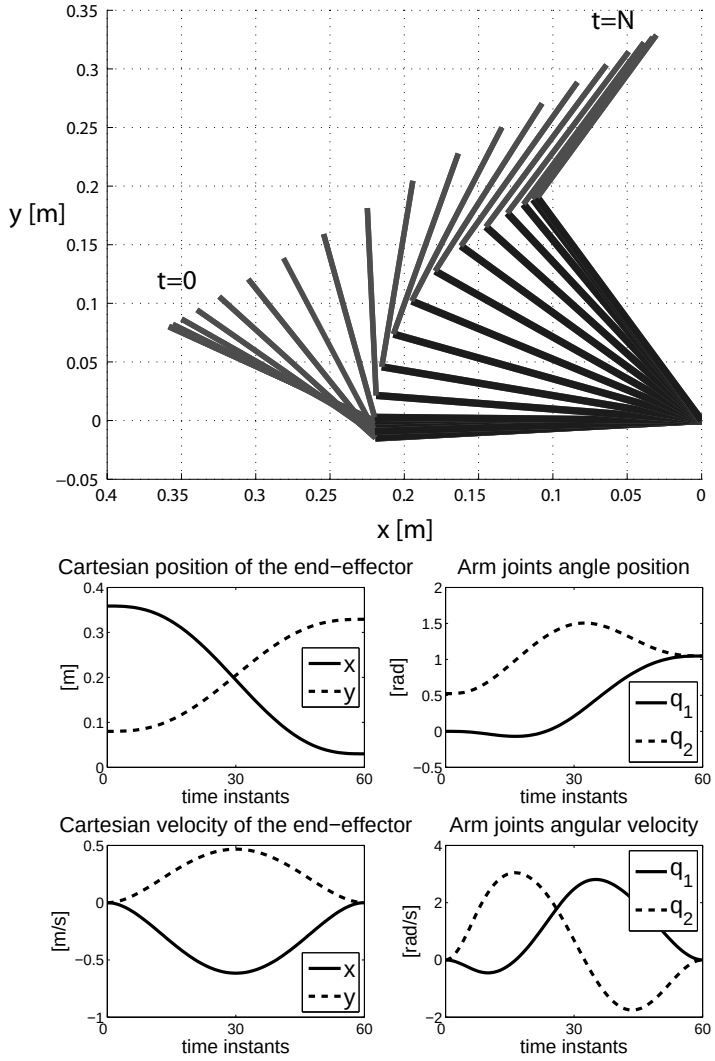


Figure 4.20: A minimum jerk planar movement of a two DOF arm: Cartesian and joints position and velocity are shown, as well as samples of the planar trajectory. The neural approximation and the analytical solution [Flash and Hogan, 1985] are almost coincident (MSE $\cong 10^{-7}$).

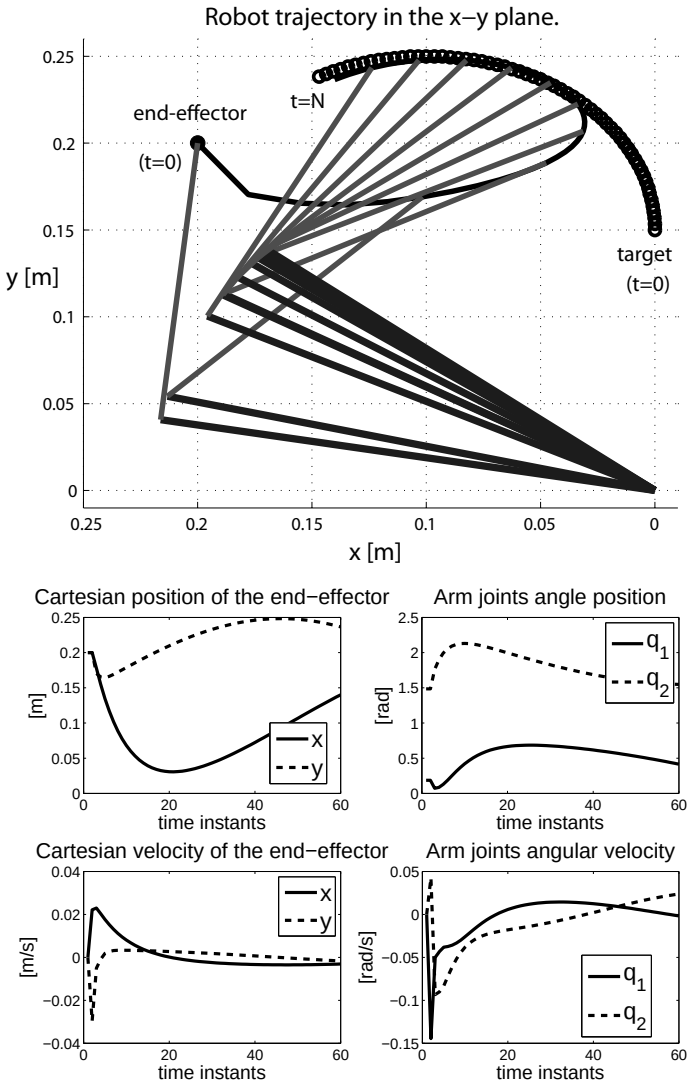


Figure 4.21: The end-effector of a two DOF arm, tracking a point moving in an unpredictable way on a planar space.

u_t. Moreover, $V_i = \mathrm{diag}(1.0, 80.0, 5.0, 10.0),\ i = 0, \ldots, N-1$, $V_N = 40I$ (being I the identity or unit matrix), while $\alpha = 0.01$ and $\beta = 50$. In Figure 4.21 the end-effector, starting still in a certain position, tracks a target moving unpredictably in the space. Cartesian coordinates as well as joints coordinates (for the two DOF) during the movement are shown. It is worth noting that although the target dynamics is quite fast, and the end-effector was initially "far" from the target, the arm follows the moving point with reasonable performances.

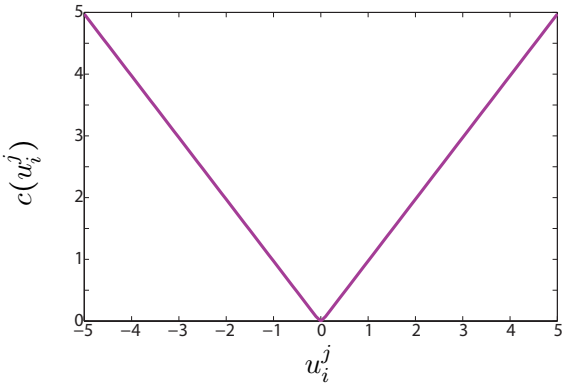


Figure 4.22: The cost function (4.62), with $K = 0.01$ and $\beta = 50$.

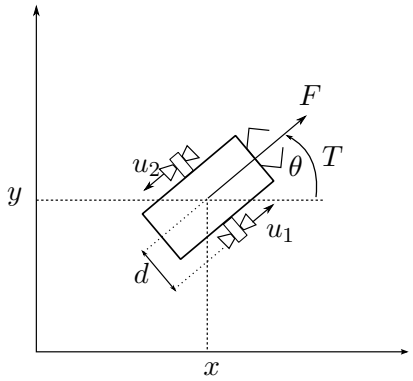


Figure 4.23: The mobile robot described by Eq. 4.63.

4.5.2 A three DOF nonholonomic mobile robot in a planar space

A nonholonomic mobile robot moves on a planar space. The robot position with respect to the coordinate system is described by the Cartesian coordinates in the space x,y and by the angle θ of its axis of symmetry with respect to the x axis (see Figure 4.5.1). On the robot sides two couples of thrusters, aligned with the axis of symmetry, are mounted at constant distance d, and can be modulated so as to obtain the desired intensity of the force F and the desired torque

C by which to control the robot motion. The thrusts identify the two controls u_1 and u_2:

$$\begin{aligned} F &= (u_1 + u_2)e = m\ddot{x} \\ C &= (u_1 - u_2)d = J\dot{\theta} \end{aligned} \tag{4.63}$$

Mass m and moment of inertia J are assumed to be constant during the maneuver. The system state ξ is then composed of six variables, $\xi = \mathrm{col}(x, \dot{x}, y, \dot{y}, \theta, \dot{\theta})$, and the nonlinear differential dynamic of the robot is:

$$\begin{array}{ccc} \dot{\xi}_1 = \xi_2, & \dot{\xi}_3 = x_4, & \dot{\xi}_5 = \xi_6 \ , \\ \dot{\xi}_2 = \frac{1}{m}(u_1 + u_2)\cos\xi_5 \ , & \dot{\xi}_4 = \frac{1}{m}(u_1 + u_2)\sin\xi_5 \ , & \dot{\xi}_6 = \frac{d}{J}(u_1 - u_2) \end{array}$$

subject to constraints to the maximum allowed thrust values: $|u_i| \leq U,\ i = 1, 2$.

Let us now denote $\xi_t \triangleq \xi(t\Delta t),\ u_t \triangleq u(t\Delta t)$, where Δt is the sampling period, obtained by dividing the duration T of the maneuver into N discrete stages. As done before, the j-th component of vectors ξ_t, u_t is denoted by $\xi_{j,t}, u_{j,t}$. Then a discrete-time version of the system can be obtained by using a first-order Euler's approximation:

$$\begin{aligned} \xi_{1,t+1} &= \xi_{1,t} + \Delta t\ \xi_{2,t} \\ \xi_{2,t+1} &= \xi_{2,t} + \Delta t \frac{1}{m}(u_{1,t} + u_{2,t})\cos(\xi_{5,t}) \\ \xi_{3,t+1} &= \xi_{3,t} + \Delta t\ \xi_{4,t} \\ \xi_{4,t+1} &= \xi_{4,t} + \Delta t \frac{1}{m}(u_{1,t} + u_{2,t})\sin(\xi_{5,t}) \\ \xi_{5,t+1} &= \xi_{5,t} + \Delta t\ \xi_{6,t} \\ \xi_{6,t+1} &= \xi_{6,t} + \Delta t \frac{d}{J}(u_{1,t} - u_{2,t}) \end{aligned}$$

which can be written in a more general and compact form as

$$\xi_{t+1} = f(\xi_t, u_t)\ ,\ t = 0, 1, \ldots, N-1$$

where ξ_t may take values from a finite set $X_t \subseteq \mathbb{R}^n$, while u_t is constrained to take values from a finite set $U_t(\xi_t) \subseteq \mathbb{R}^m$ (related to the physical limits of the thrusters). The desired robot configuration at instant t is denoted by ξ_t^*: a desired trajectory to track can be described by the sequence of vectors $\xi_0^*, \ldots, \xi_N^*$.

It is important to remark that the linearized models of such nonholonomic systems are not controllable. So far a number of control strategies have been proposed. In [Aicardi et al., 1995] a Lyapunov stabilizing control law was proposed, while in [Gu and Hu, 2005] a switching controller was presented, based on a Lyapunov-like function, gathering stability over a receding horizon. In [Parisini and Zoppoli, 1995] a receding horizon control strategy was introduced, which was based on the use of neural approximators: in particular, the generalized control law was found as the best interpolation of a set of optimal deterministic controls, within a specific training set.

A reaching problem over a Finite Horizon

We address the problem of regulating the planar robot in Figure 4.5.1, that is to steer the robot towards a target position ξ^* from any initial one, by minimizing a certain cost function, which generally takes into account the fuel consumption and the maneuver accuracy. The criterion for the task accomplishment is a tradeoff between the minimization of the fuel consumption, taken into account by a general nonlinear convex function $c(u_{i,t}), (i = 1, 2)$, and the "best" robot proximity to the target at the end of the maneuver (it could not be able to reach it perfectly, as a consequence of the robot's intrinsic physical limits):

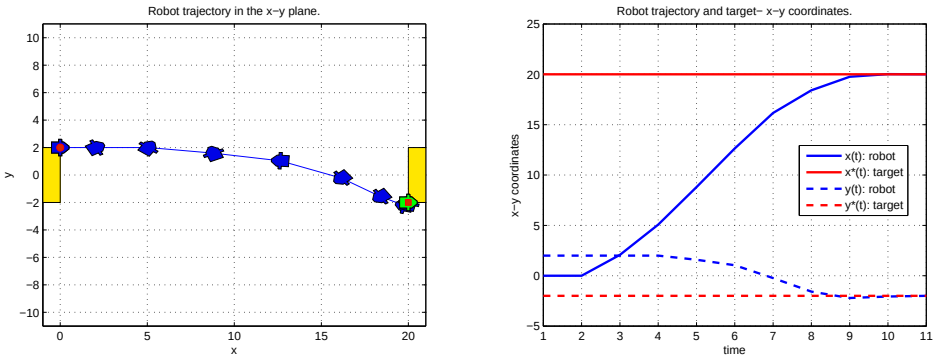
$$\mathcal{J} = \sum_{t=0}^{N-1} c(u_{1,t}) + c(u_{2,t}) + \|\xi^*_{t+1} - \xi_{t+1}\|^2_{V_{t+1}} \tag{4.64}$$

As previously done for the manipulator example, the weight matrices V_t are chosen so as to obtain a reasonable compromise between the attractiveness of the target and the fuel consumption. More specifically, we used $N = 10$ control stages in $T = 10$ (so $\Delta t = 1.0$); moreover $V_t = \mathrm{diag}[1, 0.1, 40.0, 0.1, 40.0, 0.1]$, for $t = 1, \ldots, T - 1$, and $V_N = \mathrm{diag}(40.0)$. $c(u_{i,t})$ has been indeed designed to be realistically proportional to the thrust, and approximate at the same time the ideal but non differentiable cost $K|u_{i,t}|$: again, we used Eq. 4.62, with $K = 0.01$ and $\beta = 50$. The control functions were implemented by OHL-NN, using 12 inputs variables (6 from ξ_t and 6 from ξ^*_t) and $\nu = 80$ neural units in the hidden layer.

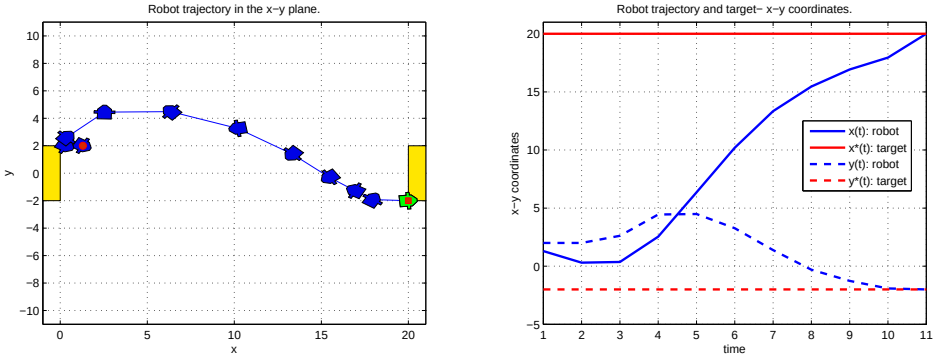
A tracking problem over a Receding Horizon

There are many applications of mobile robots following a target: for example, when they must follow a human as their leader. To this purpose, a RH regulation problem can be stated. As done for the 2 DOF, an instance of Problem 13 has been stated. Again, the general RH control law can be easily found after the solution of the equivalent FH regulation problem, which is stated for any possible robot and target configuration. In detail, a set of L admissible configurations of both robot and target, x^r and x^g were generated, then fed to the iterative algorithm for the computation of the optimal parameters of the approximating NNs. Note that formulating the problem as a regulation one, the system state (i.e. the difference between robot and target) is steered to the origin of the state space: this brings the advantage to halve the number of inputs to the NN, as already discussed in Figure 4.12. In the following numerical examples, the cost is the same as in (4.64), $N = 30$, $\Delta t = 1s$ and the OHL-NNs have $\nu = 80$ neurons.

Figure 4.25 shows some FH trajectories, after $\approx 10^6$ training steps, which have been used to provide the solution to the RH problem. It must be remarked that the entire workspace was sampled to get a suitable set of configurations for both robot and target, but in the regulation problem the goal is to bring the difference between the robot and the target configuration to zero, that is why the trajectories are directed towards the origin of the Cartesian space. The starting points are found after computing the difference between the robot and the target's configuration (both position and velocities). To keep the plots as clear as possible, each trajectory is shown in the planar space, and corresponding positions and velocities are shown. It is evident from Figure 4.25 that the training phase was not sufficient to approximate the global optimal solution: some trajectories in fact are not perfect as expected in the proximity of the origin



(a)



(b)

Figure 4.24: Optimal trajectories from $\xi_0 \in \Xi_0$ to $x^* \in \Xi^*$. In particular, Ξ_0, Ξ^* accounts for any possible configuration of the robot (in terms of position, velocity and orientation) on the two platforms. **4.24(a)**, $(x_0, y_0) = (0.0, 2.0)$ and $(x^*, y^*) = (20.0, -2.0)$, orientation and velocities are null. **4.24(b)** $(x_0, y_0, \theta_0) = (1.3, 2.0, 0.523596)$ and $(x^*, y^*, \theta^*) = (20.0, -2.0, -0.087266)$, but the robot starts with a velocity which is in opposite direction with respect to the target, $\dot{x}_0 = -1.0$, and arrives with a non-null velocity $\dot{x}_T^* = -2.0$.

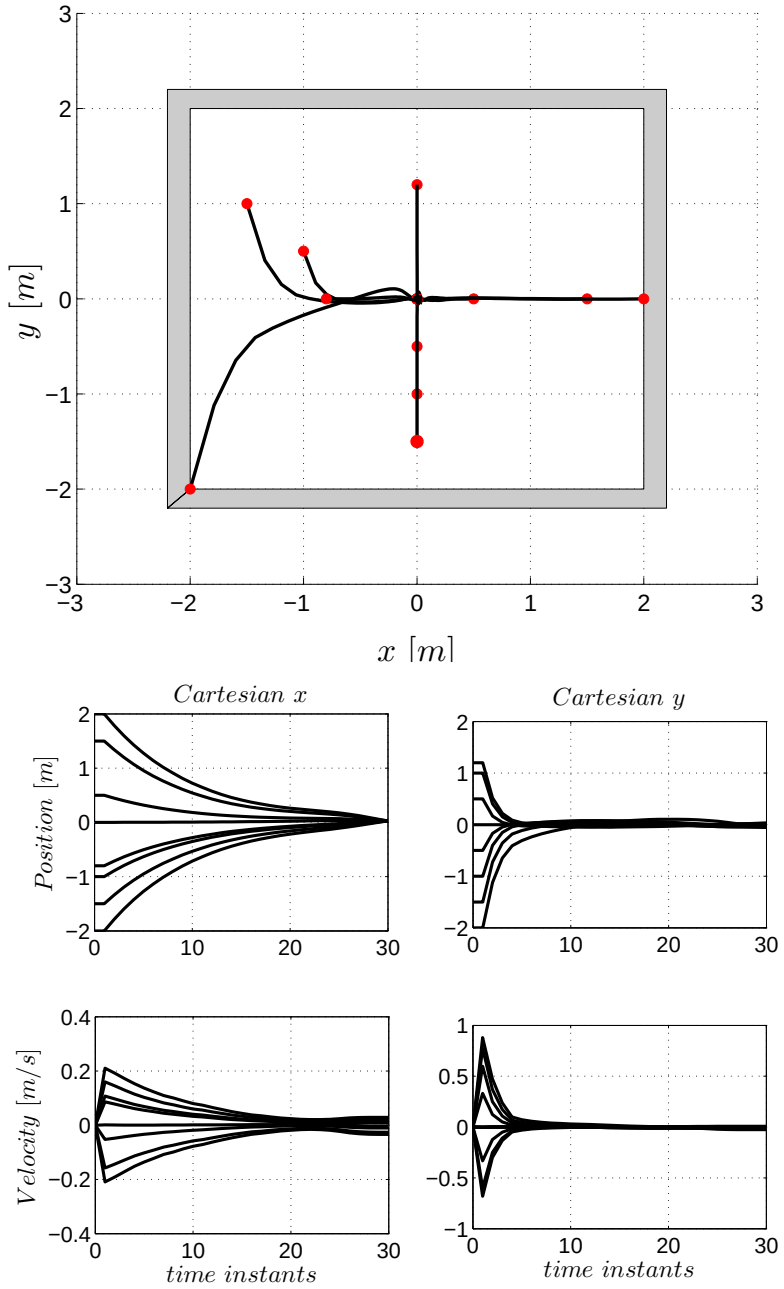


Figure 4.25: FH neural trajectories, where the difference between the robot and the target's configuration is driven to zero in N steps. Each trajectory is found using the outcome of a sequence of N neural control functions, after 10^6 training steps and as many robot and target samples. The red circles are the starting points in the Cartesian space. Some examples of FH trajectories in the planar space are shown.

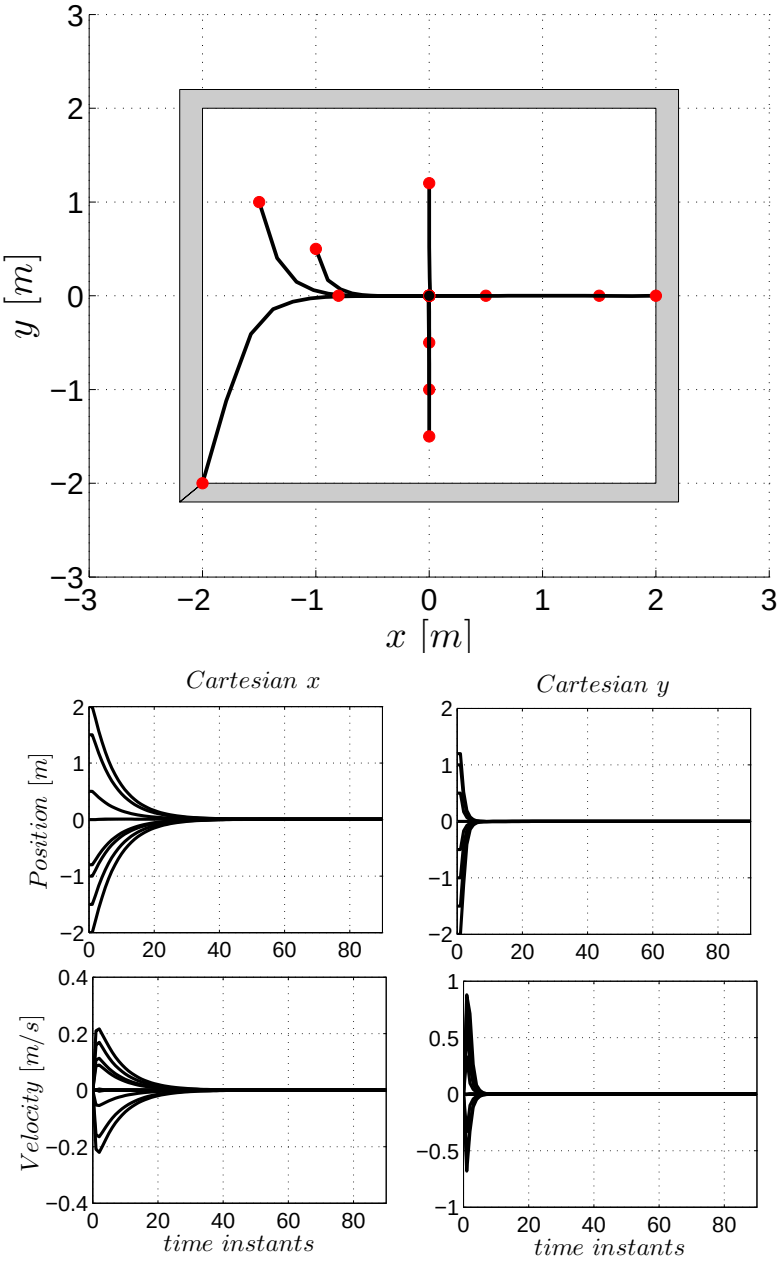


Figure 4.26: RH neural trajectories, using the first neural control function of the FH set in Figure 4.25. The same starting points for the trajectories are shown. Here, only the first neural controller of the ones used in Figure 4.25 is used to generate the RH control at each time instants. The numerical simulation shows the effect of the RH control law for a longer time with respect to the fixed horizon used to compute the control functions. Remarkably, the RH controllers are still "active", though generating null controls, when the target is reached by the robot.

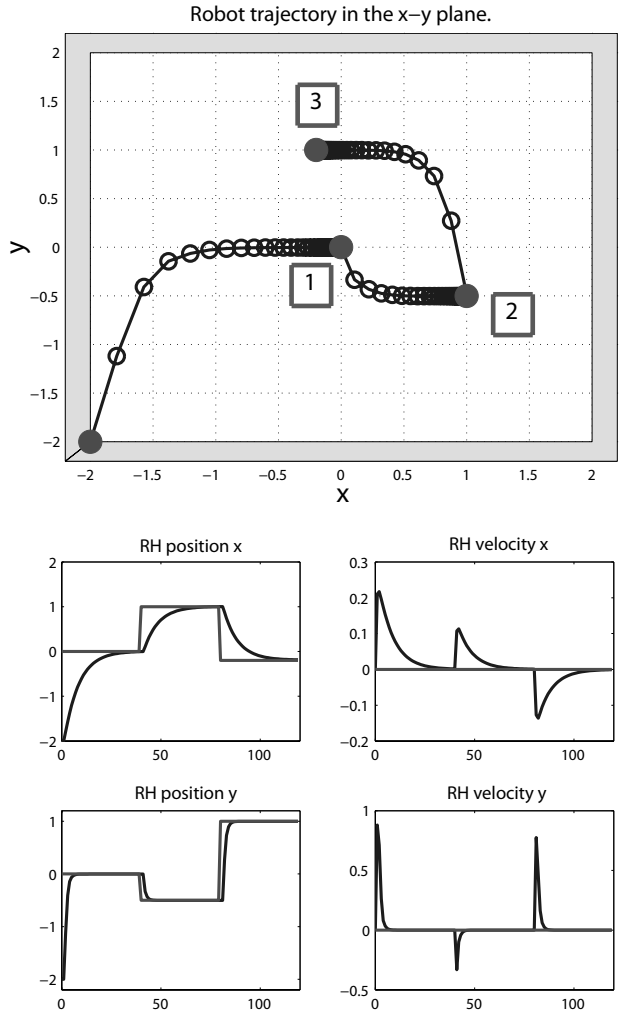


Figure 4.27: The RH controller applied to a mixed tracking/reaching task: the robot moves toward a target which arbitrarily and suddenly changes its position in the state space in an unpredictable way. This case is representative, for example, of the following situation: the mobile robot exploring its workspace (i.e. a room) and while its attentive systems continuously looks for interesting objects that the robot must pick up and take somewhere else.

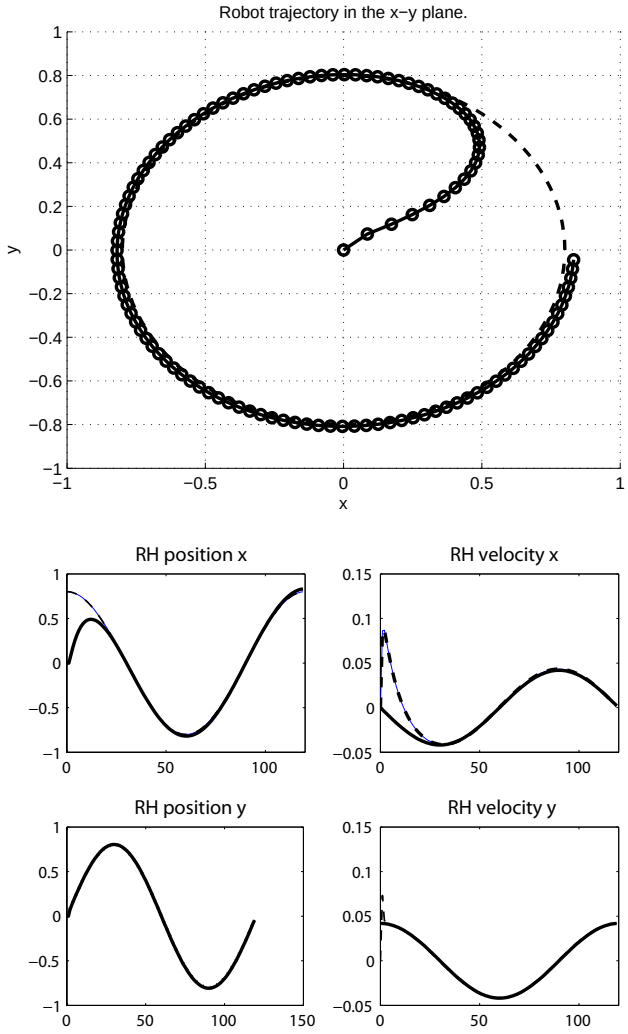


Figure 4.28: The RH controller applied to a tracking task: the mobile robot follows a target, assuming a totally unpredictable motion. (even if here, for the sake of simplicity, the target trajectory here is a simple circle. This case is representative, for example, of a mobile robot following a moving target, according to a predator-prey paradigm.

(as optimality suggests). This is a common problem when training the chain of neural networks (see hereinafter the discussion in Section 4.5.4): possible causes can be the insufficient number of training iteration, or simply the fact the stochastic gradient algorithm stopped at a local minima and both gradient and descent step cannot make the cost lower any further, and consequently enhancing the performance of the neural controls. However, dealing with RH, only the first control law is used and the solution of the FH is only preparatory (i.e. we need to solve the complete chain of neural networks, but once found we retain the first and discard the remaining) so it is possible that even if some control laws are not "optimal" the one needed for the RH is already "trained". This is also a consequence of the learning algorithm, which makes the first neural networks in the chain train faster with respect to the last ones. In fact Figure 4.26 shows the trajectories produced with the RH technique, using the first neural controller of the ones used for the FH trajectories in Figure 4.25. It must be noticed that the time to reach the origin of the planar space is approximately the same of the FH case, and the shape of the control function is similar also. However, it is important to put in evidence that the RH controller remains "active" even when the target is reached: thus, if the robot has reached the target, it keeps controlling even if the difference between the two vectors is zero. The latter is an important property, since it shows (in this regulation example) that $\gamma^{\circ}(0) = 0$.

Finally, Figures 4.27 and 4.28 show the RH controller applied in two different tracking tasks: a multiple reaching case and a pure tracking task, where the target is moving unpredictably.

4.5.3 A two DOF arm actuated by elastic joints

We consider the following system, describing an arm actuated by n flexible/elastic joints:

$$\begin{cases} M(q)\ddot{q} + N(q,\dot{q}) + K(q-\theta) + J^{\top}(q)f &= 0 \\ B\ddot{\theta} + K(\theta - q) &= \tau \end{cases} \tag{4.65}$$

where for the n elastic joints, $q \in \mathbb{R}^n$ and $\theta \in \mathbb{R}^n$ are the generalized coordinates of the driven links and actuators. A simplified model is assumed, where $B = \mathrm{diag}(b_1, \ldots, b_n)$ is the inertia matrix of the actuators, $\tau \in \mathbb{R}^n$ are the motor torques. $M(q)$ is the inertia matrix of the manipulator links, while $N(q,\dot{q})$ contains the centrifugal, Coriolis and gravity forces. f is a generic external force field, e.g. a divergent force field, perceived by the end-effector. K is the stiffness matrix, generally symmetric and definite positive: here, we assume $K = \mathrm{diag}(k_1, \ldots, k_n), k_i > 0 \forall i$. Given the invertibility of the mass matrices M, B, Eq. 4.65 can be written as:

$$\begin{cases} \ddot{q} = & -M^{-1}(q)N(q,\dot{q}) - M^{-1}(q)K(q-\theta) - M^{-1}(q)J^{\top}(q)f \\ \ddot{\theta} = & B^{-1}\tau + B^{-1}K(q-\theta) \end{cases} \tag{4.66}$$

which is a control-affine system of the form

$$\dot{x} = a(x) + b(x)u + c(x)f \tag{4.67}$$

where the system state vector x and the control vector u are $x = \mathrm{col}(q, \dot{q}, \theta, \dot{\theta})$ and $u = \mathrm{col}(k, \tau)$ respectively. Of course, (4.67) is in continuous form, and it is straightforward to find its

discrete-time form, for example by applying an Euler's discretization algorithm. The discrete-time state vector is then $\xi_i = \mathrm{col}(q_t, \dot{q}_t, \ddot{q}_t, \theta_t, \dot{\theta}_t, \ddot{\theta}_t), t = 0, 1, \ldots, N-1$, while the control vector is $u_t = \mathrm{col}(k_t, \tau_t), t = 0, 1, \ldots, N-1$, and the generic state equation is

$$\xi_{t+1} = \tilde{a}(\xi_t) + \tilde{b}(\xi_t)u_t + \tilde{c}(\xi_t)f_t \ , \quad t = 0, 1, \ldots, N-1$$

Here, the following situation is considered: a two DOF arm ($n = 2$), performing point-to-point movements with fixed and known initial and final positions. In this example, all joints have variable stiffness, thus K is not constant, but in general is a function of time $K = K(t)$. Indeed, joint stiffness can change in response to different commands from the robot control system, as a function of the system configuration or of some motion criteria. The goal is to find the optimal torque and stiffness profiles that drive the arm from a known initial pose ξ_0^* to a desired ξ_N^* in N stages, while minimizing a given cost function $\mathcal{J}$:

$$\mathcal{J} = \sum_{i=0}^{N-1} R\, u_i^2 + Q\, \xi_i^2 + Q_N\, \xi_N^2$$

where R, Q, Q_N are suitable weight matrices.

This motor control problem is formulated as a FH neural reaching problem, described by Problem 12. The OHL-NN approximating the control laws have $6n = 12$ inputs and $2n = 4$ outputs. The following parameters were used to describe the arm dynamics:

- mass: $m_1 = 1.4$, $m_2 = 1.1$
- link length: $a_1 = 0.3$, $a_2 = 0.33$
- link COM: $l_1 = 0.11$, $l_2 = 0.16$
- link inertia: $I_1 = 0.025$, $I_2 = 0.045$

Moreover $B = I$, while friction has been neglected (or considered null).

Figure 4.29 shows a suboptimal solution, for the case $N = 15$, with $\Delta t = 0.1$s. Controls were generated by N OHL-NN with $\nu = 5$. The cost weights are $Q = \mathrm{diag}[1, 1, 0.0001, 0.0001, 0, 0]$, $Q_N = Q$, $R = \mathrm{diag}[0.001]$. The stiffness profile is highlighted: notably, its variation during the movement recalls co-contraction, that is the human ability to change intrinsic musculo-skeletal compliance. This feature is crucial when dealing with uncertainties and unpredictability in the model.

4.5.4 Discussion of methods and results

The proposed method is effective and has several advantages, in particular in the capability of providing approximating optimal control functions which can be used to implement closed loop control in limited processing and resource machines. The ERIM is potentially capable of managing difficult problems, and the only requirements, preventing its application to FH and RH problems, are the differentiability of all the items involved in the optimization problem (state and cost functions, controls, constraints) and the sufficient smoothness of the control functions to be found. However, its effectiveness is evident only if the complexity of the problem is "small". It is true that other nonlinear optimization methods show similar limitations when dealing with NMPC schemes. However, two are the main drawbacks.

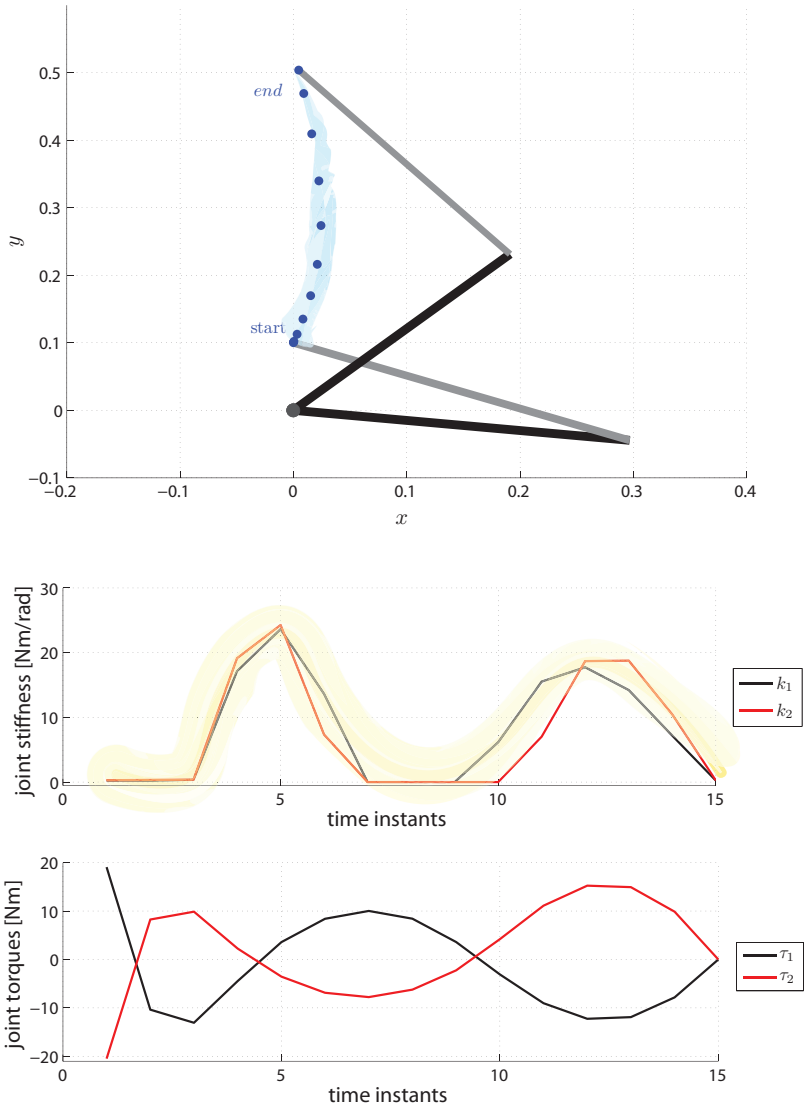


Figure 4.29: A point-to-point neural suboptimal trajectory, for a 2DOF arm actuated by elastic joints. The graphs show the control trajectories of variable joint stiffness and torque.

First the optimal solutions are approximated: local as well as global optima are possible, and it is not possible to distinguish a local from the (or a) global solution during and after the training phase. Second, the stability of the weight update cannot be guaranteed, since a precise analytical condition is missing. It is possible to partially overcome these two issues in practice by re-training the chain of neural networks multiple times, with different values of its update parameters, i.e. changing the learning stepsize, the initialization rule of weights and biases, etc. This approach has been adopted also for Neural Dynamic Optimization [Seong and Widrow, 2001a, Seong and Widrow, 2001b, Seong and Widrow, 2001c], which face the same issues (the two methods are similar). In practical terms, many issues occur, and cannot be neglected since the method has been presented as a universal one and one of the most convenient in solving such optimization problems.

The "curse of dimensionality"

The ERIM is preferred over DP because it can manage a significant number of parameters without incurring in the COD, i.e. the exponential growth of the number of parameters with the complexity of the problem. In Section 4.2, the polynomial growth with respect to the number of parameters was discussed.

The number of parameters to handle is but considerable, and is due to the following facts.

- A plausible sequence of controls must be provided. The number N of neural networks constituting the "chain" and providing controls at each time instant $(t, t+1, \ldots, t+N-1)$ must be chosen considering a finite-time movement (i.e. steering the robot from a starting configuration to a final one) and the controls frequency. Two issues arise. If the frequency is high, i.e. Δt is in the order of ms, hundreds or thousands of time instants must be considered, each corresponding to a "neural" element in the chain.

  Example 2. *The end-effector must perform a movement in T seconds, and controls are sent each Δt ms. For $\Delta t = 5ms$ and $T = 1, 5, 10, 20s$ the number of "neural" elements in the chain is* $200, 1000, 2000, 4000$ *respectively.*

  The second problem is the fact that the duration of the longest movement should be considered in order to set the FH problem properly: unfortunately, the maximum duration cannot be predicted a priori, because it can be context or task dependent (e.g. if you want to catch a moving object, the movement is fast and can last few seconds; conversely a precise pre-grasp movement can last many seconds). In particular, for a FH controller it is fundamental since a too small number of controls N could lead to control sequences which are not able to drive the end-effector to the desired position correctly.

- The NN must have a sufficient number of parameters to approximate correctly the desired optimal control function. In our framework, the function γ to be approximated is the unknown. With the ERIM, we constrain the function to take on a parameterized structure, such that the NN can approximate any possible desired function, but there is no indication on the nature nor on the "shape" of the admissible control function. The underlying assumption is that the control function is continuous and differentiable, so

that it can be approximated by the NN: in general, the smoother the function, the easiest the approximation (i.e. less parameter are necessary to approximate it).

Example 3. *Suppose a polynomial $y(t) = \sum_{i=0}^{N} c_i t^i$ is used to interpolate a function $f(t)$ from noiseless data. If the function is a line, theoretically then only two parameters are necessary, as it can be described by the equation $y(t) = c_0 + c_1 t$. If the function is a cubic, then more are necessary, being $y(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$. The more complex the function, the more parameters are necessary.*

Remark 13. *Some functions are "hard" to approximate with few parameters, e.g. Heaviside's step function, Dirac delta-like splines. In general, rough derivatives require more parameters, or more basis functions.*

A specific theorem assessing the number of parameters of an OHL-NN necessary to approximate a certain function does not exist. However, bounds exist and are mainly related to the smoothness properties of the function to be approximated.. If there are no *a priori* hypothesis on the "shape" of the control functions, and only differentiability is assumed, then the number of "neurons" for each OHL-NN must be chosen according to some practical heuristics. For example, one can solve multiple instances of the same problem, augmenting the number of parameters progressively until certain performances are met.

In general, given an OHL-NN with ν neurons, n inputs, m outputs, the total number of parameters is $N_\nu = (n+1)\nu + (\nu+1)m$. The "chain" used to solve a FH problem contains $T/\Delta t$ neural elements. In this configuration the total number of parameters is:

$$N_{\text{tot}} = \frac{T N_\nu}{\Delta t} = \frac{T(n+1)\nu + T(\nu+1)m}{\Delta t} \tag{4.68}$$

which grows linearly with ν. Still, its growth is not negligible with the problem complexity, and even for a simple problem (like the control of a two or three DOF robot) it is enough large to invoke a COD issue. A graphical representation of the growth of the number of parameters N_{tot} with respect to the problem complexity is shown in Figure 4.30: it can be clearly seen that in a practical situation where $\Delta t = 5ms, T = 10s$ and $\nu = 200$, the total number of parameters is about 4 millions!

Training a chain of neural networks is "hard"

Another drawback is the training complexity, which is a combination of the effects of the number of parameters, the back-propagation algorithm, and stochastic gradient. For the sake of simplicity, the stochastic element will be discussed later on, and here the focus is only on the back-propagation algorithm. Denoting as o_F, o_B the number of operations required for a single forward and backward phase (on a single NN) respectively, $T/\Delta t$ the number of neural elements in a chain, and K the number of iterations of the algorithm to compute the optimal parameters, the total number of operations for the training phase is:

$$o_{\text{train}} = \frac{T\,K}{\Delta t}(o_F + o_B)$$

which encounters the same problem mentioned for the growth of N_{tot}. Moreover, since the algorithm autonomously stops only if the global optimum is reached, or a local one has been reached and a certain stop condition is verified, the number of iterations K is practically unknown, but usually very large. In addition, there is not the guarantee that a single instance of the learning algorithm (one "shoot", i.e. one instance given the random initialization of the parameters and a specific data set) can lead to the solution: actually it is common practice to "re-train" the chain multiple times, changing the learning parameters, e.g. changing the descent rule, the regularization terms, etc., or launching in parallel multiple shoots of the optimization algorithm. Sometimes the neural networks are "stuck", immovable in flat regions where the cost function has a very smooth gradient, so that even considerable changes in the stepsize do not lower the cost. This is a known problem in literature [Gori and Tesi, 1992] and so far a number of heuristics have been suggested, like simulated annealing techniques which basically perturb the descent. However, in a stochastic context the gradient is already "perturbed" and sometimes these techniques are not of any use, like discussed in [Spall, 2003].

Training a chain of neural networks in a stochastic context is "harder"

If training a chain of neural networks and system blocks is hard in a deterministic context, it is even harder to do it in a stochastic context. Difficulties are evident in the simplest case when only the initial state is stochastic, i.e. the initial state takes values from a finite set, $\xi_0 \in \Xi_0$, according to a certain known distribution:

- Because of stochastic steepest descent, it is not guaranteed that the expected cost $\mathcal{F}$ always decreases during training. Recalling the stochastic gradient technique from Section 4.2.3, since $\nabla\mathcal{F}$ cannot be computed exactly, the gradient coming from a single realization of the stochastic variables is used. This means that at each iteration step a "perturbed" gradient is used, instead of the correct one. It is possible to use a better approximation of the gradient, computing its expected value over a certain number M of realizations of the stochastic variable, i.e.

$$\nabla\bar{\mathcal{J}} \approx \frac{1}{M}\sum_{h=1}^{M}\nabla\mathcal{J}(\tilde{\eta}(h), w(k))$$

  where at iteration step k, for the current weights $w(k)$, M realizations of the stochastic variables vector $\tilde{\eta}$ are computed, and forward and backward phases are performed on the chain of neural networks with fixed parameters $w(k)$, so that it is possible to compute an average of all the partial derivatives. Obviously, M must be set balancing the benefits of a better gradient with the computational burden which is introduced by making M additional forward-backward phases. Additional ways to address the problem of global minima of stochastic functions can be found in [Spall, 2003].

- A large number of training, test and validation data is required. The cost functional depends on a set of stochastic variables, whose probabilistic distributions are known. Thus, a significant amount of data is required, because we need to sample properly the whole space of stochastic parameters. If, as frequently happens, the stochastic variable is

the initial (or desired) system state (or both), then a suitable sampling of the state space is required, taking into account the dimension of the space as well as the resolution (i.e. the minimum distance for which two samples are considered as "different") and the samples distribution.

Example 4. *Consider a state space of unitary dimension ($n = 1$) where the state variable θ is an angle, e.g. defining the orientation of a mobile robot with respect to a fixed frame. The variable range is $[0, 2\pi]$. Calling θ_s the sampling resolution, the number of samples (if the sampling is uniform) is $N_{samples} = \frac{2\pi}{\theta_s}$. If the dimension of the state space is n, supposing all the variables are of the same type, then the total number of uniform samples is $N_{samples} = \left[\frac{2\pi}{\theta_s}\right]^n$. A graphical representation is shown in Figure 4.31: note that in a practical situation where for example $n = 6$ and $\theta_s = \pi/180$, $N_{samples}$ is in the order of 10^{15}!*

In general, if the dimension of the state space is n and M the number of samples in a single dimension (supposing the same number of samples for each state variable), the total number of samples is at least $N_{\text{samples}} = M^n$. The quality of the data set may also influence the learning phase. Not only a significant number of data is required, but the sampling type may also affect the training process. In [Fumagalli et al., 2010a] a dissertation on the difference between uniform sampling and random sampling for different machine learning methods was presented. In particular, a random sampling method was compared to a selective one, where a certain "sparsity" of the training set was guaranteed taking a subset, such that the inter-sample distance (computed as the Euclidean distance between the two samples in the standardized input space) was at least bigger than a certain threshold. Experimental results showed that the two methods perform identically for large training sets: the difference between the two was evident only for small amounts of data (very small, considering the problem dimension). A plot about such performances is shown in Figure 4.32.

Initialization of the NN affects the optimization algorithm

Finally, the initialization of the networks certainly plays a role. A randomized initialization is "correct" from a theoretical point of view, since it provides an unbiased starting point for the steepest descent algorithm: having multiple shootings (i.e. multiple instances of the learning algorithm, for different randomized initial conditions) enhances the chance to lead to the global minima avoiding the local ones. However, if there exist hypotheses on the "shape" of the control laws, it is possible to exploit such information to "pre-shape" the approximating networks, e.g. to initialize with a suboptimal solution. In this case, a classical Least Squares algorithm can be used to train the NN [Hagan and Menhaj, 1994], which consists of the following steps:

1. Generate (or measure) L different stochastic variables, such as desired state $\xi^* \in \Xi^*$: $\xi^*(0), \ldots, \xi^*(L-1)$.
2. For each realization $\xi^*(\ell)$, solve the corresponding T-stage optimal control problem (which is deterministic, if the stochastic variables are fixed) and find the deterministic sequence of optimal controls $u_0(\ell), u_1(\ell), \ldots, u_{T-1}(\ell)$

3. Train the T neural networks with a least-square based method: that is, train the t-th neural network with the training set $\{(\xi_t(0), u_t(0), \ldots, (\xi_t(L-1), u_t(L-1))\}$

At the end of the aforementioned procedure, each NN is initialized so as to minimize the least-square error between its prediction and the output from the training set. Basically, this method simply initialize the network with a suboptimal solution found with a supervised learning technique.

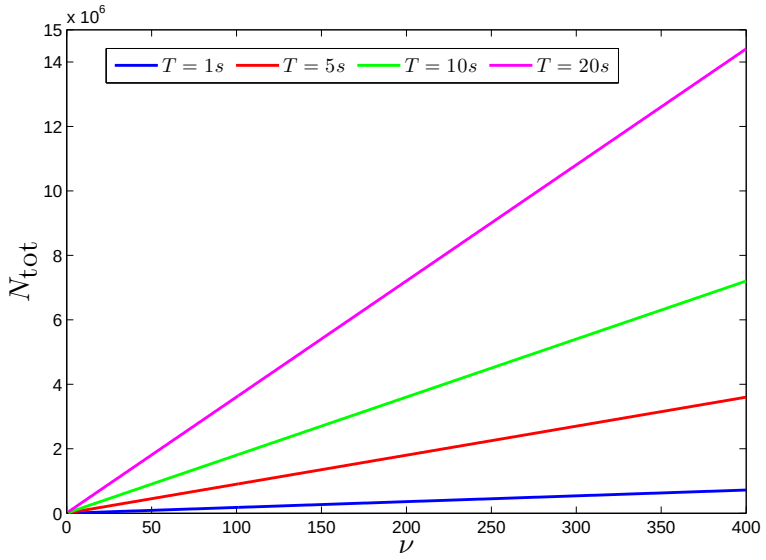


Figure 4.30: The growth of the total number of parameters to be optimized, N_{tot}, with respect to the FH problem complexity: the number of neurons ν, the duration of motion (in seconds) T. Here, $\Delta t = 5ms, n = 6, m = 2$.

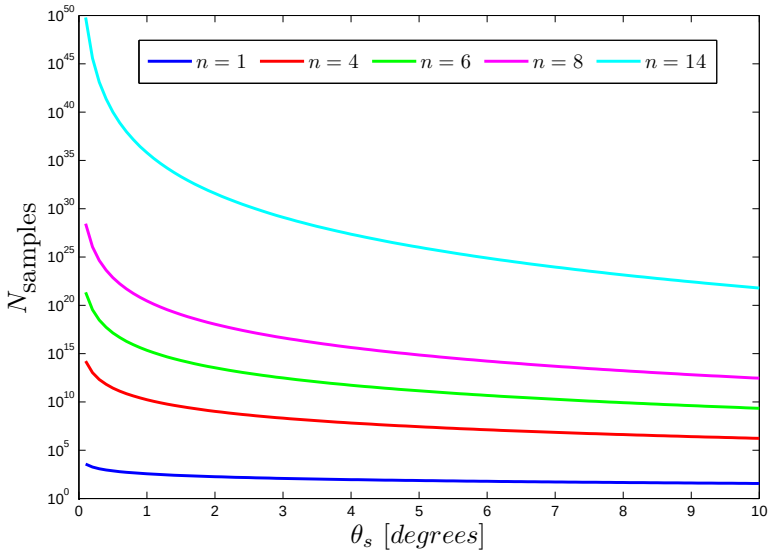


Figure 4.31: The growth of the number of data samples, when the variable is an angle, and its range $[0, 2\pi]$ is uniformly sampled. The number grows exponentially with the dimension of the data space n. Note that the y axis has a logarithmic scale.

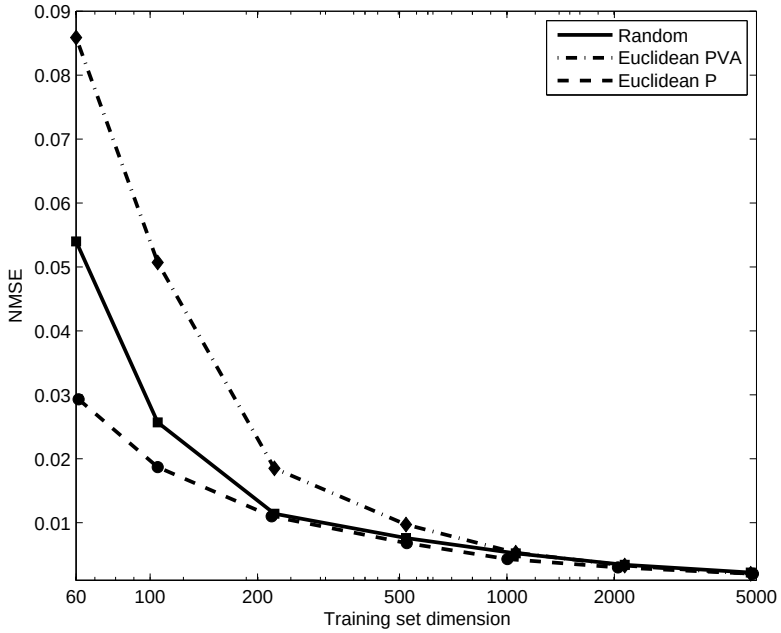


Figure 4.32: Comparison of random and selective subsampling based on standardized Euclidean distance. This image is taken from [Fumagalli et al., 2010a], where the input space of a simple mapping problem (mapping joints position, velocities and accelerations with forces and torques measured by a proximal sensor) was randomly or uniformly sampled. Euclidean P and Euclidean PVA denote subsampling based on Euclidean distance in the standardized input space (using only position or using position-velocity-acceleration). The performance index was the Normalized Mean Squared Error (NMSE) computed on a common validation set. The evidence is that the two different subsampling strategies are equivalent for large training sets. The Euclidean-based sampling outperforms for small data-sets, while the two methods perform nearly identically with the increase of the data-set.

Chapter 5

Motion control on humanoids

5.1 A closed loop control scheme

In Section 4.4 a general approach for *optimal neural motor control* was presented. More precisely, it was shown that a neural approximation of FH and RH control laws can be exploited for optimal planning in a tracking/reaching control scheme, satisfying some performance criterion. It was shown that it is possible to concentrate all the resource demanding computations in an offline phase, and to retrieve almost instantly the controls online, taking into account the feedback on the current status of the system.

In Section 4.5 several examples for different robots were presented. Trajectories were conveniently planned in the task space (e.g. the Cartesian space), and the optimization procedure provided the optimal control law $\hat{\gamma}^\circ$ (and consequently controls $u^\circ(t)$), either found using a FH or RH paradigm, and the optimal system state trajectory. An example of planning in joint space was also discussed, where the dynamics of the robot was fundamental to the task.

In this chapter we will discuss how to combine the neural optimal motion planning with a humanoid robotics control scheme. As anticipated in Chapter 2, most humanoids robots can be controlled through a series of control layers, where different control loops are integrated. In particular, two are the principal methods to control the robot from a "high" level: sending joint velocity commands, or joint torques commands. At lower level (i.e. on the boards) each control type is converted into suitable commands for the actuators, but let us just consider the aforementioned controls as the ones "supported" by the platforms and their architecture without entering into the details of the low level control.

In the above control problems, it was assumed that the desired trajectory was expressed in the joint space or in the task space, and in the latter case that an equivalent description in terms of joint positions, velocities and acceleration was implicitly available. In humanoids robotics, if the desired trajectory for the end-effector is specified in the operational space, it is necessary to interpose some Inverse Kinematics (IK) module to transform operational space references into the corresponding joint space references. If torque control is enabled, a Forward Dynamics (FD) module can further translate them into suitable joint torque commands.

In Figure 5.1, a conceptual scheme combining the neural planning module with IK and FD is shown (where iKin and fDyn represent the IK and FD module respectively). Both IK

and FD can be learned from experimental data, or estimated if an accurate model of the robot kinematics and dynamics is available. Before entering into the detail of these modules, let us briefly overview the main operation they perform.

Let us denote by $x \in \mathbb{R}^m$ the position and orientation of the end-effector and with $q \in \mathbb{R}^n$ the joint angles of the robotic manipulator. The forward kinematics can be generically indicated by $x = K(q)$, while its inverse relation is $q = K^{-1}(x)$. The Jacobian of the manipulator is $J \in \mathbb{R}^{6\times n}$, such that $\dot{x} = J(q)\dot{q}$.

The IK module "translates" the desired Cartesian velocities of the end-effector into proper joint velocity commands This can be easily done by either Jacobian inverse control or Jacobian transpose control, as discussed in [Sciavicco and Siciliano, 2005]. Here we adopt a Closed Loop Inverse Kinematics Controller (CLIK), in the form:

$$\dot{q}^* = J^\dagger v + (I - J^\dagger J)\dot{q}_a$$

where $J^\dagger$ is a regularized damped Least Squares pseudo-inverse of the Jacobian J; $\dot{q}_a$ an arbitrary joint velocity vector projected in the null-space of the Jacobian matrix by the operator $(I - J^\dagger J)$ (I is the identity matrix). The use of the regularized pseudo-inverse rather than the inverse of the Jacobian is motivated by the singularity of some configurations of the manipulator and its redundancy properties. For redundant manipulators, where $n > m$, the solutions to the latter equation are not unique (if they exist), that is there can exist multiple joint configurations corresponding to the same end-effector position (and orientation) in the space. Secondary tasks projected in the null-space of the Jacobian are usually exploited to choose among the possible admissible configuration, solving the redundancy problem. A scheme illustrating how the neural network copes with the CLIK controller is shown in Figure 5.2.

The FD module generally estimates the joint torque commands from the desired joint configurations, including position, velocity and acceleration. If the manipulator dynamics is known, the dynamic equation describing a n-link robot manipulator (according to the standard rigid body model) is:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\,\mathrm{sgn}(\dot{q}) + g(q) = \tau - J^\top(q)h \tag{5.1}$$

where:

- $q \in \mathbb{R}^n$ are the variable joint angles, and $\dot{q}, \ddot{q} \in \mathbb{R}^n$ respectively denote the joint velocities and accelerations
- $B(q) \in \mathbb{R}^{n\times n}$ is the inertia or mass matrix
- $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis matrix
- $F_v \in \mathbb{R}^{n\times n}$ is the diagonal matrix of viscous friction coefficients
- $F_s \in \mathbb{R}^{n\times n}$ is the diagonal matrix of static friction coefficients ($F_s\,\mathrm{sgn}(\dot{q})$ is a simple model of Coulomb friction torques)
- $g(q) \in \mathbb{R}^n$ the gravitational forces
- $\tau \in \mathbb{R}^n$ the control torques to each joint
- $J(q)$ is the Jacobian matrix
- h is the vector of forces and moments exerted by the end-effector on the environment (if there is no interaction, it is simply null)

The computation of joint torques per se is not an issue, if a precise dynamics model (e.g. a rigid body model) of the manipulator is known (which can be found for example from the CAD model of the robot). The key point is whether a joint torque feedback is available on the robot in order to implement joint force or torque control: if this feedback is not available, it is possible to "cancel" the FD module and command exclusively in the joint velocity space.

The remainder of the chapter is organized as follows. In Section 5.2, the IK module is introduced, along with a detailed description of the CLIK method. In Section 5.4, the FD module will be discussed. In particular, since iCub is not yet provided with joint torque sensing capabilities, Section 5.4 will also illustrate a method to estimate joint torques from a set of inertial and force/torque sensors, which has been used to enable such controls. Finally, Section 5.5 will show some experimental results.

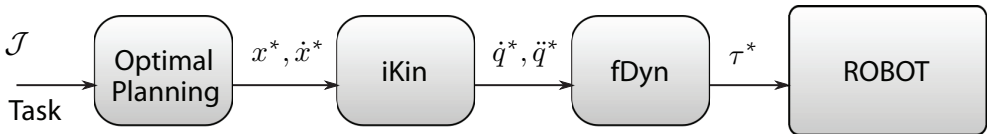


Figure 5.1: A conceptual scheme of a classic layered/hierarchical control scheme for robotics. The task parameters, such as the control function to be minimized, the current status of the robot, the task goal etc. are fed to the optimal planner, which computes the optimal trajectory, typically in the operational space (e.g. Cartesian space). Desired velocity in the operational space is converted into desired velocity in the joint space, by means of an Inverse Kinematic layer. Joint velocity commands can be converted into joint torque commands (if the robot architecture support torque control) by a Forward Dynamics layer. In this scheme, feedback loops are not voluntarily depicted.

5.2 Closed Loop Inverse Kinematics

Given a desired trajectory for the end-effector, computed by the optimal neural controller and denoted by $x^*(t)$, with its velocity profile $\dot{x}^*(t)$, Cartesian and joint space velocity commands, $v(t)$ and $q^*(t)$ respectively, are computed with a Closed Loop Inverse Kinematic (CLIK) algorithm, as shown in Figure 5.2, which allows avoiding the "drift" effect due to the discretization of the joints positions [Sciavicco and Siciliano, 2000].

Among the possible ways to invert the kinematics the following is used:

$$\begin{aligned} \dot{q}^* &= J^\dagger v + (I - J^\dagger J)\dot{q}_a \\ &= J^\top (JJ^\top + k^2 I)^{-1} v + (I - J^\dagger J)\dot{q}_a \end{aligned} \tag{5.2}$$

where $J^\dagger$ is a damped least-squares pseudo-inverse of the Jacobian J; $\dot{q}_a$ represents an arbitrary joint velocity vector which is projected in the null-space of the Jacobian matrix by the operator $(I - J^\dagger J)$ (I is the identity matrix). $\dot{q}_a$ is usually chosen such that joints positions are maintained far from their mechanical physical limits, precisely:

$$\dot{q}_a = -k_{\text{null}} \frac{\partial \mathcal{H}}{\partial q}$$

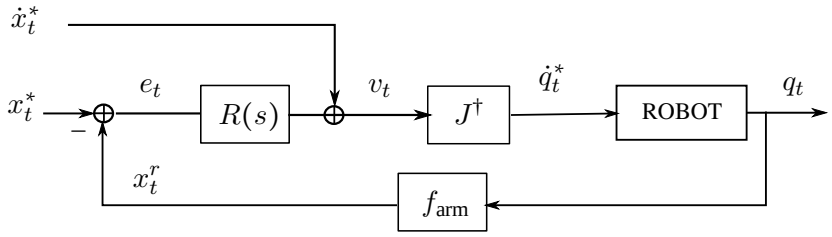


Figure 5.2: A simple CLIK scheme. The block $J^\dagger$ refers to (5.2). The retrieving of the target's cartesian coordinates is not modeled, as it would require to discuss the robotic visual system, the target identification module etc. For the sake of simplicity, many details about the closed loop control are voluntarily neglected, to keep the scheme clear, e.g. the computation of the feed-forward command is not detailed.

which minimize the cost function

$$\mathcal{H}(q) = \frac{1}{2}\sum_{i=1}^{N}\left(\frac{q_i - \bar{q}_i}{q_i^{\max} - q_i^{\min}}\right)$$

where $[q_i^{\min}, q_i^{\max}]$ is the range for the i-th joint, and $\bar{q}_i$ its midpoint. In such manner it is possible to cope with singularities and to exploit the intrinsic redundancy of the manipulator. The parameter k^2 in (5.2) can be determined adaptively in different ways, for example using the condition number of the Jacobian matrix or the manipulability measure [Chiaverini et al., 2008]. In [Sugihara, 2009] a pseudo-inverse Jacobian based on the Levenberg-Marquardt formula was proposed, where the damping factor was found considering the position/orientation error and the singular value decomposition of a suitable weighted Jacobian. The algebraic operation yielded smallest joint deviations in the proximity of the singular configuration of the redundant arm, however the author himself pointed out that the method could lead to physically unfeasible motions, as the continuity of the solution was not preserved. Here[1], we prefer the method proposed in [Chiaverini et al., 1991], where k depends on the smallest singular value $\sigma_{\min}$ of the Jacobian matrix:

$$k^2 = \begin{cases} 0 & \sigma_{\min} > \bar{\sigma} \\ [1 - (\frac{\sigma_{\min}}{\bar{\sigma}})^2]\bar{k} & \sigma_{\min} < \bar{\sigma} \end{cases}. \tag{5.3}$$

For example, in James we set $\bar{\sigma} = 0.20$ and $\bar{k} = 0.10$, after manually driving the arm to singular configurations and studying the singular values, and setting a safety threshold. In fact, it must be pointed out that the smallest singular value can be lower than the threshold $\bar{\sigma}$ even in non-singular configurations of the arm.

Note that the singularity is solved by acting on the singular values, which are configuration-dependent, while redundancy is resolved by selecting the solution which stays furthest away from the joints bounds. A point-wise approach like this may not lead to the best solution for the overall trajectory. It is reasonable to solve the IK this way, since "globally" the control functions are already an approximation of the global ones, and the smoothing properties of the neural networks should prevent rough behaviors.

[1] All cited method were evaluated: best results were found using (5.3).

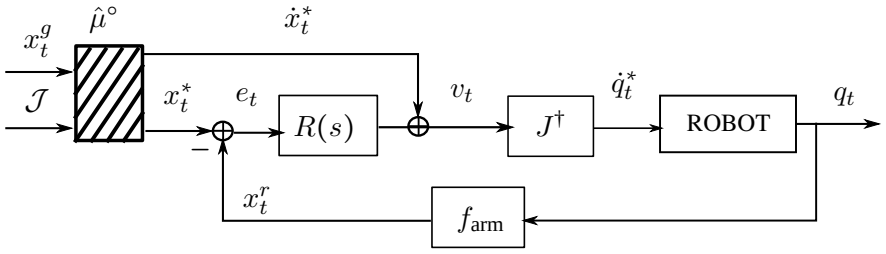


Figure 5.3: James's arm CLIK controller, with the contribution, in evidence, of the neural controller.

The generation of the commanded Cartesian velocities $v(t)$ in (5.2) is shown in Figure 5.2. With an abuse of notation[2], $v(t) = \dot{x}^*(t) + R(s) * e(t)$, where $e(t) = x^*(t) - x(t)$ and $R(s)$ a regulator which will be discussed hereinafter. The classical CLIK scheme relies on a purely proportional regulator, i.e. $R(s) = K_e$, where K_e is a diagonal positive defined matrix. In an ideal situation, the correction term $K_e e(t)$, where $e(t) = x^*(t) - x(t)$, guarantees convergence to zero of the Cartesian error and the error dynamic $\dot{e} + K_e e = 0$ is asymptotically stable. The convergence velocity of such system depends on the eigenvalues of the gain matrix $K_e > 0$ [Chiaverini et al., 2008, Sciavicco and Siciliano, 2000].

When the robot is performing tracking tasks, the discrete-time system must respond to rapid variations of the target trajectory: hence, the gain K_e should be raised until some reasonable performances are met. Unfortunately, K_e cannot be raised *ad libitum*: its upper-limit is determined by the physics of the problem, and high-frequency terms, delays and unmodeled dynamics also prevent to increase K_e to a desired value. For these reasons the simple proportional gain K_e is substituted with the following:

$$R(s) = K_e \frac{1 + s\tau_e}{s} \tag{5.4}$$

where K_e is still a positive diagonal matrix, τ_e a time constant. Incidentally, (5.4) corresponds to a PI controller, where the proportional gain is $K_e\tau_e$ and the integral one is K_e. The sampling time Δt is then fundamental for the proper tuning of the digital integral gain. The time constant τ_e (corresponding to the zero $-1/\tau_e$) can be then manually adjusted (e.g. looking at step and ramp response, and their transient trajectories), so as to raise the proportional gain while preserving the system safety.

In general, regulator (5.4) can guarantee asymptotic stability with faster response of the system with respect to the purely proportional regulator. The commanded Cartesian velocities are then:

$$v(t) = \dot{x}^*(t) + K_e\tau_e e(t) + K_e \int e(t)dt. \tag{5.5}$$

Also in this case, it is possible to guarantee stability and convergence, for $\tau_e > 0, K_e > 0$ and the trend of convergence depends on the two eigenvalues resulting from $s^2 + k_e\tau_e s + k_e = 0$ ($K_e = k_e I$).

The semi-global stability of the regulator $R(s)$ by means of a Lyapunov function can be also proved. Given a desired trajectory $x^* \in C^1$, where $x^*(t), \dot{x}^*$ are bounded, define with

[2] $R(s)$ denotes a Laplace transform of the regulator R.

$e(t) \triangleq x^*(t) - x(t)$ the trajectory error, such that $\dot{e} = \dot{x}^* - \dot{x} = \dot{x}^* - J\dot{q}$. Consider the following candidate Lyapunov function $V(e) = \frac{1}{2}e^\top e > 0$ ("globally" definite positive, since $V(0) = 0$), with $\dot{V} = e^\top \dot{e} = e^\top(\dot{x}^* - J\dot{q})$. Take the following velocity

$$\dot{q} = J^\dagger \left(\dot{x}^* + K_e \tau_e e + K_e \int e(t)dt \right) ,$$

then substitute it into $\dot{V}$:

$$\begin{aligned} \dot{V} &= e^\top [\dot{x}^* - JJ^\dagger (\dot{x}^* + k_e \tau_e e + k_e \int e(t)dt)] \\ &= -e^\top K_e \tau_e e - e^\top K_e \int e(t)dt \\ &\leq -k_e \tau_e \|e\|^2 - k_e \alpha \|e\| \end{aligned}$$

where $\alpha \triangleq \int e(t)dt$. Assume that $\exists \alpha \ : \ \int e(t)dt \leq \alpha \|e\|$. If V is a Lyapunov function, i.e. $\dot{V} \leq 0$, then $e = 0$ is a globally stable equilibrium state. Since $k_e, \tau_e > 0$, the sign of α determines the stability property. If $\alpha > 0$, then $\forall \|e\|, \dot{V} \leq 0$, since $\dot{V}$ is a quadratic function, with $\dot{V}(0) = 0$. If $\alpha < 0$, then $\dot{V}$ is still a quadratic function, with $\dot{V}(0) = 0$, but $\dot{V}(e) > 0$ for $0 < \|e\| < \frac{\alpha}{\tau_e}$: in this region, there is no attractiveness, however it is possible to shorten that region to a desired, by increasing suitably τ_e. In this case, $e = 0$ is semi-globally stable. □

5.3 Forward Dynamics

In Section 5.2 we presented a method for converting desired trajectories in the operational space into desired trajectories in the joint velocity space. The latter can be either sent to the robot, to a joint velocity control interface, or further translated into joint torques commands, if there is a force/torque control interface is available. Furthermore, desired trajectories can be planned directly in torque space, considering a nonlinear model of kinematics and dynamics of the system. In each of the following cases, it is necessary to compute the Forward Dynamics of the robot, i.e. the mapping between its proprioceptive configuration (joint positions, velocities and accelerations) and joint torques. Of course, a suitable dynamics model (e.g. a rigid body dynamics model) must be known.

The classical dynamic equation describing a n-link robot manipulator is described by Eq. 5.1. A notable property is the linearity of the model with respect to the dynamic parameters which characterize the manipulator, particularly in the absence of external forces ($h = 0$) (5.1) can be written as:

$$\tau = Y(q, \dot{q}, \ddot{q})\pi \tag{5.6}$$

where $Y \in \mathbb{R}^{n \times p}$ is the *regressor* and $\pi \in \mathbb{R}^p$ is the vector collecting the set of constant parameters which describe the manipulator dynamics [Sciavicco and Siciliano, 2005].

Such parameters, including link mass, inertia, Center of Mass (COM) location, etc., can be usually retrieved from the CAD model of the robot. Frequently, they are partially known, or known with some uncertainties: in this case, supervised learning techniques such as Support Vector Machines (SVM) and NN can be used either to find the best set of parameters or

to directly approximate the forward dynamics of the robot. An interesting analysis and comparison of model based versus supervised learning technique for this problem can be found in [Fumagalli et al., 2010a], and will be shortly discussed in Section 5.3.1.

5.3.1 Robot dynamics: model or learning?

The robot dynamics, expressed by $Y(\cdot)$ in equation Eq. 5.1, can be identified either deriving it analytically, or approximating it using a set of experimental data and a machine learning technique. In the latter case, the learning algorithm is agnostic to the underlying dynamics model that is used to produce the examples, but the main advantage is that nonlinear effects do not need to be explicitly modeled, as these are learned implicitly by the algorithm. In this section[3], we will report some significant results about a comparison of the model-based approach and two machine learning algorithms, applied to the identification of a robotic arm dynamic model:

1. **Model-Based Approach**

   Eq. 5.1 defines torques as a linear product of matrix $Y(q, \dot{q}, \ddot{q})$ and vector π: the first depends solely on the joint positions, velocities and accelerations, whereas π contains the dynamical parameters that we must estimate to describe the manipulator dynamics [Kozlowski, 1998]. In particular, π is the minimum set of identifiable parameters, i.e. a linear combination of a multitude of elements, containing products among each link mass $m_i \in \mathbb{R}$, inertial parameters $I_i \in \mathbb{R}^6$ (the inertia matrix can be defined by six parameters because of its symmetric properties), Center Of Mass $C_i \in \mathbb{R}^3$ (COM - in form of a distance vector between the COM frame and the reference frame of the link, eventually a roto-translational matrix if frames have also different orientations), length, and so on[4]. The system dynamical parameters π can be often retrieved from an accurate model of the robot (e.g. CAD drawings), but they are not generally accurate, hence a weighted linear Least Squares technique can be used to improve their estimate. Given L samples, consisting of measurements $(\tau_\ell, q_\ell, \dot{q}_\ell, \ddot{q}_\ell)$, the set of optimal parameters can be found as:

   $$\pi^\circ = \arg\min_{\pi} \sum_{\ell=1}^{L} (\tau_\ell - Y(q_\ell, \dot{q}_\ell, \ddot{q}_\ell)\, \pi)^\top \omega (\tau_\ell - Y(q_\ell, \dot{q}_\ell, \ddot{q}_\ell)\, \pi) \quad .$$

   where ω is a suitable weighting diagonal matrix. The explicit solution is given by:

   $$\pi^\circ = \Delta_\Omega^\dagger \tilde{\tau} = \left[\Delta^\top \Omega \Delta\right]^{-1} \Delta^\top \Omega \tilde{\tau} \quad ,$$

   where $\Omega = \mathrm{diag}(\omega)$, $\Delta = \mathrm{col}(Y(q_1, \dot{q}_1, \ddot{q}_1), \ldots, Y(q_L, \dot{q}_L, \ddot{q}_L))$, and $\tilde{\tau} = \mathrm{col}(\tau_1, \ldots, \tau_L)$.

2. **Least Squares Support Vector Machines for Regression**

[3] The detailed description of the methods can be found in [Fumagalli et al., 2010a].

[4] Each kinematic chain link has an associated reference frame, defined by the Denavit-Hartenberg convention [Denavit and Hartenberg, 1955, Lagarde et al., 2009]. All the dynamic and kinematic quantities of each link (COM, inertia, length, etc.) refer to the associated reference frame.

Least Squares Support Vector Machines (LS-SVMs) belong to the class of kernel methods which use a positive definite kernel function to estimate a linear approximator in a (usually) high-dimensional feature space [Suykens et al., 2002]. Let us define the data set $\mathcal{S} = \{x_\ell, y_\ell\}_{\ell=1}^{L}$, where inputs $x_\ell \in \mathbb{R}^n$ and corresponding outputs $y_\ell \in \mathbb{R}$ for $\ell = 1, \ldots, L$. LS-SVM estimates a linear decision function of the form $f(x) = \langle w, \phi(x) \rangle + b$, where b is a bias term and $\phi(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^f$ maps samples from the input space into a (usually) high-dimensional feature space. The weight vector w and bias b are chosen such that both the squared norm of w and the sum of the squared errors $\epsilon_\ell = y_\ell - f(x_\ell)$ are minimized, which is found by solving a dual optimization problem in the form

$$\text{maximize } \frac{1}{2}\|w\|^2 + \frac{1}{2}C\sum_{\ell=1}^{L}\epsilon_\ell^2 - \sum_{\ell=1}^{L}\alpha_\ell\left(\langle x_\ell, w\rangle + b + \epsilon_\ell - y_\ell\right) \quad ,$$

where $\alpha_\ell \in \mathbb{R}$ are the Lagrange multipliers associated with each sample. The decision function can be rewritten as $f(x) = \sum_{\ell=1}^{L}\alpha_\ell \langle \phi(x_\ell), \phi(x)\rangle + b$. Hence, a kernel function $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle$ can be used to implicitly map the data into the feature space. Given a kernel matrix $K = \{k(x_i, x_j)\}_{i,j=1}^{L}$, the solution to the optimization problem in (2) is given by a system of linear equations:

$$\begin{bmatrix}\alpha \\ b\end{bmatrix} = \begin{bmatrix}K + C^{-1}I & 1 \\ 1^T & 0\end{bmatrix}^{-1}\begin{bmatrix}y \\ 0\end{bmatrix} \quad .$$

which is reduced to a $(L+1) \times (L+1)$ matrix inversion, solved efficiently by state of the art algorithms such as Cholesky decomposition [Cawley, 2006].

3. **Neural Networks** Lastly, a multiple input - multiple output OHL-NN can be used, for its generalization and approximation capabilities [Hornik et al., 1989], and de-noising property when dealing with experimental data. Given a batch data set, a typical training algorithm for NN is based on the well known Levenberg-Marquardt (LM) algorithm [Levenberg, 1944, Marquardt, 1963], where the criterion for training the network (that is to find the optimal parameters w°) is to minimize the mean squared error between the estimated and the measured data:

$$w^\circ = \arg\min \Phi(w) = \arg\min \frac{1}{2}\sum_{i=1}^{L}\epsilon_\ell^\top(w)\epsilon_\ell(w) \quad ,$$

where $\epsilon_\ell(w) = \tau_\ell - \hat{\mu}(\tilde{\tau}_\ell, w)$ is the error between the measured and the predicted data, estimated by a NN $\hat{\gamma}(\cdot, w)$, with the same structure seen in Chapter 4. The iterative algorithm consists in a back-propagation of the error function, to compute its partial derivatives with respect to w, and a weight update equation

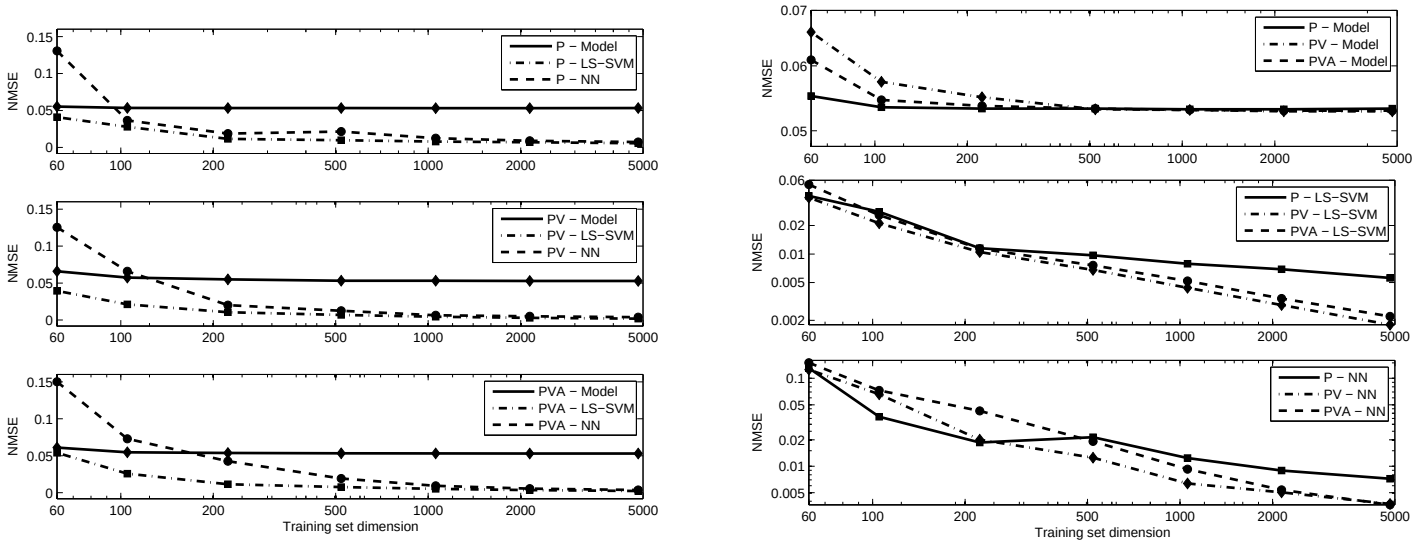
$$w_{k+1} = w_k - \left[J^\top(w_k)J(w_k) + \mu I\right]^{-1}J^\top(w_k)\epsilon(w_k) \quad ,$$

where $\epsilon(w_k) = [\epsilon_1(w_k), \ldots, \epsilon_{L-1}(w_k)]$, and $J(w_k) \in \mathbb{R}^{L\times W}$ is the Jacobian matrix of the errors with respect to the parameters of the NN. The parameter μ, adjusted iteratively, balances the LM between a steepest descent and a Gauss-Newton algorithm.

Generally, learning algorithms outperform the rigid body dynamic model in terms of prediction accuracy, given that a sufficient amount of training data is available. The generalization performance of these methods improves steadily as more training samples become available. LS-SVM converges slightly faster than Neural Networks, but their final performance on large data sets is nearly identical. The model-based method, on the other hand, requires very few samples to achieve acceptable predictions.

In [Fumagalli et al., 2010a], the three methods have been evaluated experimentally on a common data set that has been gathered during a sequence of random arm movements, performed in joint space by the humanoid robot James. Without entering into the details of the experiment and the full comparison among the methods, we report here some interesting remarks:

- **Number of Training Samples**: the two learning methods have a strong dependency on the size of the training set. They consistently improve performance with the increase of the training set, eventually outperforming the model-based approach by an order of magnitude, as shown in Figure 5.4(a); whereas the model-based approach appears to perform at a constant level, regardless of the number of samples. This means that the model-based approach is the preferred approach when only very few samples are available.
- **Contribution of Velocity and Acceleration on the Estimation**: including joint velocities and accelerations does not always improve the generalization performance of the learning methods when training is done on a small number of samples. This is probably due to the fact that learning algorithms require an increasing amount of training samples to make effective use of this additional information (i.e. the COD [Duda et al., 2001]). Figure 5.4(b) shows that both LS-SVM and NN use joint velocities to improve their predictions only if given a sufficiently large training set. Joint accelerations do not seem to contribute positively to the estimation, but this is probably caused by the experiment itself, where motions were smooth, and by the fact that accelerations were not measured directly but derived from positions (thus, very noisy).
- **Selective subsampling**: to avoid data oversampling in an abundance of training data, a selective subsampling strategy was designed to remove samples that were nearly identical to each other, which particularly affects LS-SVM performance. A certain "sparsity" of the training set was guaranteed by taking a subset, such that the inter-sample the Euclidean distance in the standardized input space was at least a threshold t. When this distance is determined solely based on the joint positions, then Euclidean subsampling results in a significant improvement for small data sets. For large data sets, and thus a small inter-sample threshold, the Euclidean and random subsets have very similar sample distributions and therefore similar performance. In contrast, random subsampling performs better than Euclidean subsampling based on joint positions, velocities and accelerations. Some results are shown in Figure 5.5.



(a) (b)

Figure 5.4: Comparison of the three methods on random training subsets of increasing dimension and three different input spaces. Four joints of James' arm were involved in the experiment. *P* denotes the input space containing only joint positions ($q \in \mathbb{R}^4$), *PV* contains both joint positions and velocities ($q, \dot{q} \in \mathbb{R}^8$), and *PVA* contains joint positions, velocities and accelerations ($q, \dot{q}, \ddot{q} \in \mathbb{R}^{12}$). Note that in **5.4(b)** both axes are in logarithmic scale to accentuate differences in final performance.

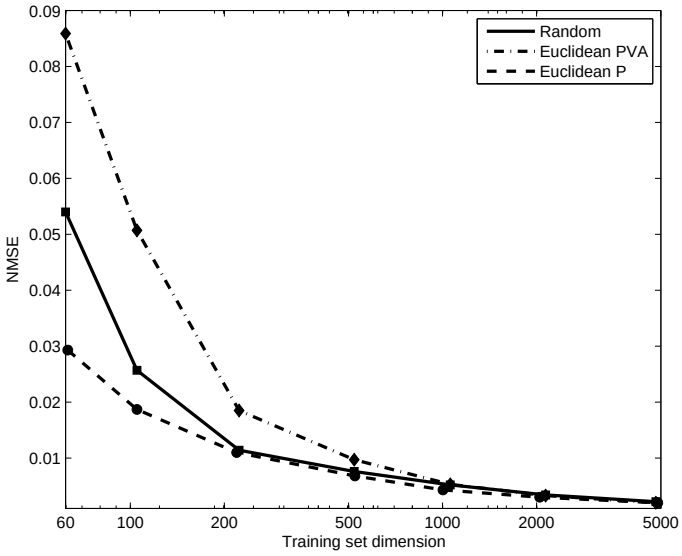


Figure 5.5: Comparison of random and selective subsampling based on standardized Euclidean distance. *Euclidean P* and *Euclidean PVA* denote subsampling based on Euclidean distance thresholds $t = \{1.35, 1.15, 0.88, 0.65, 0.5, 0.35, 0.18\}$ using position inputs and thresholds $t = \{6.0, 5.3, 4.5, 3.7, 3.1, 2.5, 1.8\}$ using position-velocity-acceleration inputs, respectively.

5.4 Force/Torque feedback for control

Once the FD of the robot is known, it is possible to compute the desired torque commands for an optimal trajectory. In an ideal case, where the robot dynamics is perfectly known and all possible interactions with the environment are perfectly measurable (i.e. we have a full knowledge of the system) one could send torque commands in open-loop. In real situations, a closed loop feedback control must be implemented.

When force or torque commands are applied to the robot, its behavior is grounded on the sensory system: joint torque sensors rather than force sensors (the latter typically placed at the end-effector), are required. Exploiting the sensors information, the robot not only can carry out force regulation, but also react safely whenever unexpected contacts occur during task execution [Siciliano and Villani, 1996, De Luca, 2006, Haddadin et al., 2010b, Mistry et al., 2010, Calinon et al., 2010, Fumagalli et al., 2010a].

Traditionally, torque feedback is provided by specific joint torque sensors, distributed over the entire structure of the robot, measuring the momentum component which works along the axis of rotation of the joints. However, retrieving such measurements using localized torque sensors might not allow a full perceptual representation of the interaction scenario, in terms of forces and torques which rise over the whole structure. Even if torque sensors are distributed over the entire structure of the robot and can measure the internal dynamic as well as the

interaction occurring on their link, they can measure a single component of momentum which works along the axis of rotation of the joints. More specifically, since forces and torques are linearly related by the transposed Jacobian, which might have a non-empty null-space, there exist singular configurations where some interaction forces at sensory level cannot be fully retrieved (e.g. pure forces working on a direction which is parallel to the joint torque sensor axis are *hidden*).

A more robust and complete representation of the interaction forces can be retrieved by Force/Torque Sensors (FTS). Classically, robots are equipped with six-axis FTS mounted in their end-effectors, where the most interaction with the environment occurs during manipulation [Sciavicco and Siciliano, 2005]. This solution is compact and less invasive with respect to the design of joint torque sensors [Parmiggiani et al., 2009, Luh et al., 1983], allows a complete representation of the interaction and thus the achievement of active compliance [Caccavale et al., 2005, Chiaverini et al., 1999, Siciliano and Villani, 2000]. However, this information is localized at the tool level. In other words, a FTS at the end-effector does not allow retrieving neither the information about the manipulator dynamics, nor about the potential interaction occurring on any link of the robotic system. In this situation, this information must be retrieved with other sensors.

If a FTS is located proximally, for example in the middle of a kinematic chain, it is possible to exploit its measurements in a different way, and basically "propagate" its measures through the chain. A simple way consists in "projecting" its measurements on the joints through the transposed Jacobian. A more complete representation can be found by applying recursively Newton's laws, and will be briefly introduced in Section 5.4.1. However, the analytical solution for computing joint torques becomes quite complex for more complicated kinematic structures.

In Section 5.4.2, we describe instead a procedure which allows retrieving a more complete and better representation of the interaction forces over the entire structure of single and multi-branched kinematic chains, such as a humanoid robot. The proposed approach makes use of three sets of sensors, distributed along the kinematic chain:

- *force/torque*, used to measure dynamical wrenches, i.e. the forces and moments that are due to the dynamics of the structure;
- *inertial*, since their measurements can be propagated through the chain if the kinematic model of the robot is known, thus allow retrieving a complete description of the kinematics information of the links in the manipulator;
- finally, distributed *tactile* sensors provide the information about the location of contacts occurring dynamically with the environment.

Under suitable assumptions, we will show that by exploiting these sensors it is possible both to detect external wrenches and to obtain complete information of the wrenches transmitted along the structure (and thus also joint torques). Basically, if a precise dynamical model of the robot is known (i.e. a rigid body model), internal forces and torques can be computed by means of recursive algorithms, such as the classical *Recursive Newton-Euler Algorithm* (RNEA) [Sciavicco and Siciliano, 2005]. To the best of our knowledge, a similar solution has been adopted only once in [Morel and Dubowsky, 1996, Morel et al., 2000] where a single FTS placed at the base of a 3 Degrees Of Freedom (DOF) PUMA manipulator was used to estimate

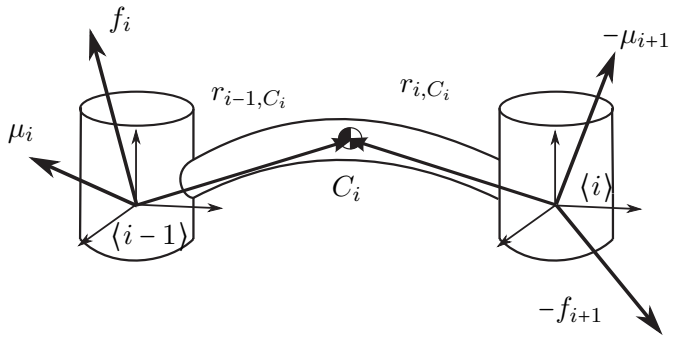


Figure 5.6: Notation for the i-th link of a kinematic chain. A more complete description can be found in [Sciavicco and Siciliano, 2005].

the joint torques. In Section 5.5.2, some experiments are shown, where the proposed method has been applied to estimate forces and torques on 32 of the 53 DOF of the iCub. Moreover, we enrich the estimation by also computing external forces at any contact location. This information can be fixed *a priori* for particular robot tasks, but in general must be updated on-the-fly: in iCub it is provided by its "artificial skin". In the remainder of this section we will introduce the Enhanced Oriented Graph (EOG) method, which is used to perform the RNEA computations on both single and multiple branched open kinematic chains, when one or multiple FTS are available. It must be pointed out that the RNEA is here proposed as a tool to compute kinematic and dynamic information recursively, but its adoption is not a must and other recursive algorithms could be used. The proposed method is thus shown to be generic and applicable to every open kinematic tree. Force sensors are used to improve the estimation of the internal wrenches and for the computation of external interaction. The number of FTS to employ and their placement along a manipulation structure is not defined. We only suggest spreading them on different links of the system to increase the quality and reliability of the results. The method, in fact, presents a systematic procedure for computing $N + 1$ external wrenches from N internal wrenches (i.e. measurements from FTS). Remarkably, under some conditions that will be presented in next sections, all link wrenches and joint torques can be theoretically computed.

It is necessary to introduce the notation of the variables required for computing the internal and external forces and torques. Given a force $f \in \mathbb{R}^3$ and a moment $\mu \in \mathbb{R}^3$, a wrench $w \in \mathbb{R}^6$ is the vector $w = \mathrm{col}\,(f, \mu)$. FTS, which actually measure a wrench, are named according to the physics terminology, where μ is called torque. To discriminate from the joint torque τ, we call μ moment accordingly to the mechanical terminology. The description of the kinematics and the dynamics of a link (see Figure 5.6) adopts the Denavit-Hartenberg notation. We limit the discussion to revolute joints for the sake of simplicity, but the method is generic for both revolutionary and linear joints. Here is a list of the adopted symbols:

- $\langle \cdot \rangle$ generic Cartesian reference frame
- v^a given $v \in \mathbb{R}^n$ a generic n-dimensional vector, v^a is v expressed in $\langle a \rangle$
- R_a^b the $SO(3)$ rotation matrix from $\langle a \rangle$ to $\langle b \rangle$
- $r_{a,b}$ distance vector r from $\langle a \rangle$ to $\langle b \rangle$

z_i z-axis of $\langle i \rangle$, aligned with the axis of rotation of joint i

θ_i the angle associated to the i-th joint

$\ddot{p}_i \in \mathbb{R}^3$, denotes the linear acceleration of $\langle i \rangle$

$\omega_i, \dot{\omega}_i \in \mathbb{R}^3$, the angular velocity and acceleration of $\langle i \rangle$

m_i mass associated with the i-th link

$\bar{I}_i^i \in \mathbb{R}^{3\times3}$ represents the inertia tensor of the i-th link, defined with respect to the center of mass oriented as the frame $\langle i \rangle$

$C_i \in \mathbb{R}^3$ the coordinate vector of the center of mass of link i-th, with respect to $\langle i \rangle$

$f_i \in \mathbb{R}^3$, represents the forces applied on $\langle i \rangle$, that link $i+1$ exert on the i-th link

$\mu_i \in \mathbb{R}^3$, represents the moment applied on $\langle i \rangle$, that link $i+1$ exert on the i-th link

$\tau_i \in \mathbb{R}$ the joint torque, i.e. the component of μ_i along z_i

$w_i \in \mathbb{R}^3$ is the wrench $w = \left(\begin{smallmatrix} f \\ \mu \end{smallmatrix}\right)$ applied on $\langle i \rangle$, that link $i+1$ exerts on link i

5.4.1 Wrench transformations and FTS measurements

Consider a kinematic chain, where a FTS is embedded in the i_S-th link as shown in Figure 5.8(b). Name the sensor frame $\langle s \rangle$ and number progressively the links frames $\langle i_S \rangle, \langle i_S + 1 \rangle, \ldots$ as $\langle s+1 \rangle, \langle s+2 \rangle$ and so on: interestingly, all links wrenches $w_{s+k} = [f_{s+k}^{s+k}, \mu_{s+k}^{s+k}]$ can be expressed as function of the FTS measurements $w_s = [f_s^s, \mu_s^s]$ by the compact formula:

$$
\begin{aligned}
w_{s+k} &= A_{s+k}\, w_s + \tilde{w}_{s+k} \\
\tau_{s+k} &= ws^\top\, C_{s+k}\, z_0 + \tilde{\tau}_{s+k}
\end{aligned}
$$

where A_{s+k} and C_{s+k} basically contain the rotations and distance vectors between frames to transform wrenches between different coordinate frames, specifically:

$$
\begin{aligned}
A_{s+k} &= \begin{bmatrix} {R_{s+k}^s}^\top & 0 \\ -{R_{s+k}^s}^\top ({r_{s-1,s+k-1}^{s+k-1}})^\wedge & {R_{s+k}^s}^\top \end{bmatrix} \\
C_{s+k} &= \begin{bmatrix} ({r_{s+1,s+k-1}^{s+k-1}})^\wedge\, R_{s+k-1}^s \\ R_{s+k-1}^s \end{bmatrix}
\end{aligned}
$$

$\tilde{w}_{s+k} = [\tilde{f}_{s+k}^{s+k}, \tilde{\mu}_{s+k}^{s+k}]$ and $\tilde{\tau}_{s+k}$ instead contain all the dynamic terms which are not taken into account by simply applying a wrench transformation between different frames in a rigid-body. Indeed, matrix A_{s+k} is the Adjoint matrix defining a wrench transformation in a rigid body [Murray et al., 1994].

As an example, in the "static" case (i.e. when $\dot{q}_i = \ddot{q}_i = 0, \forall k$) we have:

$$
f_{s+k}^{s+k} = {R_{s+k}^s}^\top\, f_s^s - \sum_{j=0}^{k-1} R_{s+k}^{s+j}\, m_{s+j}\, \ddot{p}_{C_{s+j}}^{s+j}
$$

$$\begin{aligned}\mu_{s+k}^{s+k} = {R_{s+k}^{s}}^{\top}\,\mu_s^s - {R_{s+k}^{s}}^{\top}\,(r_{s-1,s+k-1}^{s+k-1})^{\wedge}\,f_s^s \\ - \sum_{j=0}^{k-1} {R_{s+k}^{s+j}}^{\top}\, m_{s+j}\,(r_{s+j,C_{s+j}}^{s+j})^{\wedge}\,\ddot{p}_{C_{s+j}}^{s+j} \\ + \sum_{j=0}^{k-2} {R_{s+k}^{s+j}}^{\top}\, m_{s+j}\,(r_{s+j,s+k-1}^{s+k-1})^{\wedge}\,\ddot{p}_{C_{s+j}}^{s+j}\end{aligned}$$

$$\begin{aligned}\tau_{s+k} = \Big[\, {R_{s+k-1}^{s}}^{\top}\mu_s^s - {R_{s+k-1}^{s}}^{\top}(r_{s-1,s+k-1}^{s+k-1})^{\wedge}\,f_s^s \\ - \sum_{j=0}^{k-1} {R_{s+k-1}^{s+j}}^{\top} m_{s+j}\,(r_{s+j,C_{s+j}}^{s+j})^{\wedge}\,\ddot{p}_{C_{s+j}}^{s+j} \\ + \sum_{j=0}^{k-2} {R_{s+k-1}^{s+j}}^{\top} m_{s+j}\,(r_{s+j,s+k-1}^{s+k-1})^{\wedge}\,\ddot{p}_{C_{s+j}}^{s+j}\,\Big]^{\top} z_0\end{aligned}$$

which gives:

$$\begin{bmatrix}\tilde{f}_{s+k}^{s+k} \\ \\ \tilde{\mu}_{s+k}^{s+k}\end{bmatrix} = \begin{bmatrix} -R_{s+k}^{s} m_s \ddot{p}_{C_s}^{s} & -R_{s+k}^{s+1} m_{s+1} \ddot{p}_{C_{s+1}}^{s+1} & \cdots \\ \\ -{R_{s+k}^{s}}^{\top}(r_{s,C_s}^{s})^{\wedge} m_s \ddot{p}_{C_s}^{s} & -{R_{s+k}^{s+1}}^{\top}(r_{s+1,C_{s+1}}^{s+1})^{\wedge}\, m_{s+1}\ddot{p}_{C_{s+1}}^{s+1} & \cdots \\ -{R_{s+k}^{s}}^{\top}(r_{s,s+k-1}^{s+k-1})^{\wedge} m_s \ddot{p}_{C_s}^{s} & -{R_{s+k}^{s+1}}^{\top}(r_{s+1,s+k-1}^{s+k-1})^{\wedge} m_{s+1}\ddot{p}_{C_{s+1}}^{s+1} & \end{bmatrix}$$

$$\begin{bmatrix} \cdots & -R_{s+k}^{s+k-2} m_{s+k-2} \ddot{p}_{C_{s+k-2}}^{s+k-2} & -R_{s+k}^{s+k-1} m_{s+k-1} \ddot{p}_{C_{s+k-1}}^{s+k-1} \\ \\ \cdots & -{R_{s+k-1}^{s+k-2}}^{\top}(r_{s+k-2,C_{s+k-2}}^{s+k-2})^{\wedge} m_{s+k-2}\ddot{p}_{C_{s+k-2}}^{s+k-2} & -{R_{s+k-1}^{s+k-1}}^{\top}(r_{s+k-1,C_{s+k-1}}^{s+k-1})^{\wedge} m_{s+k-1}\ddot{p}_{C_{s+k-1}}^{s+k-1} \\ & -{R_{s+k}^{s+k-1}}^{\top}(r_{s+k-2,s+k-1}^{s+k-1})^{\wedge} m_{s+k-2}\ddot{p}_{C_{s+k-2}}^{s+k-2} & \end{bmatrix}$$

and the following torques:

$$\tilde{\tau}_{s+k} = \begin{bmatrix} m_s(\ddot{p}_{C_s}^{s})^{\top}\,(r_{s,C_s}^{s})^{\wedge} - m_s\,(\ddot{p}_{C_s}^{s})^{\top}\,(r_{s,s+k-1}^{s+k-1})^{\wedge}\,R_{s+k-1}^{s} \\ \vdots \\ m_{s+k-2}(\ddot{p}_{C_{s+k-2}}^{s+k-2})^{\top}\,(r_{s+k-2,C_{s+k-2}}^{s+k-2})^{\wedge}\,R_{s+k-1}^{s+k-2} + \\ -m_{s+k-2}(\ddot{p}_{C_{s+k-2}}^{s+k-2})^{\top}\,(r_{s+k-2,s+k-1}^{s+k-1})^{\wedge}\,R_{s+k-1}^{s+k-2} \\ \\ m_{s+k-1}(\ddot{p}_{C_{s+k-1}}^{s+k-1})^{\top}\,(r_{s+k-1,s+k-1}^{s+k-1})^{\wedge}\,R_{s+k-1}^{s+k-1} \end{bmatrix} z_0$$

The “dynamic” case (i.e. when joint velocities and accelerations are not zero, thus during motion) is not explicitly reported for the sake of brevity and simplicity, because the equations become much more complicated. However, even in that case the joint torques can be retrieved just applying the aforementioned formulas recursively.

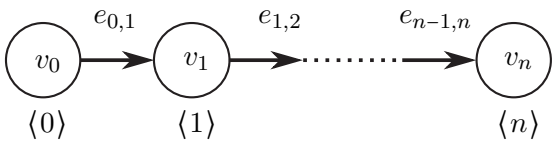


Figure 5.7: An open chain represented as a graph.

5.4.2 Enhanced Oriented Graphs

Graph theory has been extensively used to represent mechanical systems [Wittenburg, 1994, Featherstone and Orin, 2008] and kinematic chains, producing compact and clear models, in matrix forms with beneficial properties (e.g. branch-induced sparsity [Featherstone, 2010]) when the connectivity among its elements is expressed. There is not a unique choice for a graph representing a chain: for example, in [Featherstone, 2007] graphs are undirected, nodes and arcs represent bodies and joints respectively; the resulting graph is undirected (i.e. non-oriented), but nodes are "labeled" according to a "regular numbering scheme".

This section presents the theoretical framework of the *Enhanced Oriented Graphs* (EOG), applied to the computation of both internal and external wrenches applied to single and multiple branches, generally non-grounded, kinematic chains. The proposed method is an extension of the classical RNEA [Featherstone and Orin, 2008, Sciavicco and Siciliano, 2005]. Similarly to the classical approach we represent a kinematic chain as a graph such that computations of the system dynamics can be obtained performing a pre-order and a post-order traversal visit of the graph itself. However, we enhance the graph with specific nodes representing both known and unknown (kinematic or dynamic) variables. Remarkably, not all the unknowns will be specified a-priori (e.g. contacts at arbitrary locations might appear and other contacts might be removed) and therefore the graph structure will be adapted accordingly[5]. This dynamically evolving graphical description of the chain modifies the way the graph is visited during the Newton-Euler recursion, thus changing in particular the direction along which the recursion is propagated in the graph. In order to cope with this evolving representation we introduced another difference with respect to previous RNEA graphical representations by representing the kinematic chain as an *oriented* graph: the direction along which edges are traversed will determine either the use of the classical Newton-Euler recursion formula or a slightly modified version of it.

The enhanced graph representation

We here consider an open (single or multiple branches) kinematic chain with n DOF composed of $n + 1$ links.

Adopting the Denavit-Hartenberg notation [Sciavicco and Siciliano, 2005], we define a set of reference frames $\langle 0 \rangle$, $\langle 1 \rangle$, ..., $\langle n \rangle$ attached at each link.

[5]Within this context, a crucial role is played by the distributed tactile sensor, primarily used to compute the presence and the location of externally applied wrenches. Even if the tactile sensor would be capable of measuring also the component of the force normal to the skin surface, this information is not used in this paper where we focus computing both the applied force and torque (i.e. the whole externally applied wrench) exploiting the embedded sensors.

The i-th link of the chain is described by a vertex v_i (sometimes called node), usually represented with the symbol $\textcircled{i}$. A hinge joint between the link i and the link j (i.e. a rotational joint) is represented by an oriented edge $e_{i,j}$ connecting v_i with v_j: $\textcircled{i} \rightarrow \textcircled{j}$. In a n DOF open chain, each vertex (except for the initial and terminal, v_0 and v_n respectively) has two edge connections. Therefore, the graph representation of the n-links chain is an oriented sequence of nodes v_i, connected by edges $e_{i-1,i}$.

The orientation of the edges can be either chosen arbitrarily (it will be clear later on that the orientation simply induces a convention) or it can follow from the exploration of the kinematic tree according to the "regular numbering scheme" [Featherstone and Orin, 2008], which induces a parent/child relationship such that each node has a unique input edge and multiple output edges. Following the classical RNEA and the classical Denavit-Hartenberg notation, we assume that each joint has an associated reference frame with the z-axis aligned with the rotation axis; this frame will be denoted $\langle e_{i,j} \rangle$. In kinematics, an edge $e_{i,j}$ from v_i to v_j represents the fact that $\langle e_{i,j} \rangle$ is fixed in the i-th link. In dynamics, $e_{i,j}$ represents the fact that the dynamic equations will compute (and make use of) $w_{i,j}$, i.e. the wrench that the i-th link exerts on the j-th link, and not the equal and opposite reaction $-w_{i,j}$, i.e. the wrench that the j-th link exerts on the i-th link (further details in Section 5.4.2). In order to simplify the computations of the inverse dynamics on the graph (see Section 5.4.2), kinematic and dynamic measurements have been explicitly represented. Specifically, the graph representation has been enhanced with a new set of graphical symbols: a triangle to represent kinematic quantities (i.e. velocities and acceleration of links – ω, $\dot{\omega}$, $\ddot{p}$), and a rhombus for wrenches (i.e. force sensors measurements on a link – f, μ). Moreover these symbols have been further divided into *known* quantities to represent sensors measurements, and *unknown* to indicate the quantities to be computed, as in the following:

- $\bigtriangledown$: unknown kinematic information
- $\blacktriangledown$: known (e.g., measured) kinematic information
- $\Diamond$: unknown dynamic information
- $\blacklozenge$: known (e.g., measured) dynamic information

Kinematic variables can in general be measured by means of gyroscopes, accelerometers, or simply inertial sensors. When attached on link i-th, these sensors provide angular and linear velocities and accelerations (ω, $\dot{\omega}$, $\dot{p}$ and $\ddot{p}$) at the specific location where the sensor is located. We can represent these measurement in the graph with a *black triangle* ($\blacktriangledown$) and an additional edge from the proper link where the sensor is attached to the triangle[6]. As usual, the edge has an associated reference frame , in this case corresponding to the reference frame of the sensor. Similarly, an unknown kinematic variable is represented with a *white triangle* ($\bigtriangledown$) with an associated edge going from the link (where the unknown kinematic variable is attached) to the triangle. The reference frame associated to the edge will determine the characteristics of the retrieved unknown kinematic variables as it will be clear in Section 5.4.2.

Similarly, we introduce two new types of nodes with a rhomboidal shape: *black rhombi* ($\blacklozenge$) to represent known (i.e. measured) wrenches, *white rhombi* ($\Diamond$) to represent unknown

[6]According to our kinematic convention an edge $e_{i,j}$ is fixed on the i-th link. Therefore a sensor fixed in the i-th link, will be represented by $e_{i,s}$, i.e. an edge from the link to the sensor.

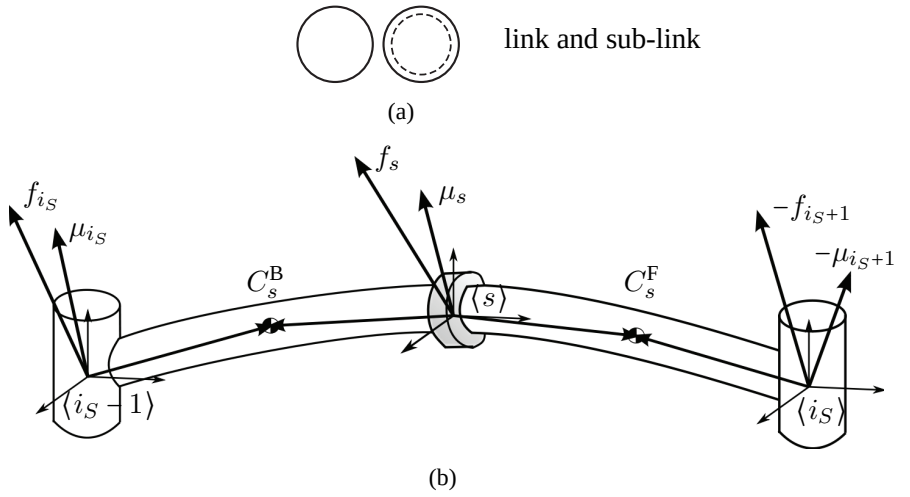


Figure 5.8: 5.8(a): The notation introduced to represent node (vertex, link) and sub-node (sub-link). **5.8(b)**: A representation of a FTS within the i_S-th link. Note that the sensor divides the link into two sub-links, each with its own dynamical properties. In particular, it is evident that the center of mass (COM) of the original link, C_{i_S}, differs from $C_s^{\mathrm{F}}, C_s^{\mathrm{B}}$, i.e. the COM of the two "sub-links".

wrenches which need to be computed. The reference frame associated to the edge will be the location of the applied or unknown wrench.

Remark 14. *There is not a fixed rule to determine the orientation of the edge connecting the rhombi to the graph: according to our convention for representing the wrenches, the edge can be either directed from the rhombus to the link or vice versa depending on the variable we are interested in representing (i.e. the wrench from the link to the external environment or the equal and opposite wrench from the environment to the link).*

It is important to point out that, whereas the position of ⧫ is static within the graph (because sensors are fixed in the manipulator), the location of ◊instead can be dynamic (contact point locations are dynamically detected by the distributed tactile sensor). If a contact moves along a chain, the graph is accordingly modified. This rule shows a big benefit of the EOG, which dynamically adapts in response to the location of the unknown external wrenches.

Within this representation, embedded FTS can be inserted by "cutting" the manipulator chain where the FTS is located and creating two virtual "sub-links" from the link hosting physically the sensor. The EOG is then split into two sub-graphs, where black rhombi (⧫, i.e. known wrenches representing the FTS measures, one per graph) are introduced and attached to the sub-links. In practice, suppose that an FTS is placed in the i_S-th link (see Figure 5.8(b)). Let $\langle s \rangle$ be the frame associated to the sensor. The sensor virtually divides link i_S into two "sub-links" (hereafter denoted forward and the backward sub-links). The sensor therefore measures the wrench exchanged between the "forward" and the "backward" sub-links (this will be represented by two rhomboidal nodes). Under these considerations, the FTS within a link is represented by splitting the node associated to the link into two sub-nodes (with suitable dynamical properties, see Figure 5.9). Two known wrenches in the form of black rhombi are then

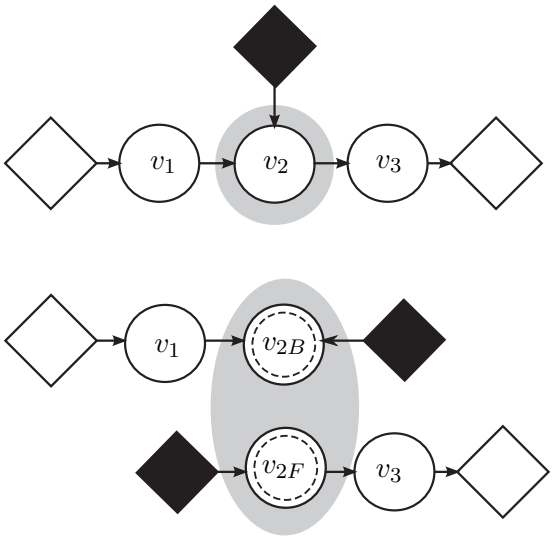


Figure 5.9: The graph shows how to insert an FTS in a graph representation of a kinematic chain. The node on which the sensor is attached (highlighted), is practically divided into two sub-nodes. The graph is divided into two sub-graphs and two black rhombi (known wrenches corresponding to the sensor measurement) are connected to the sub-nodes.

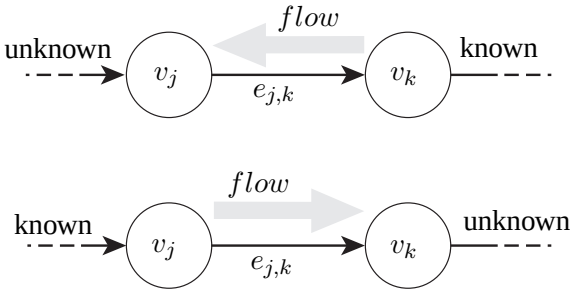


Figure 5.10: The basic operation for propagating information across an EOG. Given v_j we assume to know ω_j, $\dot{\omega}_j$, $\ddot{p}_j$. This information can then be propagated to all the connected nodes. If v_k is connected to v_j by $e_{j,k}$ (i.e. the edge is directed from v_j to v_k) then we can compute ω_k, $\dot{\omega}_k$, $\ddot{p}_k$ using (5.7) (just replace $i + 1$ with k). If v_k is connected to v_j by $e_{k,j}$ (i.e. the edge is directed from v_k to v_j) then we can compute ω_k, $\dot{\omega}_k$, $\ddot{p}_k$ using (5.7) (just replace $i - 1$ with k). Similar considerations can be done for dynamic variables.

attached to the sub-nodes, with suitable edges whose associated reference frame is $\langle s \rangle$ for both edges.

Exploiting the RNEA for EOG

The graphical representation proposed in the previous section can be used to compute the internal dynamics of a (floating) kinematic chain provided with sufficient tactile, haptic and inertial sensors. In particular, in this section we describe how to compute both kinematic and dynamic variables, associated to the edges of the graphical representation.

A first recursion on the graph (pre-order traversal) will compute the linear acceleration ($\ddot{p}$) and the angular velocity and acceleration (ω, $\dot{\omega}$) for each of the reference frames associated to the edges of the graph. This procedure practically propagates the information coming from a single inertial sensor to the entire kinematic chain. At each step, the values of ($\ddot{p}$, ω, $\dot{\omega}$) for a given link are propagated to neighbor links by exploiting the encoder measurements and a kinematic model of the chain.

A second recursion (post-order traversal) will compute all the (internal and external) wrenches acting on the chain at the reference frames associated with all the edges in the graph. In this case, Newton-Euler equations are exploited to propagate force information along the chain. At each step, all but one wrench acting on a link are assumed to be known and the remaining unknown wrench is computed exploiting a dynamic model of the link and the output from the kinematic recursion.

Kinematics

We here describe the basic equations for propagating the kinematic information within the graph. The proposed notation might seem a little bit too general, especially if compared with the classical computations where the major simplification is the assumption that kinematics are propagated in the kinematic tree along a constant path. In our case instead, we are interested in a formulation capable of exploiting multiple (dynamically inserted) inertia sensors to propagate the kinematic information from the sensors to the surrounding links. Therefore the flow of kinematics cannot be predefined but needs to be dynamically adapted to the current structure of the EOG.

The basic step here described consists in propagating the kinematic information associated to an edge connected to a node v to all the other edges connected to v. As usual, for each edge i we consider the associated reference frame $\langle i \rangle$. Referring to Fig. 5.11(a)-5.11(c) we assume that knowing the linear acceleration ($\ddot{p}_j$) and the angular velocity and acceleration (ω_j, $\dot{\omega}_j$) of the reference frame $\langle j \rangle$ we want to compute the same quantities for the frame $\langle i \rangle$ sharing with $\langle j \rangle$ a common node v. Fig. 5.11(a) represents the case where the edge i exits v but the edge j enters v; recalling the kinematic meaning of the edge directions, the sketch in Fig. 5.11(a) represents a situation where $\langle i \rangle$ is attached to v while $\langle j \rangle$ is rotated by the joint angle θ_j around z_j. The situation is exactly the one we have in the classical Denavit-Hartenberg forward

kinematic description and therefore we have [Sciavicco and Siciliano, 2005][7]:

$$\begin{aligned}\omega_i &= \omega_j + \dot{\theta}_j z_j,\\ \dot{\omega}_i &= \dot{\omega}_j + \ddot{\theta}_j z_j + \dot{\theta}_j \omega_j \times z_j,\\ \ddot{p}_i &= \ddot{p}_j + \dot{\omega}_i \times r_{j,i} + \omega_i \times \big(\omega_i \times r_{j,i}\big),\end{aligned} \tag{5.7}$$

where z_j and θ_j indicate the rotational axis and the angular position of the joint associated to the edge j. Similarly, Fig. 5.11(b) represents the case where the edge i enters v but the edge j exits the node; therefore Fig. 5.11(b) represents a situation where $\langle j \rangle$ is attached to v while $\langle i \rangle$ is rotated by the joint angle θ_i. The situation is exactly the opposite encountered in classical Denavit-Hartenberg so that we have:

$$\begin{aligned}\omega_i &= \omega_j - \dot{\theta}_i z_i,\\ \dot{\omega}_i &= \dot{\omega}_j - \ddot{\theta}_i z_i - \dot{\theta}_i \omega_j \times z_i,\\ \ddot{p}_i &= \ddot{p}_j - \dot{\omega}_j \times r_{i,j} - \omega_j \times \big(\omega_j \times r_{i,j}\big).\end{aligned} \tag{5.8}$$

Finally, Fig. 5.11(c) represents the case where both $\langle i \rangle$ and $\langle j \rangle$ are attached to the link represented by v. In this case, continuity formulas are obtained putting $\dot{\theta}_i = 0$ and $\ddot{\theta}_i = 0$ in Eq. 5.7 (or equivalently Eq. 5.8):

$$\begin{aligned}\omega_i &= \omega_j,\\ \dot{\omega}_i &= \dot{\omega}_j,\\ \ddot{p}_i &= \ddot{p}_j + \dot{\omega}_i \times r_{j,i} + \omega_i \times \big(\omega_i \times r_{j,i}\big).\end{aligned} \tag{5.9}$$

These rules can be used to propagate kinematic information across different edges connected to the same node. The only situation which cannot be solved is the one where all edges enter the node v, i.e. none of the associated reference frames is fixed to the link v. We can handle these cases *a posteriori* by defining a new arbitrary reference frame $\langle v \rangle$ attached to the link. In our formalism, this is achieved by adding a kinematic unknown ($\bigtriangledown$) and an edge from v to $\bigtriangledown$ with associated frame $\langle v \rangle$.

Remark 15. *If the edge directions are chosen according to a "regular numbering scheme" as proposed in Section 5.4.2, each edge will have a unique ingoing edge and multiple outgoing edges.*

The only nodes with no outgoing edges will be the ones corresponding to the leaves of the kinematic tree (typically the end-effectors). For these nodes, we will add a kinematic unknown ($\bigtriangledown$) and an edge from v to $\bigtriangledown$ with associated frame $\langle v \rangle$ (typically the end-effector reference frame of the classical Denavit-Hartenberg notation).

[7] In the classical recursive kinematic computation [Sciavicco and Siciliano, 2005] there is a one-to-one correspondence between links and joints (see Figure 5.6) thus resulting in a kinematic equations slightly different from Eq. 5.7. Classically, the i-th link has two joints and associated reference frames $\langle i \rangle$ and $\langle i-1 \rangle$, respectively. Only $\langle i \rangle$ is attached to the i-th link while $\langle i-1 \rangle$ is attached to the link $i-1$. The rotation between these two links is around the z-axis of $\langle i-1 \rangle$ by an angle which is denoted θ_i and therefore the analogous of Eq. 5.7 in [Sciavicco and Siciliano, 2005] refer to $\dot{\theta}_i$ in place of $\dot{\theta}_j$ and z_{i-1} in place of z_i. In our notation, we get rid of this common labeling for joints and links by explicitly distinguishing the link represented with the node v and the attached joints represented with the edges $i, j, \ldots$ and associated frames $\langle i \rangle, \langle j \rangle, \ldots$ whose axes are therefore z_i, $z_j, \ldots$ with associated angles θ_i, θ_j.

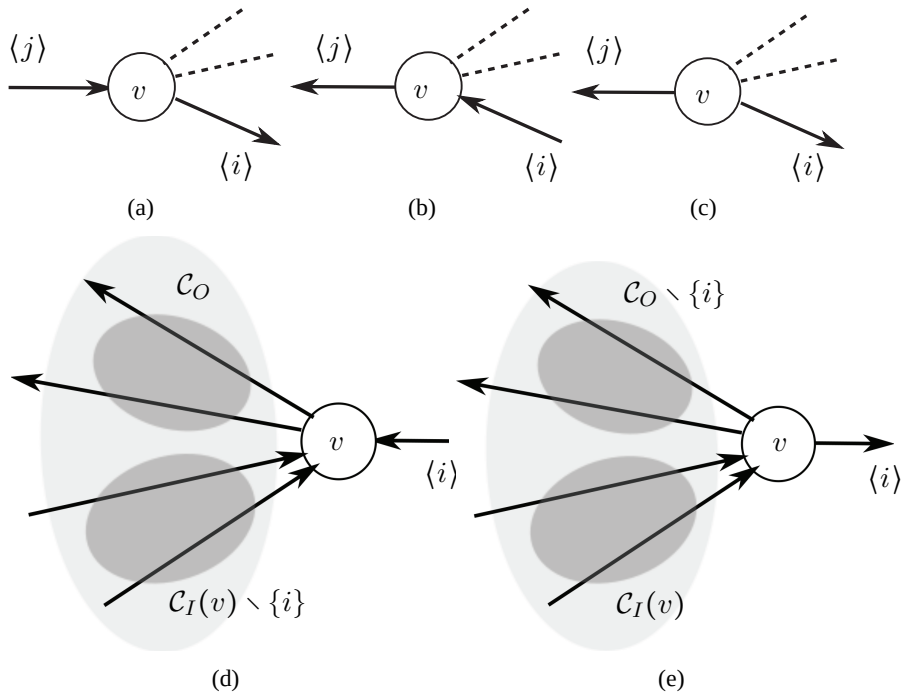


Figure 5.11: 5.11(a)-5.11(c): the three cases accounting for the exchange of kinematic information. **5.11(d)-5.11(e)**: the two cases accounting for the exchange of dynamic information.

Dynamics

We here describe the basic equations for propagating the dynamic information within the graph. Also in this case, the flow of dynamical information cannot be predefined because the graph structure continuously changes according to the position of the applied external wrenches (as detected by the distributed tactile sensor). The basic step proposed in this section assumes that all but one wrench acting on a link are known and the remaining unknown wrench is computed by using the Newton-Euler equations. Using the graph representation, a node v with all its edges represents a link with all its joints. As proposed in Section 5.4.2, at each edge $e_{u,v}$, we can associate the wrench $w_{e_{u,v}}$ that u exerts on v. At each edge $e_{v,u}$ we can associate the wrench $w_{e_{v,u}}$ that v exerts on u. The Newton-Euler equations for the link v can therefore be written as follows [Sciavicco and Siciliano, 2005]:

$$\begin{aligned} &\sum_{e_I \in \mathcal{C}_I(v)} f_{e_I} - \sum_{e_O \in \mathcal{C}_O(v)} f_{e_O} = m_v \ddot{p}_{C_v}, \\ &\sum_{e_I \in \mathcal{C}_I(v)} (\mu_{e_I} + f_{e_I} \times r_{e_I,C_v}) \\ -&\sum_{e_O \in \mathcal{C}_O(v)} (\mu_{e_O} + f_{e_O} \times r_{e_O,C_v}) = \bar{I}_i \dot{\omega}_i + \omega_i \times (\bar{I}_i \omega_i), \end{aligned} \tag{5.10}$$

where[8]:

$$\ddot{p}_{C_v} = \ddot{p}_i + \dot{\omega}_i \times r_{i,C_v} + \omega_i \times (\omega_i \times r_{i,C_v}), \tag{5.11}$$

and where $\mathcal{C}_I(v)$ is the set of ingoing edges, $\mathcal{C}_O(v)$ is the set of outgoing edges and where the index i refers to any edge in $\mathcal{C}_O(v)$ (necessarily non-empty in consideration of what we discussed in Section 5.4.2). In other terms, recalling the kinematic meaning of outgoing edges, i is an edge associated with any of the arbitrary reference frames $\langle i \rangle$ fixed with respect to the link v. As anticipated, Eq. 5.10 can be used to propagate the dynamic information across the graph. Assuming that all but one wrench acting on a link are known, the remaining unknown wrench can be computed with Eq. 5.10. Let us denote with i the edge associated with the unknown wrench. If $i \in \mathcal{C}_I(v)$, then the situation is the one represented in Fig. 5.11(d) and we have:

$$\begin{aligned} f_i &= -\sum_{\substack{e_I \in \mathcal{C}_I(v) \\ e_I \neq i}} f_{e_I} + \sum_{e_O \in \mathcal{C}_O(v)} f_{e_O} + m_v \ddot{p}_{C_v}, \\ \mu_i &= -f_i \times r_{i,C_v} - \sum_{\substack{e_I \in \mathcal{C}_I(v) \\ e_I \neq i}} (\mu_{e_I} + f_{e_I} \times r_{e_I,C_v}) \\ &\quad + \sum_{e_O \in \mathcal{C}_O(v)} (\mu_{e_O} + f_{e_O} \times r_{e_O,C_v}) + \bar{I}_i \dot{\omega}_i + \omega_i \times (\bar{I}_i \omega_i). \end{aligned} \tag{5.12}$$

[8] With slight abuse of notation we indicated with $r_{\star,C_v}$ the vector connecting the generic frame $\langle \star \rangle$ to the one placed on the center of mass C_v of the v-th link.

If $i \in \mathcal{C}_O(v)$, then the situation is the one represented in Fig. 5.11(e) and we have:

$$\begin{aligned} f_i &= \sum_{e_I \in \mathcal{C}_I(v)} f_{e_I} - \sum_{\substack{e_O \in \mathcal{C}_O(v) \\ e_O \neq i}} f_{e_O} - m_v \ddot{p}_{C_v}, \\ \mu_i &= -f_i \times r_{i,C_v} + \sum_{e_I \in \mathcal{C}_I(v)} (\mu_{e_I} + f_{e_I} \times r_{e_I,C_v}) \\ &\quad - \sum_{\substack{e_O \in \mathcal{C}_O(v) \\ e_O \neq i}} (\mu_{e_O} + f_{e_O} \times r_{e_O,C_v}) - \bar{I}_i \dot{\omega}_i - \omega_i \times (\bar{I}_i \omega_i). \end{aligned} \tag{5.13}$$

Remark 16. *With reference to Eq. 5.12-5.13, it must be noted that if only one edge is connected to the generic node v, then $\mathcal{C}_I(v) \cup \mathcal{C}_O(v) = \{i\}$. Hence, the sums $\sum f_k$, $\sum(\mu_k + f_k \times r_{k,C_v})$ (being k is the generic index for the edge) are null and the equations are basically simpler. This case is peculiar, and its significance will be clear later on when the solution of the EOG is discussed in detail.*

Building EOG for Computing Dynamics and External Wrenches

In Section 5.4.2 and 5.4.2 we presented the basic steps for propagating kinematic and dynamic information across a graph representing a kinematic tree. In this section we describe how to use these basic steps to compute the whole-body dynamics, with specific attention at getting estimates for the externally applied wrenches (denoted with $\Diamond$). During these computations the graph structure is assumed static but it might change from one computation to the next. Initially, the graph structure needs to be defined.

1. Create the graph representing the kinematic tree; define a node for each link and an edge for each joint connecting two links. The edge orientation is arbitrary and in particular it can be defined according to a "regular numbering scheme".

2. For each inertial sensor (measuring the linear acceleration and the angular velocity and acceleration) insert a *black triangle* ($\blacktriangledown$) and an edge from the node v to the triangle, where v represents the link to which the sensor is attached. Associate to the edge the reference frame $\langle s \rangle$ corresponding to the sensor frame.[9]

3. For any node v with only ingoing edges, add a *white triangle* ($\triangledown$) and an edge from v to the triangle. Associate to the edge an arbitrary reference frame $\langle v \rangle$[10].

These steps define the kinematic EOG which can be used to compute the kinematics of the entire chain. Specifically, if this graph contains a single inertial sensor (represented by a $\blacktriangledown$ node), the associated measurements can be used to compute the linear acceleration and angular acceleration and velocity for all the edges of the graph. Computations can be performed following the procedure in Algorithm 3, that is a *pre-order*[11] traversal of the tree with elemen-

[9] Kinematic chains are often grounded and therefore there exists a base link with null angular kinematics, $\omega = [0,0,0]^\top$, $\dot{\omega} = [0,0,0]^\top$ and gravitational linear acceleration $\ddot{p} = g$, being g the vector representing the gravity force. This situation is mathematically equivalent to an inertial sensor attached to the base link and measuring constantly $\omega = 0$, $\dot{\omega} = 0$ and $\ddot{p} = g$.

[10] See also Remark 15.

[11] *pre-* and *post-order* refer to different classical graph visiting algorithms [Cormen et al., 2002].

tary operations defined by Eq. 5.7, Eq. 5.8 or Eq. 5.9. If multiple ▼nodes (i.e. inertial sensors) are present in the graph, each path between two of these nodes corresponds to a set of three equations containing the measurements: one for the linear accelerations, one for the angular velocity and one for the angular accelerations. These equations can be used to refine the sensor measurements or to give better estimates of the joint velocities and accelerations (typically derived numerically from the encoders and therefore often noisy).

Remark 17. *In this respect, a possible algorithm for computing the better estimate of the kinematics, given the multiple sources, is briefly reported in Algorithm 4. Basically, given a set of K kinematics sources ▼, which for brevity we name $\kappa_1, \ldots, \kappa_K$, Algorithm 3 is solved K times. At each time k, κ_k is the only kinematic source which is not being removed from the EOG, and then the only ▼in the graph. The solution of the EOG K times yields a set of conditional estimates $\omega_{j|\kappa_1}, \ldots, \omega_{j|\kappa_K}$, $\forall j$ (analogous considerations hold for $\dot{\omega}$ and $\ddot{p}$), which can be used by classical filters to provide the better estimate (e.g. maximum likelihood filters, Kalman filters etc).*

A clarifying example is shown in Figure 5.12(a): notice that the visit order is not related to the edge direction, since the latter only affects the recursive equations that must be used to propagate the variables, as shown in Figure 5.10. Once velocities and accelerations have been computed for all edges, a new series of steps needs to be performed on the EOG to obtain the dynamic enhanced subgraphs.

4. For each FTS embedded in the link v cut the graph into two subgraphs according to the procedure Figure 5.9. Divide v into two nodes v_B and v_F representing the sub-links (with suitable dynamic properties); define two *black rhombi* (⧫) and add two edges from the rhombi to the nodes. Associate to both the edges the same reference frame $\langle s \rangle$ corresponding to the sensor frame.

5. If there are other known wrenches acting on a link (e.g. sensors attached at the end-effector), insert a *black rhombus* (⧫) and an edge from the rhombus to v, where v represents the link to which the wrench is applied. Associate to the edge the reference frame $\langle s \rangle$ corresponding point where the external wrench is applied.

6. If the distributed tactile sensor is detecting externally applied wrenches, insert a *white rhombus* (◊) for each externally applied unknown wrench. Add an edge connecting the rhombus with v, where v represents the link to which the wrench is applied. The edge orientation is arbitrary depending on the wrench to be computed (i.e. the wrench from the link to the external environment or the equal and opposite wrench from the environment to the link). Associate to the edge the reference frame $\langle c \rangle$ corresponding to the location where the external wrench is applied.

After these steps have been performed, we basically obtained the dynamic enhanced subgraphs, each of which can be considered independently. Wrenches can be propagated to the unknown nodes (◊) if and only if a unique unknown for each sub-graph exists. If this is the case, then for each unknown we can define a tree with the node ◊as root. Wrenches can be

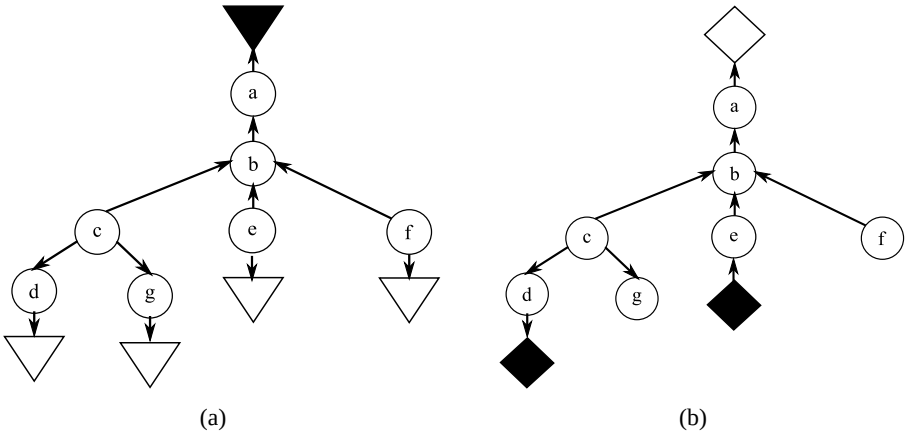


Figure 5.12: 5.12(a):An example of kinematic EOG with multiple branches. Starting from the root ▼, the propagation of kinematics information follows the pre-order traversal of the tree. Thus, the order of "visiting" nodes is: a, b, c, d, g, e, f. **5.12(b)**: an example of dynamic EOG with multiple branches. The propagation of dynamics information follows the post-order traversal of the tree: starting from leaves, information is propagated from children to parents, until the root ◊. Thus, the order of "visiting" nodes is: d, g, c, e, f, b, a. It must be noted that leaves are not necessarily ⧫, as explained in Remark 16.

propagated from the leaves to the root following the procedure in Algorithm 5, which is basically a *post-order* traversal of a tree [Cormen et al., 2002] with elementary operations defined by Eq. 5.12 or Eq. 5.13. If there is no ◊node in a subgraph (i.e. no external forces are acting on the subgraph), then the *post-order* traversal of this graph produces two equations (one for forces and the other for wrenches) with no unknowns[12]. These equations can be used to estimate on-line the dynamical parameters of the corresponding kinematic sub-tree exploiting the linearity of these parameters in the equations [Sciavicco and Siciliano, 2005].

Remarkably, in the considered cases (one ◊per subgraph at maximum) each edge in the subgraph is visited during the *post-order* traversal. As a result, all internal wrenches are computed and therefore a complete characterization of the whole-body dynamics is retrieved.

As a consequence of what has been shown, given N FTS distributed on a chain, $N+1$ subgraphs are produced and therefore a maximum of $N+1$ external wrenches can be estimated (one for each sub-graph).

Case Studies

In order to clarify how to exploit computation of wrenches on an EOG, different situation are hereafter reported.

Once again, the reader should note that the employment of the Denavit-Hartenberg notation

[12]Practically, these equations can be obtained by defining an arbitrary ◊connected to an arbitrary node. A *post-order* traversal of the graph with ◊as root determines the equations by simply assuming that the wrench associated to the edge connected to ◊is null.

Algorithm 3 Solution of kinematic EOG exploiting a tree

Require: EOG, $\omega_0, \dot{\omega}_0, \ddot{p}_0$
Ensure: $\omega_i, \dot{\omega}_i, \ddot{p}_i, \forall v_i$
1: Attach a node ▼ for every kinematic source (e.g. inertial sensor)
2: Set $\omega_0, \dot{\omega}_0, \ddot{p}_0$ in ▼
3: Re-arrange the graph with a ▼ as the root of a tree
4: *KinVisit*(EOG,v_{root})

KinVisit(EOG,v_i)
1: Compute $\omega_i, \dot{\omega}_i, \ddot{p}_i$ with Eq. 5.7 or 5.8 or 5.9 according to direction of the edges i, j connected to v
2: **for each** child v_k of v_i **do**
3: *KinVisit*(EOG,v_k)
4: **end for**

Algorithm 4 Fusion of multiple kinematic sources

Require: EOG, $\kappa_k = [\omega_k, \dot{\omega}_k, \ddot{p}_k]$, $k = 1, \ldots, K$
Ensure: $\hat{\omega}_i$, $\hat{\dot{\omega}}_i$, $\hat{\ddot{p}}_i$, $\forall i$
1: **for each** $k = 1 : K$ **do**
2: Attach a node ▼ for κ_k
3: Compute $\omega_{i|\kappa_k}, \forall i$
4: **end for**
5: Compute $\hat{\omega}_i$ = filter* $(\omega_{i|\kappa_1}, \ldots, \omega_{i|\kappa_K})$

* filter is a generic filter for data fusion from multiple sensors

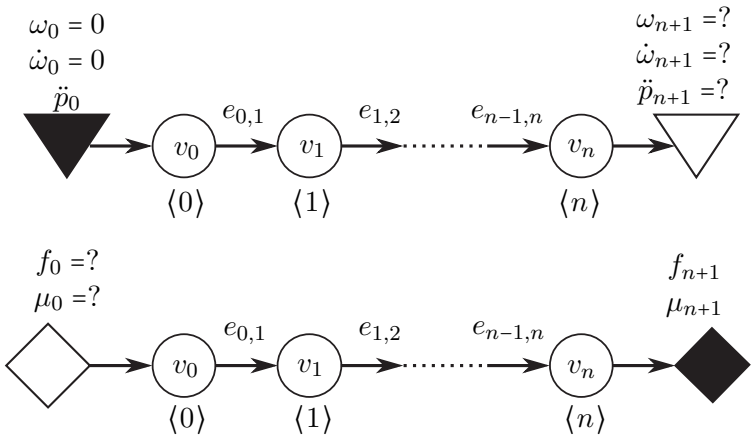


Figure 5.13: A representation of classical Newton-Euler computations for the graph in Fig. 5.7. Kinematics is assumed known at the base (▼). Wrenches are assumed known at the end-effector (⧫, e.g. if an external FTS is used) and propagated to the base.

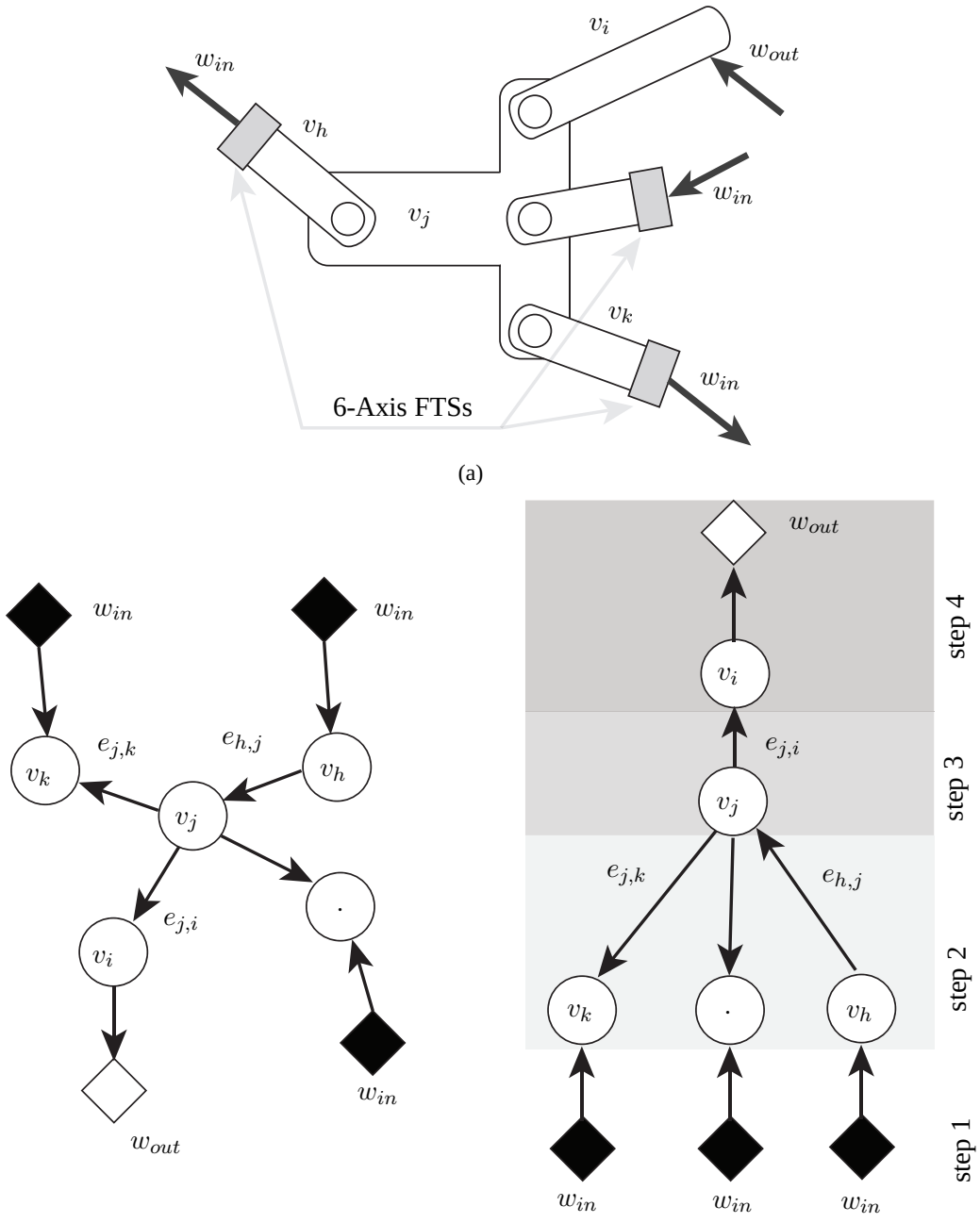


Figure 5.14: **5.14(a)**: an example of link with multiple connections, which represents a typical case of multi-branched tree. At the extremity of the links, except for one, FTS are located. The situation can be represented with an EOG. **5.14(b)**: an EOG representing a situation similar to the one of 5.14(a). On the left side, the sketch of the graph; on the right side, the figure shows the order in which the graph will be visited for computation.

Algorithm 5 Solution of dynamic EOG exploiting a tree

Require: EOG, $w_s \forall$ FTS
Ensure: w_i, $\forall v_i$
1: For every FTS, attach a node ⧫ to the corresponding link
2: Set w_s in each ⧫
3: For each ⧫, split the graph and create two sub-graphs (see text for details)
4: Attach a node ◊ to each link where a contact is detected: if there is no contact in a subgraph, choose an arbitrary position and attach a fictitious ◊[13]
5: Re-arrange each sub-graph with a ◊ as the root of a tree
6: **for each** subgraph **do**
7: *DynVisit*(EOG,v_{root})
8: **end for**

DynVisit(EOG,v)
1: **if** v has children **then**
2: **for each** child $e_{v,h} \in \mathcal{C}(v), e_{v,h} \neq i$ **do**
3: $w_{e_{v,h}}$ = *DynVisit*(EOG,h)
4: **end for**
5: **end if**
6: Compute w_i with Eq. 5.12 or Eq. 5.13 according to the direction of the edges

for the definition of the kinematic structure of the links, and the RNEA for the definition of the dynamics of the system, are not mandatory. Custom choices can be adopted.

When passing from one vertex to the other, Eq. 5.12 or Eq. 5.13 is used. This equation provides the computation of both internal and external forces, depending on the definition of known and unknown variables on the graph. With reference to Figure 5.10, when the flow of the information is along the same direction of the edge, f_i of Eq. 5.12 or Eq. 5.13 is to be computed. One of the forces among the $\sum f_k$ otherwise, depending on which of the k links the unknown is located.

- **Single-Branched Open Chain**

  In the case presented in Figure 5.13 for a single open chain, there exists, for the links in between the base link and the final link, one single f_k, since the maximum value of k depends on the number of links attached to the i-th. When the flow of the information is along the same direction of the edge, f_i of Eq. 5.12 or Eq. 5.13 is to be computed; f_k otherwise. Next sections show some cases which demonstrate the generality of the EOG method also for open, multi-branched kinematic chains. With respect to Figure 5.13, we point out that the unknown ◊ attached to the base is used if a contact is detected on that link (e.g. if the artificial tactile skin reveals a contact at the base). In absence of contact, the node ◊ is no longer needed. In this case, it is possible to write the recursive equations as a compact set, where all the dynamic variables are known: this formulation can be exploited to obtain, for example, a better estimate of the rigid-body model parameters,

e.g. links mass.

- **Multiple-Branched Nodes and External Forces**

  External forces may be acting in other locations different from the end-effector (e.g. on an internal link in between the base and the end-effector), as a consequence of contacts with the environment. In such cases, the application point (or the centroid of the contact region) must be known. Also in this case, Eq. 5.12 or Eq. 5.13 holds. Note that one external force can be determined if, and only if, all the other wrenches flowing through the edges connected to the link can be determined.

  Consider the general example of one link connected to N other links, $N \geq 2$. The situation is represented in Figure 5.14(b). The graph associated to a similar situation instead is shown in Figure 5.14(a), left side. The right side of Figure 5.14(a) shows the steps the algorithm performs to determine the unknown wrench w_{out} acting on link i, when a direct measurement of that wrench is missing.

  The first step consists in setting the unknown wrenches given the quantities that have flown from the known leaves. These quantities can in general be measured by FTS within a link, or set *a priori* (e.g. with the assumption that link k is moving without interaction, this wrench can be set equal to zero). In the second step, each of the links connected to v_j performs the calculation (using Eq. 5.12 or Eq. 5.13) necessary to define the information passing through the edge which connects to link j, according to the direction of the edge. Moreover, vertex j preforms again the evaluation of the force transmitted to i, again from Eq. 5.12 or Eq. 5.13, according with the direction of the edge $e_{j,i}$. Finally, v_i evaluates w_{out}. Note that in this example, the assumption that w_{out} is the only unknown must hold.

- **Virtual Joint Torque Sensors** In case the Denavit-Hartenberg notation has been employed for the definition of the kinematic of the structure, an estimation of the joint torque can be performed, once the i-th wrench is known:

$$\tau_i = \mu_i^\top z_{i-1} \tag{5.14}$$

  where z_{i-1} is the z-axis of the reference frame $\langle i-1 \rangle$ as in Fig. 5.6. The method shows that it is possible to have an estimation of joint torques, which can be used, successively, for joint torque control. Moreover, this is not the only information that it is possible to extract from the method. Joint torques are here found as one component of the wrenches flowing through the edges. These wrenches allow having a better representation of the possible contact situation, which can be used as a virtual measurement, to perform every kind of tasks involving force detection and control. It is necessary to note that the more 6-axis FTS are employed, the more accurate will be the estimation.

5.5 Experimental results

In this following experimental results on the humanoid robotic platforms are illustrated.

$q_0[^\circ]$	$q_1[^\circ]$	$q_2[^\circ]$	$q_3[^\circ]$	
-10	-140	-115	0	max
150	100	15	100	min

Table 5.1: Value ranges of the James' arm joint positions.

5.5.1 Closed loop motion planning with joint velocity control in James

In Section 4.4 a method to find the neural controls that make a robot move optimally with respect to a specific criterion was presented. Numerical results were shown in Section 4.5 for a two DOF arm and a three DOF mobile robot, where theoretical and practical issues were also discussed.

Here, the method is applied to the control of James' arm, focusing on the first four joints, i.e. three joints of the shoulder and one of the elbow. The range of the four joints is reported in Table 5.1. James hand is considered as the end effector of the manipulator. At this stage, the orientation of the hand is neglected.

Denoting with $x^r = [x, y, z]$ the Cartesian coordinates of the end effector with respect to a fixed reference frame, and with $q = [q_0, q_1, q_2, q_3]$ the vector of the joint position variables of the arm (see Table 5.1), then the forward kinematics $x^r(t) = f_{\text{arm}}(q(t)),\ f_{\text{arm}} : \mathbb{R}^4 \rightarrow \mathbb{R}^3$ is found using the Denavit-Hartenberg convention [Sciavicco and Siciliano, 2005]. The target's Cartesian coordinates are denoted by $x^g(t)$. The orientation of the end effector is neglected. Within this context, the four DOFs manipulator is redundant.

Different tasks are shown: the focus is both on the response of the CLIK regulator when tracking a time varying desired trajectory, and on the performance of both FH and RH neural controllers, used to plan Cartesian trajectories for reaching and tracking tasks. In this specific case, the regularizing parameters of the Jacobian matrix in Eq. 5.3 were set to $\bar{\sigma} = 0.20$ and $\bar{k} = 0.10$: these values were chosen by driving the arm to singular positions, so as to set a safety threshold. The variable delay in the communication process (from PC to DSP boards via PC104 and back) affects the global performance of the control loop. Delay is actually variable and depends on multiple terms: delay on the CAN bus (ranging from 2 to 4ms, approximately), the delay of the DSP in elaborating the commands from the PC, computing the low-level control trajectories, sending the commands to the motors (≈ 1ms), and the delay in the backward communication (i.e. retrieval of joint positions and velocities from the DSP, from 2 to 5 ms, approximately). To avoid a stochastic modeling of delays, they were treated as a high frequency pole. At first, the simple proportional regulator $R(s) = K_e$ was tested. Unfortunately, high-frequency dynamics prevented to raise K_e to a desired value: in the experiments, $K_e = 20I$ was already revealing elastic effects (a step-response was used to investigate the system); higher, e.g. $K_e = 40I$, was compromising the safety of the robot (high oscillating velocities exalt the elasticity effect of tendons transmission). For these reasons the simple proportional gain K_e was substituted with Eq. 5.4, so as to have a high proportional gain while not incurring in instability of the system. Different settings concerning sampling time, time constant and gain were investigated (in particular varying $\Delta t = 5 \div 50ms$), and reasonable performances were obtained with $\Delta t = 20ms$, $\tau_e = 30, K_e = 0.5$ It must be remarked that the commanded velocities $v(t)$

Figure 5.15: Recording a human handwriting trajectory with the Vicon system. A marker has been placed on the subject finger: cameras recorded the kinematics of the marker while the subject performed the motion.

are checked by a further lower level control: to prevent the robot to make fast movements, joint velocities $q^*(t)$ are saturated in the range $[-25, +25](^\circ/s)$. This rough solution is necessary to avoid the stress of the elastic parts of the robot (mainly tendons) and their consequent damage.

Hand-writing: human vs. robot

In Figure 5.16 a hand-writing task is presented, where the end-effector (located in the wrist) performs a time varying 3D movement, precisely James's signature. The signature consists of a sequence of Cartesian coordinates which were measured with a VICON motion capture system [Vicon, www] during a human writing task. Therefore, the time of the global trajectory and its dimensions correspond to a smooth human movement. Only position references can be retrieved from the Vicon, thus feed-forward commands $\dot{x}^*$ were not used in the CLIK algorithm. Cartesian position and velocities of the end-effector are shown in Figure 5.17(a), while the corresponding arm's joints position and velocities are shown in Figure 5.17(b). The purpose of this experiment was mainly to tune the performances of the regulator of Eq. 5.4, since the transient response was fundamental in order to have a fluid tracking of the desired trajectory. It must be noted that joint velocities are noisy, because encoders can measure only the motor positions (though accurately), so the joint velocities are computed via numerical differentiation on the DSP boards, which cause a noisy estimation as shown in Figure 5.17(b).

"Human-like" motions

In [Ivaldi et al., 2008a] a bell-shaped motion on a planar surface was planned for a 2DOF arm, where the cost function was a *minimum jerk* one, and the movement time T was fixed. Here, a 4DOF manipulator performs multiple reaching movements, where the desired point to reach changes at each time, unpredictably, and the motion criteria follows the same principle. Thus, a MJM is implemented (see Section 3.2.1). The time to accomplish a movement is roughly determined at each trial by Fitts' law: with reference to Eq. 3.2 in the experiment, the following parameters were used: $a = 1.5, b = 0.7, c = 0.5, W = 0.0025$ (the latter meaning that the target is $2.5mm$ wide). Parameter D was computed at each trial, being the Euclidean distance between the initial position of the end effector and target position in the Cartesian space. In Figure 5.18

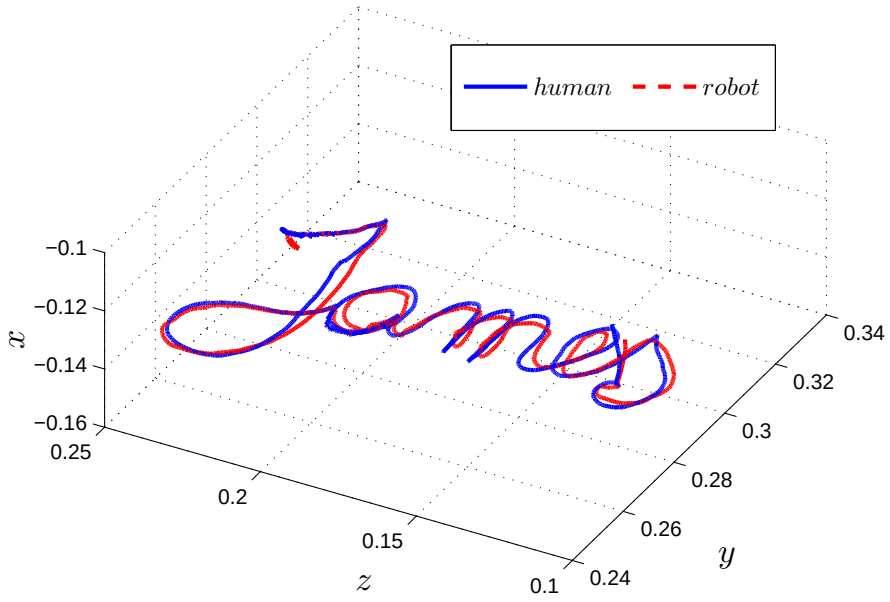


Figure 5.16: The human (blue) and James's signature (red).

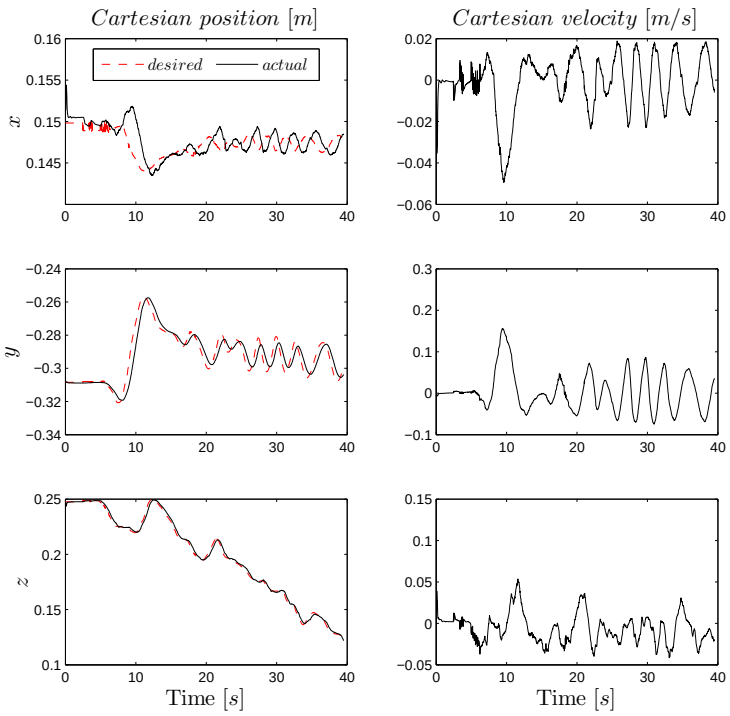
multiple subsequent reaching trajectories are shown, where the duration of each was chosen according to (3.2). Figure 5.19(a) focuses on two among the aforementioned movements: the Cartesian position and velocities are actually shaped as minimum jerk trajectories. In the second movement, an "error" is evident: looking at the joint level, in Figure 5.19(b), it must be noted that the desired velocity in Joint 3 exceeds the safety threshold (here $15\ deg/s$, then the real, executed trajectory is slightly different from the desired minimum-jerk one.

Tracking a target moving unpredictably

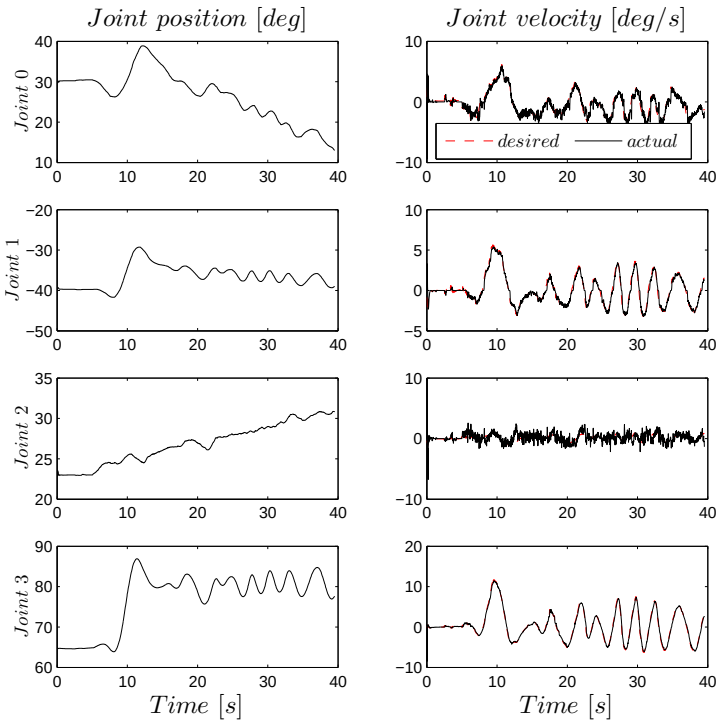
In this experiment, a RH neural controller as described by Problem 13 is used to plan Cartesian trajectories for reaching. The desired trajectory is characterized by the following convex cost function:

$$\mathcal{J} = \sum_{i=0}^{N-1} c(u_i) + \xi_{i+1}^{\top} V_{i+1} \xi_{i+1}$$

where $\xi = [\Delta x, \Delta \dot{x}, \Delta y, \Delta \dot{y}, \Delta z, \Delta \dot{z}]$, i.e. the difference between the target and the end-effector Cartesian positions and velocities, $u = [\ddot{x}, \ddot{y}, \ddot{z}] \triangleq [u^x, u^y, u^z]$. Note that $\mathcal{J}$ again represents a tradeoff between the minimization of the energy consumption (acceleration and velocity can be considered as proportional to the power consumption) and the "best" end-effector proximity to the target during and at the end of the maneuver. The second term is indeed related to the distance to the target during the maneuver, which results in high velocity desired trajectories. Conversely, the first term acts as a damping quantity, which is necessary



(a)



(b)

Figure 5.17: Comparison between human and robot handwriting. **5.17(a)** Cartesian position and velocity of the end-effector during the signature task. Red dashed lines are the desired reference (i.e. the human recorded trajectory), while black lines are the measured trajectories of the robot hand. **5.17(b)** Joint positions and velocities of the robot arm during the signature task. Red dashed lines are the desired velocities, while black lines are the effective joint measured values.

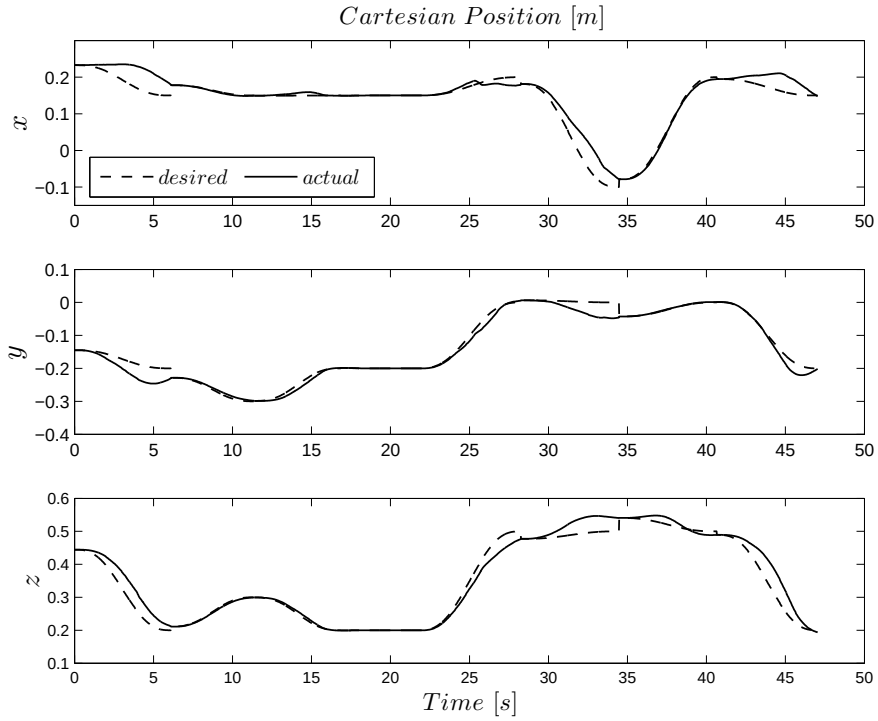
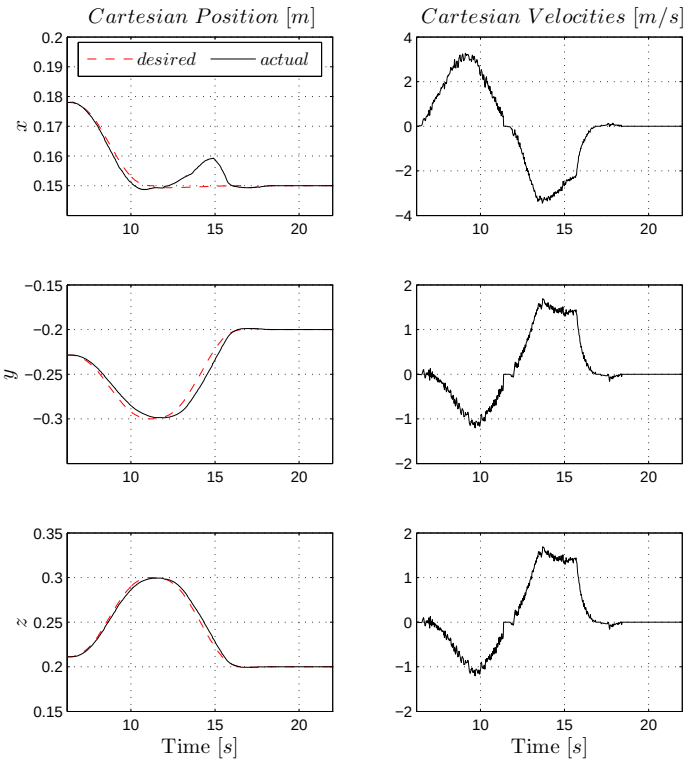
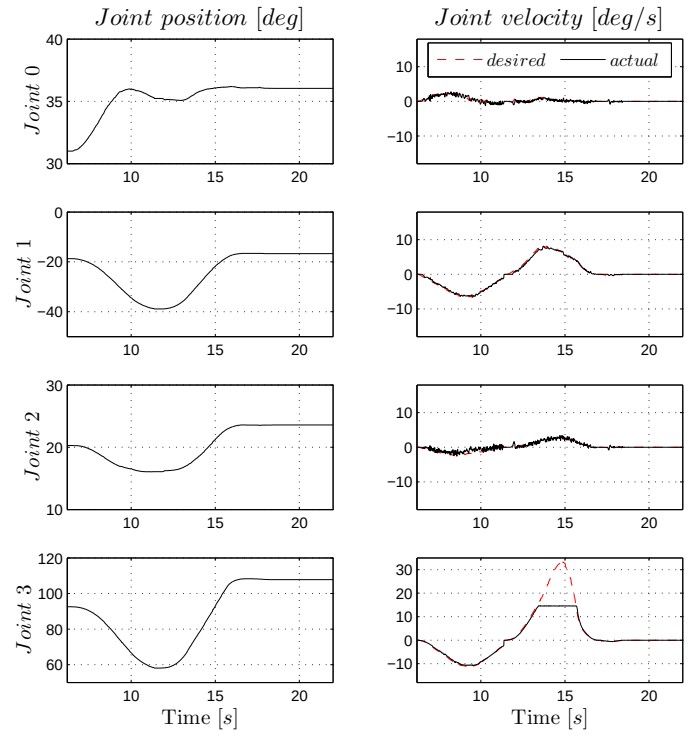


Figure 5.18: Cartesian positions of the end-effector during multiple reaching tasks. The duration of each single point-to-point movement is determined by Eq. 3.2, while the trajectories have a minimum jerk shape. Failures in reaching (i.e. inability to reach the target perfectly) are due to joint velocities limitations (for safety reasons) which obviously compromise the perfect execution of the trajectories.



(a)



(b)

Figure 5.19: A trial among the multiple reaching tasks. **5.19(a)** Cartesian positions and velocities of the end-effector during a single reaching movement. The trajectories are correctly bell-shaped. **5.19(b)** Joint positions and velocities. Note that the joint velocities are saturated within the range $[-15, 15]$.

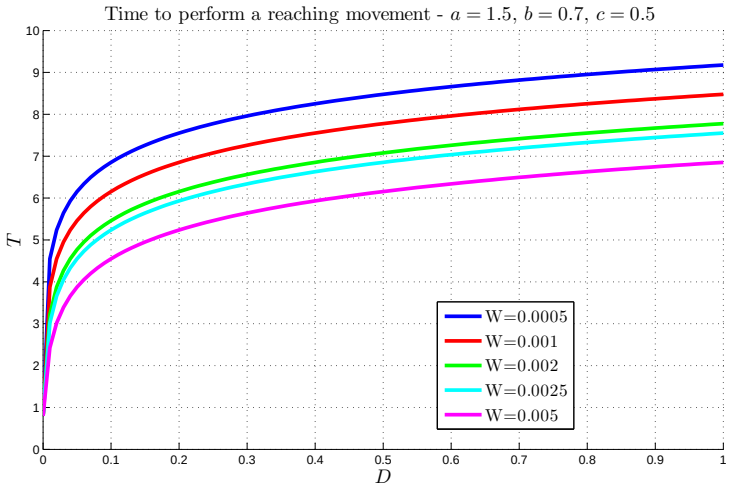


Figure 5.20: Fitts' law: the time to accomplish a movement T function of the amplitude of movement D.

to reduce the risk of damage of the platform.

Remark 18. *$c(u_i)$ reminds the absolute work term [Berret et al., 2008], measuring the energy expenditure of a movement. In this case the goal is not to minimize an absolute work term (which would require the torques and thus a more complex dynamic model), however it is interesting to show that the proposed method can be applied to the implementation of "complex" nonlinear costs, which are usually superseded due to their mathematical difficulty (e.g. induced by the absolute value).*

Weight matrices were set to $V_i = \mathrm{diag}(1.0, 80.0, 1.0, 80.0, 5.0, 10.0)$, $i = 1, \ldots, N-1$, $V_N = 40I$. $c(u_i)$ is a numerical approximation of $\|u_i\|^2$. Output velocities were bounded within a safe range. The state function f (4.57) describing the system is a double time integrator of the controlled accelerations u_i. The double integrator approximates ideally the robot and the CLIK controller; the latter takes entirely account of singularities and redundancies, and is properly tuned to achieve the desired behaviors. Thus, the overall system performances do not degrade.

The training of the neural networks chain was performed off-line with the following parameters: $N = 60, \nu = 40$ (where N represents is the number of control steps, and ν is the number of total neurons of the net), $\varphi = \tanh$. More than 10^7 random samples were used to feed the networks, considering the whole reachable Cartesian space for the robot, and an augmented space for the target (to consider also unreachable targets); velocities and position were uniformly sampled.

Remark 19. *The considerable amount of training data is required as the control problem is stated in a stochastic framework: each possible configuration (in terms of position, velocity and acceleration) of the four joints of the robot, producing a certain end-effector position and velocity in the Cartesian space, as well as the target's coordinates in the space, must be considered. It is remarked that the control problem does not take into account any prior*

knowledge of the target behavior: there's no prediction on its evolution, based neither on the past nor on some a priori information.

Remark 20. *The flops count for the on-line computation of the approximated optimal controls is about* 4633*, which in a Pentium IV 3GHz are approximately* $10\mu s$*. More specifically, the operations to make a forward "step" of a NN are* $\nu(n+m)$ *products,* $\nu(n+m+1)+m+2(\nu+m)$ *sums,* $2(\nu+m)$ *exponentials and* $\nu+m$ *divisions.*

In Figure 5.22 the task is presented: the target is "fixed" (i.e. still), but changes unpredictably its position after a variable unknown period of time. This situation is representative of the case where the attentive system of the robot selects a target to be reached in the space (e.g. when the robot recognizes a known object of interest). Whenever a new interesting object is presented to the robot, it is selected as the new target[14]. Cartesian trajectories of the end-effector and target position (named "desired") are shown in Figure 5.21(a), while the corresponding arm's joint position and velocities, computed by the CLIK, are shown in Figure 5.21(b). The saturation of velocities is evident in the joint velocity profiles.

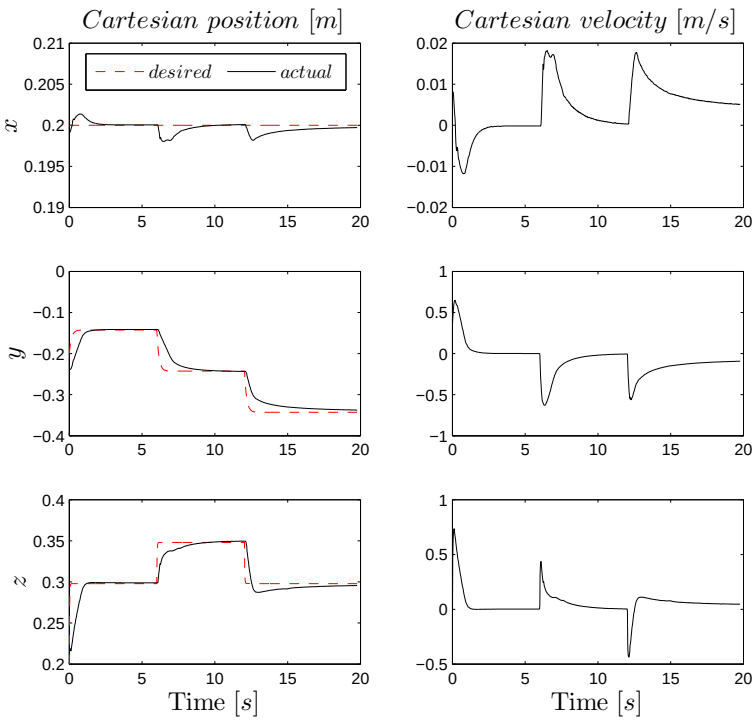
In Figure 5.23 instead, a target moving in the 3D space along a circle path is followed by the end-effector. This situation is comparable to the case when the robot wants to catch a moving object (e.g. the human engage the robot in a "catch it if you can" game, where he moves the object randomly in the space so as not to make the robot reach for the object). Even if the robot does not have any information on the target's kinematics or dynamics, and does not model or attempt to predict the behavior of the target and someway anticipate it, the robot is able to track almost perfectly the desired trajectory. Figure 5.24(a) and 5.24(b) show the Cartesian coordinates of the wrist and the joint position and velocities of the arm during the movement.

5.5.2 Estimation of intrinsic and extrinsic wrenches in iCub

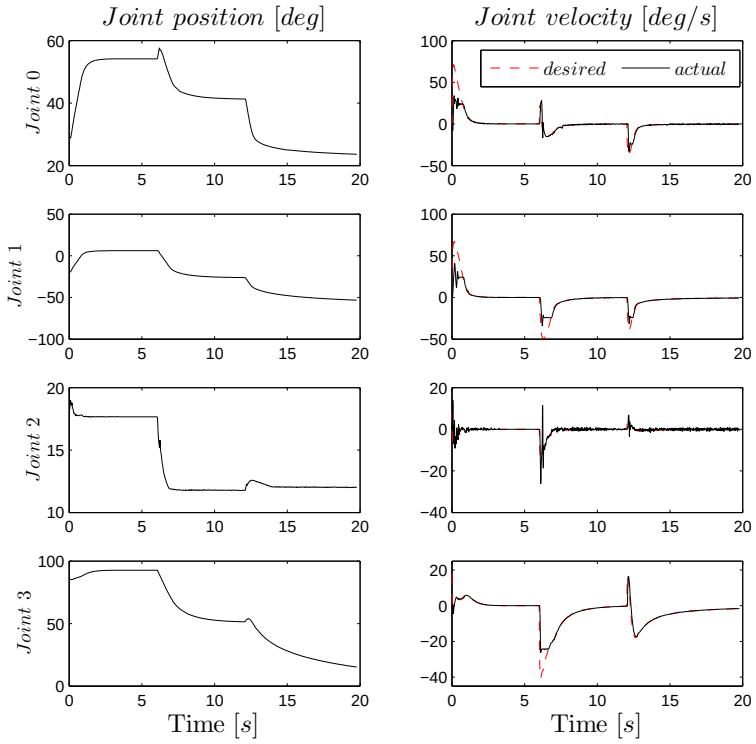
Theoretical results discussed in Section 5.4.2 have been implemented in iDyn, a library for dynamics of single and multiple-branched serial-links kinematic chains [Ivaldi et al., www]. iDyn is built on top of iKin [Pattacini, www, Pattacini et al., 2010], a library for forward-inverse kinematics of serial-links chains of revolute joints with standard Denavit-Hartenberg notation. Both libraries are part of an open source software project, released under a GPL license. Though being tailored for the iCub, remarkably iKin and iDyn are generic, cross-platform and portable C++ libraries (relying on CMake and YARP middleware [Fitzpatrick et al., 2008, Fitzpatrick et al., 2010]) that can be used to study kinematics and dynamics of potentially any robotic device.

Figure 5.25 illustrates a scheme of the global control system. Sensor measures, acquired through local boards, are sent through CAN bus or COM port to the PC104, being the local CPU on the robot (located on its head). An interface module running on the PC104 replicates collected measures to a local Gigabit Ethernet network, exploiting YARP middleware [Fitzpatrick et al., 2008]. In one PC of the iCub cluster, the so called "whole body dynamics"

[14]This situation must not be underestimated, because the evolution of the reference variable ξ^* has sudden changes, steps in the position and impulses in the velocity level.



(a)



(b)

Figure 5.21: Multiple reaching with a RH neural controller. **5.21(a)** Cartesian positions and velocities of the end-effector during the movements. It is worth noting the different velocity profiles (e.g. $\dot{y}, \dot{z}$) due to the different parameters of the cost function. **5.21(b)** Position and velocity of the arm's joints during the movements.

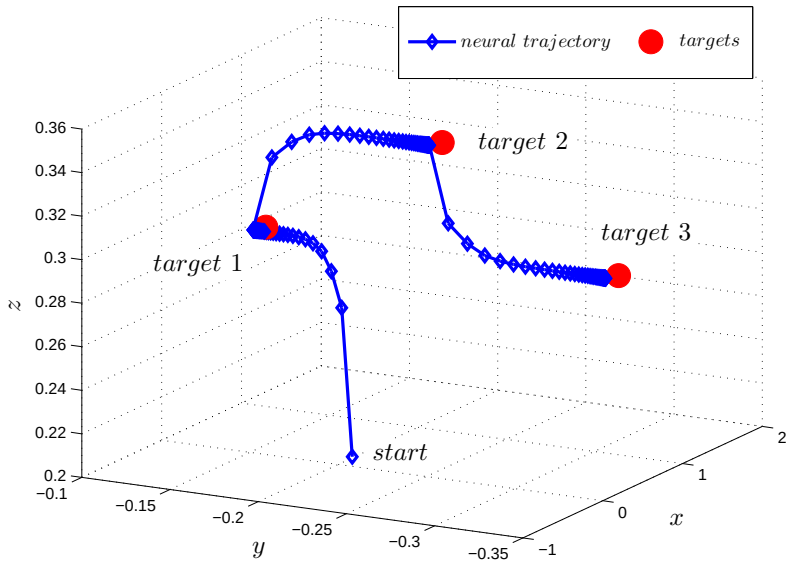


Figure 5.22: "Neural" trajectory (blue) of the end-effector, reaching a target (red) which changes unpredictably. The shape of each trajectory is determined by the cost function $\mathcal{J}$ and its parameters.

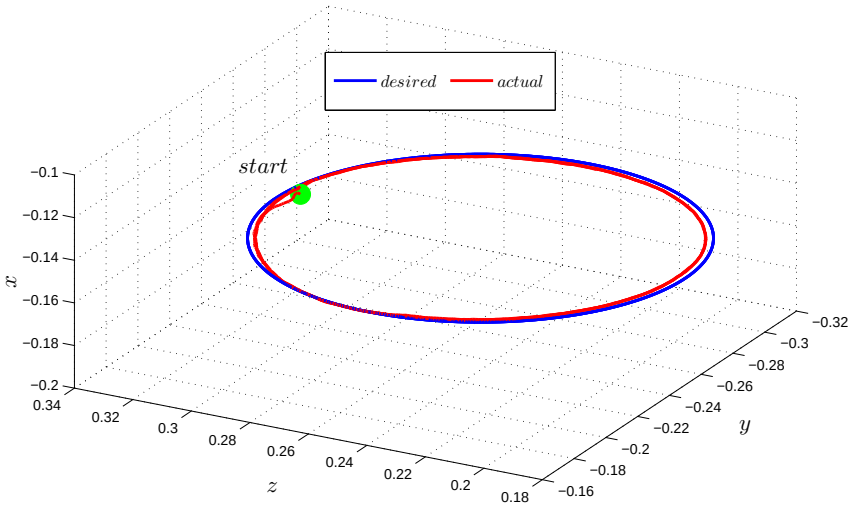
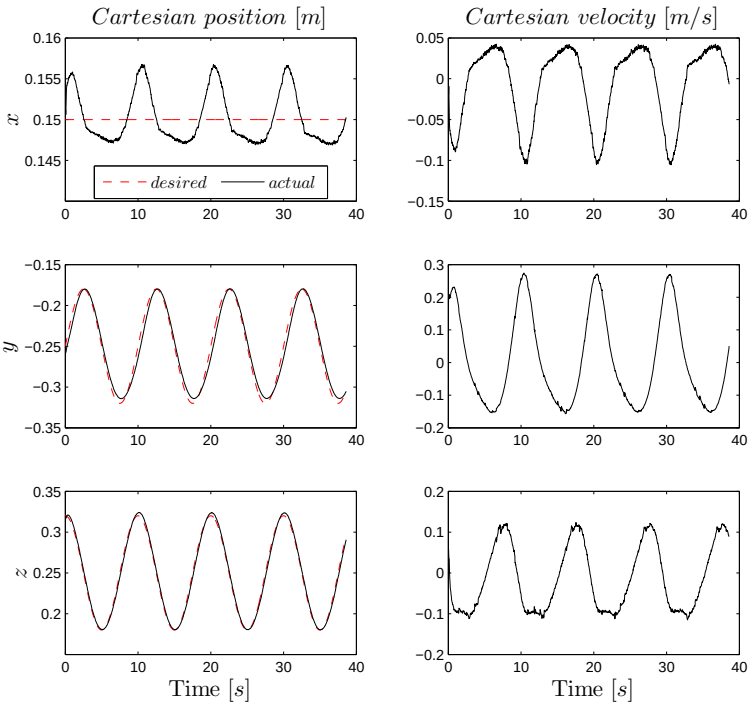
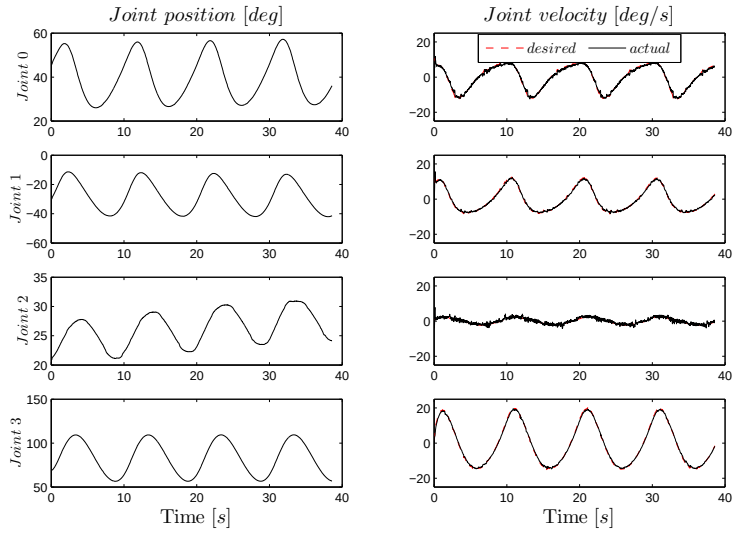


Figure 5.23: Trajectory of the end-effector (blue) when tracking a target (red) moving along a circle. The starting position of the end-effector is represented by the green sphere.



(a)



(b)

Figure 5.24: Tracking a point moving along a circle path. **5.24(a)** Cartesian positions and velocities of the end-effector. **5.24(b)** Position and velocity of the arm's joints during the movement.

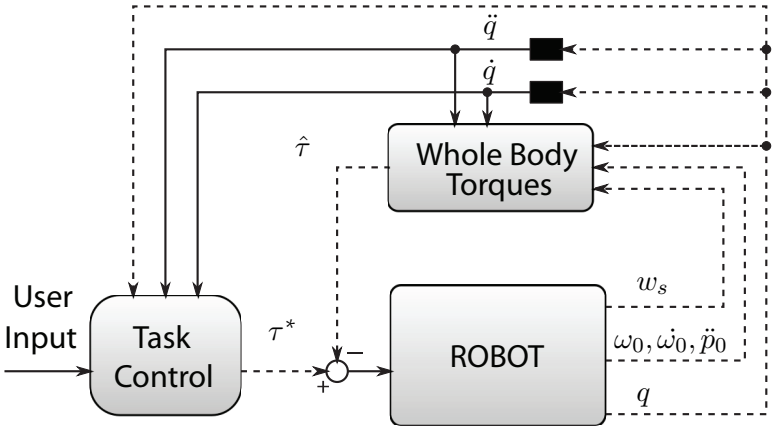


Figure 5.25: A schematic structure of the control system. From the robot, it is possible to retrieve the joints positions q, the inertial measurements $\omega_0, \dot{\omega}_0, \ddot{p}_0$ (▼) and the FTS measures w_s (⧫). Joints velocities and accelerations are estimated through specific filters (black rectangles on the top). Dashed arrows indicates information sent through YARP ports on the local network, thus subject to delays. More details on this scheme can be found in the online documentation of the iCub: `http://eris.liralab.it/wiki/Force_Control`.

is computed, i.e. the whole body joints torques are computed through iDyn. Joint torques estimated by the latter process are then sent back to the boards on the robot via PC104, where local force/torque control strategies can be finally applied. It must be noticed that while the computation of the torques (i.e. the time required by iDyn to compute the whole body torques, precisely to solve the kinematic and dynamic EOG) takes on average $3.2ms$, the time elapsed between the received measurements and the outcome of the computation is about $5.3ms$. To this time, a variable delay must be added, due to the stochastic behavior of the network, which is necessary to carry information among different PCs.

Remark 21. *The weak point of the global process is the round-trip-time of a sensor measure: practically, a delay is introduced whenever information is sent through the local network, thus between the sensor measure and the application of a suitable control strategy exploiting that measure a certain time elapses. This delay, which is varying between* 4 *and* $10ms$ *on average, is the weak point of the active control strategy as it is implemented on the platform, since it constrains the control bandwidth. We are currently investigating the possibility to modify the platform, in particular the embedded electronics, in order to support direct joint torque control, but in a totally different way, to improve control performances for certain critical joints. However, for the current configuration of the robot, the proposed approach is apparently the only possibility to provide force/torque control.*

Given the sensors position and the description of the robot kinematics, it is quite easy to build the kinematic and dynamic EOG.

The structure of the EOG needed for the computation of the robot kinematic variables for all joints is shown in Figure 5.26(a)[15]: the inertial sensor is the unique absolute source of kine-

[15]Each vertex is named as $X - k$, where $X = \{H, LA, RA, RL, LL, T\}$ is a code for the limb (head, torso,

matic information (▼) (encoders are relative sources, and their information is considered as a property of the links); unknowns (▽) have been placed at the end-effectors, so that kinematics variables are propagated through all the graph nodes.

Since the complete knowledge of the kinematic information is a prerequisite for the computation of the dynamics, the kinematic EOG shown in Figure 5.26(a) is adequate for all applications. The dynamics EOG is instead task-dependent.

As the iCub is provided with a set of four FTS, the dynamic EOG is divided in five sub-graphs, each containing a wrench measure (⧫). The head terminal wrench is usually set to zero, so it is treated as a known variable (again ⧫).

The choice of the nodes where unknown wrenches (◊) are applied is instead totally arbitrary and depends on the application point of an interaction force.

For example, if the robot is moving unconstrained in the space, without incurring into contacts with itself or the surrounding as in Figure 2.5(a), unknown wrenches (◊) can be statically attached to the end-effectors of the main limbs, hands and feet. Whereas in an interaction scenario, such as the robot crawling on the floor (see Figure 2.5(b)), external wrenches are clearly assumed on wrists and knees. This application also highlights the importance of the inertial sensor, which allows performing the Newton-Euler computations with a floating base frame. Indeed, linear and angular velocity and acceleration of the head, measured by the sensor, change continuously as an effect of the progression of the robot on the floor, combined with the head movements.

In general, unknown wrenches due to any sort of contact cannot be statically attached to a specific link, since the application point of the external force (i.e. the centroid of the contact) is unknown and generally difficult to predict (unless visual feedback is exploited to predict possible contact situations, but it is not always reliable). However, thanks to the artificial tactile skin it is possible to retrieve such information dynamically: therefore the EOG structure can be defined on the fly based on the contact position at each time instant. In such cases, as a consequence of the fact that only one unknown is allowed per each sub-graph, the external force due to contact is the unknown (◊) while wrenches located at the end-effectors are assumed to be known and null (⧫).

Examples of sub-graphs are reported in Figure 5.32, which correspond to the contacts shown in Figure 5.31, where three different contact locations in the left arm are presented.

Model Validation

A rigid-body dynamics model has been used to describe the whole robot. Kinematics and dynamics parameters were retrieved from the CAD description of the robot. Two experiments prove the reliability of the approach:

1. both arms and legs FTS measurements were compared with their model-based prediction, during unconstrained movements (i.e. null external wrenches);
2. measurements from an external FTS, applied at a given position on the end-effectors, were compared with their estimation.

right/left arm/leg) and k means that the corresponding link is the k-th for that specific limb.

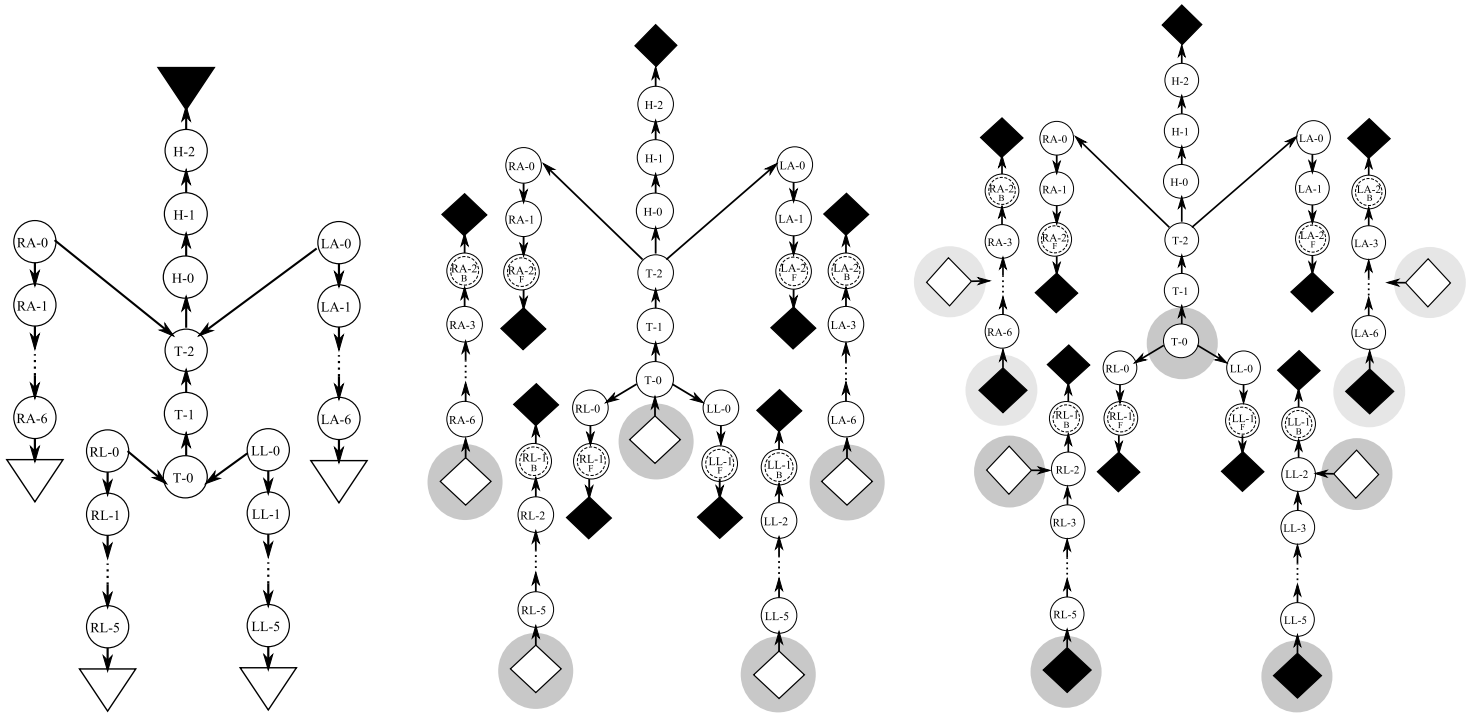


Figure 5.26: Representation of iCub's kinematic and dynamic EOG. **5.26(a)**: iCub's kinematic EOG. It is noticeable that the inertial sensor measure (▼) is the unique source of kinematic information for the whole branched system. **5.26(b)**: iCub's dynamic EOG, when iCub is standing on the mainstay and moving freely in the space, as shown in Figure 2.5(a). Given the four FTS, the main graph is cut by the four links hosting the sensors, and a total of five sub-graphs are finally generated. The unknowns are the external wrenches at the end-effector: if the robot does not collide with environment, they must be zero, whereas if a collision happens, an external wrench must arise. The displacement between the expected and the estimated wrenches allows detecting contacts with the environment. Of course, the hypothesis holds that interactions can only occur at the end-effectors. The external wrench on top of the head is assumed to be null. Notice that the mainstay is represented with a unknown wrench ◊. **5.26(c)**: iCub's dynamic EOG, when the iCub is crawling like a baby, as shown in Figure 2.5(b). As in the previous case, five sub-graphs have been generated after the insertion of the four FTS measurements, but unlike the free-standing case, here the mainstay wrench is missing, being the iCub floating (unfixed) on the floor. Specific locations for the contacts with the environment are specified as being part of the task: thus, the unknown external wrenches (◊) are placed at wrists and knees, while wrenches at the feet and palms are assumed known and null (⧫). Interestingly, while moving on the floor the contact with the upper part could be varying (e.g. wrists, palms, elbows), so the unknown wrench could be placed in different locations than the ones shown in the graph.

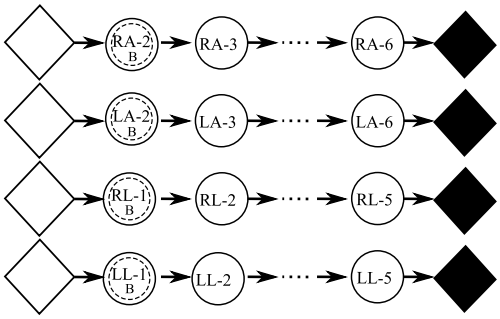


Figure 5.27: Enhanced graphs for predicting the four FTS measurements, $\hat{w}_s$, when the external wrench acting at the end-effectors (hands and feet) is known, typically null.

FTS Predictions

During unconstrained, contact-free motion, as shown in Figure 2.5(a), the measurements w_s from the four six-axes FTS embedded in the limbs have been compared with the analogous quantities $\hat{w}_s$ predicted by the dynamical model. Sensor measurements w_s can be predicted assuming known wrenches at the limbs extremities (hands or feet) and then propagating forces up to the sensors. In this case, null wrenches were assumed, because of absence of contact with the environment. It must be noticed that in Figure 2.5(a) all limbs are moving freely in the space without colliding with the robot own body or the environment. Though having a fixed base (the robot is supported by a metallic mainstay mechanically inserted into its hip), remarkably also the head is moving: thus, the presence of the inertial sensor is crucial for the computation of joint torques. The EOG in this case is shown in Figure 5.27. Table 5.2 summarizes the statistics of the errors $w_s - \hat{w}_s$ for each limb during the sequence of movements in Figure 2.5(a). In particular, the table shows the mean and the standard deviation of the errors between measured and predicted sensor wrench during the movements. Figure 5.28 plots the error between w_s and $\hat{w}_s$ for the right arm during the same sequence of movements (only one limb out of four is shown without loss of generality).

External Wrench Estimation

When solving the dynamic EOG in Figure 5.26(b), it is possible to retrieve one external wrench per sub-graph. Thus, we compared the estimation of an external wrench applied at the end-effector with a direct measure of it, through a free-standing six-axes FTS which was "pushed" on the terminal link. In particular, a wrench w^E was exerted on the left hand and measured with the external FTS. Its value was then compared with $\hat{w}^E$, the estimation of the external wrench obtained by propagating the embedded FTS measure in the sub-graph until the frame where w^E was applied. A plot of w^E and $\hat{w}^E$ is reported in Figure 5.29.

As a counter evidence of the reliability of the method we compared the torques $\hat{\tau}$, determined with (5.14) with the ones corresponding to the projection on joints of an external wrench applied at the end-effector $\tau^E = J_E^\top w^E$, where $J_E \in \mathbb{R}^{6\times n}$ is the Jacobian (here referred to the frame of the node connecting torso, head and arms).

right arm: $\epsilon \triangleq \hat{w}_{s,RA} - w_{s,RA}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	-0.3157	-0.5209	0.7723	-0.0252	0.0582	0.0197
σ_ϵ	0.5845	0.7156	0.7550	0.0882	0.0688	0.0364

left arm: $\epsilon \triangleq \hat{w}_{s,LA} - w_{s,LA}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	-0.0908	-0.4811	0.8699	0.0436	0.0382	0.0030
σ_ϵ	0.5742	0.6677	0.7920	0.1048	0.0702	0.0332

right leg: $\epsilon \triangleq \hat{w}_{s,RL} - w_{s,RL}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	-1.6678	3.4476	-1.5505	0.4050	-0.7340	0.0171
σ_ϵ	3.3146	2.7039	1.7996	0.3423	0.7141	0.0771

left leg: $\epsilon \triangleq \hat{w}_{s,LL} - w_{s,LL}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	0.2941	-5.1476	-1.9459	-0.3084	-0.8399	0.0270
σ_ϵ	1.8031	1.8327	2.3490	0.3365	0.8348	0.0498

*: $\epsilon \triangleq \hat{w} - w = [\epsilon_{f_0}, \epsilon_{f_1}, \epsilon_{f_2}, \epsilon_{\mu_0}, \epsilon_{\mu_1}, \epsilon_{\mu_2}]$
*: SI Unit: $f : [N], \mu : [Nm]$.

Table 5.2: Errors in predicting FTS measures (see text for details)

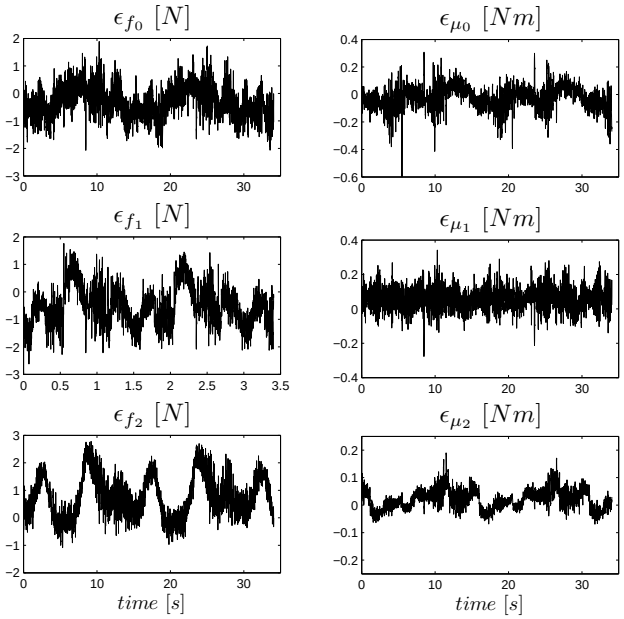


Figure 5.28: Right arm: error between the wrenches measured by the FT sensor $w_{s,RA}$ and the one predicted with the model $\hat{w}_{s,RA}$, during the "Yoga" demo.

During this experiment the arm is not moving, while the external force is applied on the hand. Joint torques measured with the *virtual torque sensors* are $\hat{\tau} = \hat{\tau}^I + \hat{\tau}^E$, being τ^I the internal joint torque, i.e. the torque which is due to the intrinsic dynamic of the system. τ^E, i.e. the external force projected on joints, instead is not affected by the internal dynamics (e.g. the gravitational component in this specific static case). Figure 5.30 shows a comparison of the variation of torque, due to an external wrench application. In particular, we show the comparison between τ^E and $\hat{\tau}^E = \hat{\tau} - \hat{\tau}^I$.

Exploiting the tactile feedback

As anticipated, the iCub artificial "skin" [Cannata et al., 2008, Roboskin Project, www] allows retrieving information about the location of possible contact points (i.e. location of externally applied wrenches) practically on the most of the robot body. Figure 5.32 shows how the dynamism of the EOG method can be fully exploited when the link where the contact occurs is known through tactile measurements.

We remark again that Figure 2.5(a)-2.5(b) are only possible instances of the EOG, and that the graph is continuously re-created along with the update of the sensory information coming from the tactile skin, indicating the contact locations.

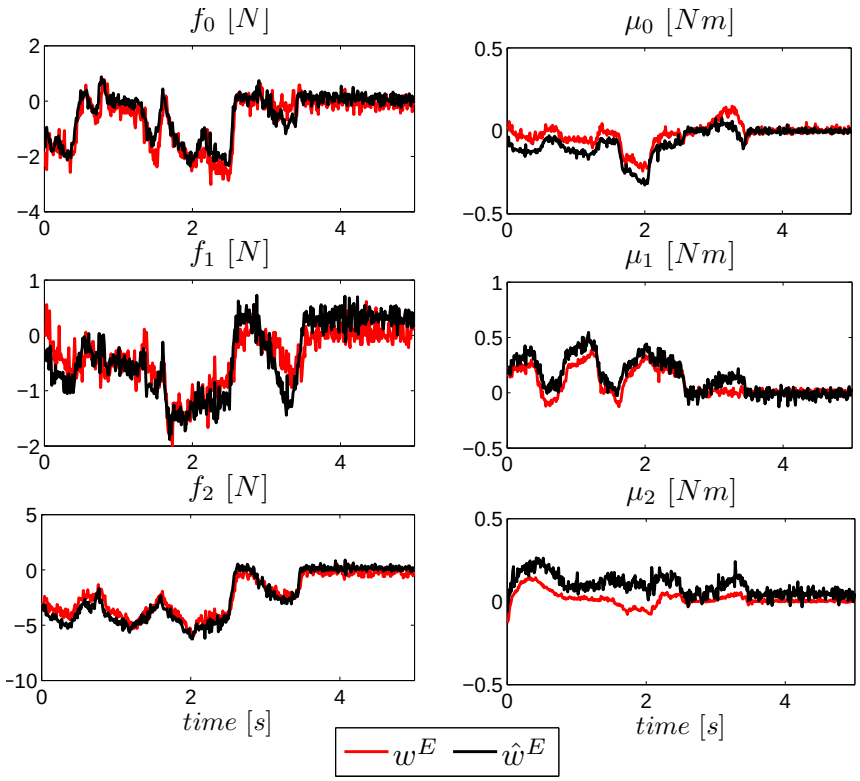


Figure 5.29: Left arm: comparison between the external wrench estimated after the FT sensor measurements and the one measured by an external FT sensor, placed on the palm of the left hand.

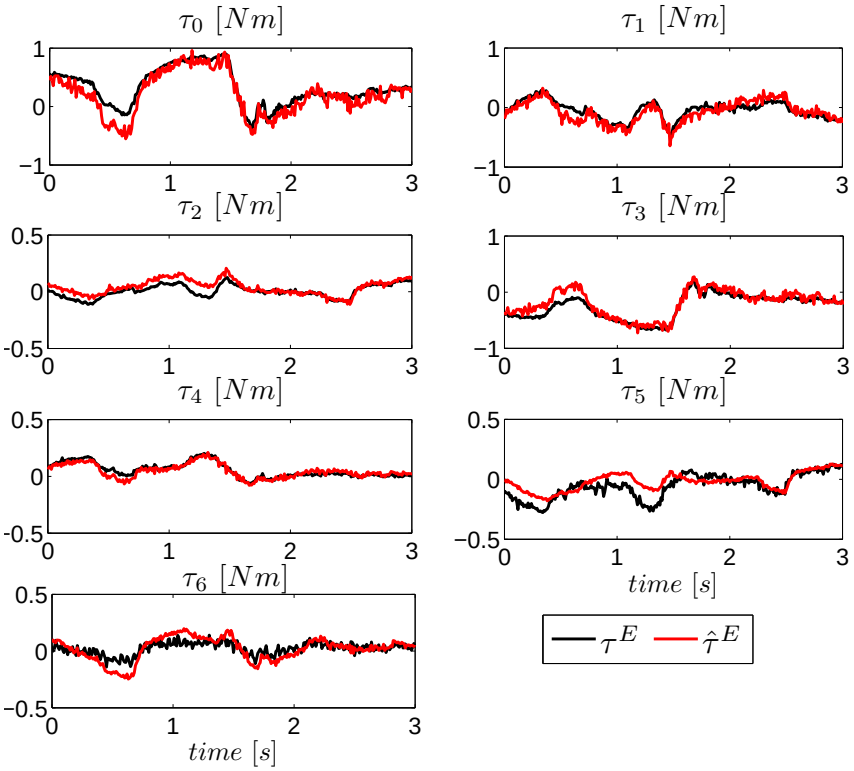


Figure 5.30: Left arm: comparison between the torques computed exploiting the embedded FTS and the ones obtained by projecting the external FTS on the joints through the Jacobian (see text).

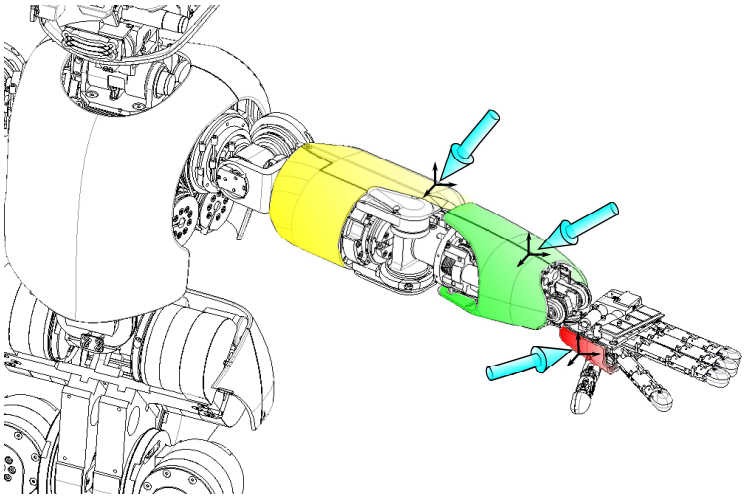


Figure 5.31: Some possible application points (marked with a frame) for external forces (arrows) arising during contact of the iCub arm with the environment. Upper, fore-arm and palm are covered with plastic shells, providing the base for the tactile elements.

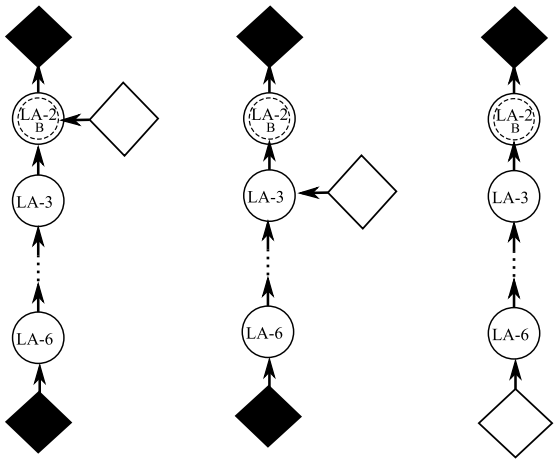


Figure 5.32: A sketch of the different situations in case of contacts occurring at different locations in the left arm, as shown in Figure 5.31. The external wrench to estimate (◊) is attached to different nodes. In the first and second sub-graphs, the wrench at the end-effector is assumed to be known (⧫), typically null, since only one unknown per graph is allowed (see text for details).

5.5.3 Joint impedance control of the iCub elbow

Robots with rigid joint, like iCub, can be controlled in torque by simply commanding a feed-forward torque, i.e. $\tau_{\text{ff}} = \tau$. If robots are actuated by flexible joints, joint stiffness can be also controlled. In iCub, actuation is provided by rigid electric motors, however a joint impedance control interface is available, which can be exploited to emulate a compliant joint: in particular, it is possible to control the equilibrium position of a virtual spring and its stiffness and damping. Precisely, the joint impedance control law is:

$$\tau^* = -k_S\,(q - q^*) - k_D\,\dot{q} + \tau_{\text{offset}} \tag{5.15}$$

where $k_S, k_D > 0$ are the stiffness and damping constants, emulating a virtual spring at the joint, and τ_{ff} a feed-forward offset torque. The desired computed torque τ^* is tracked at lower level by a simple PID algorithm. Playing with stiffness and damping, it is possible to make the robot joint behave like a soft or hard spring, while maintaining control on the desired joint position. Notably, k_S and k_D are physically related, as shown in Figure 5.33: the damping constant must increase with the stiffness, to prevent unstable behaviors. In the following experiment, we aim at demonstrating that by controlling a manipulator by suitably adapting stiffness and torque, in a feedforward manner, it is possible to cope with uncertainties and time delays in the system.

For sake of simplicity, we consider a single joint arm, moving in the vertical plane, described, as in Eq. 4.65, by the generic dynamic equation, which in the case of a single DOF can be simply written as:

$$\tau = m\,l_C\,cos(q)\,g + (m\,l^2 + I\,)\ddot{q} = G(q) + B\ddot{q} \tag{5.16}$$

where τ is the joint torque, m, l_C, I the link mass, length of the COM, and inertia respectively.

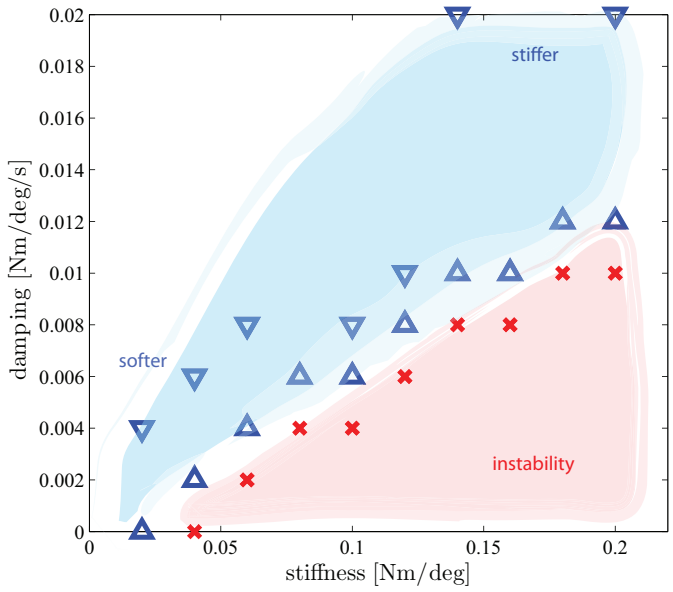


Figure 5.33: An experimental evaluation of the stiffness-damping relationship when controlling the elbow joint of the iCub. The lower region highlighted in red causes unstable behaviors of the arm when reaching its equilibrium point after an external force is applied. Boundary values were detected by overshoot of a step response. The light blue region shows feasible values, which make the joint behave like a softer or stiffer spring. The boundary region between the two can be ideally described by the following relationship: $k_D = 0.075k_S - 0.0015$.

We assume the following model of "human-like"[16] actuator holds:

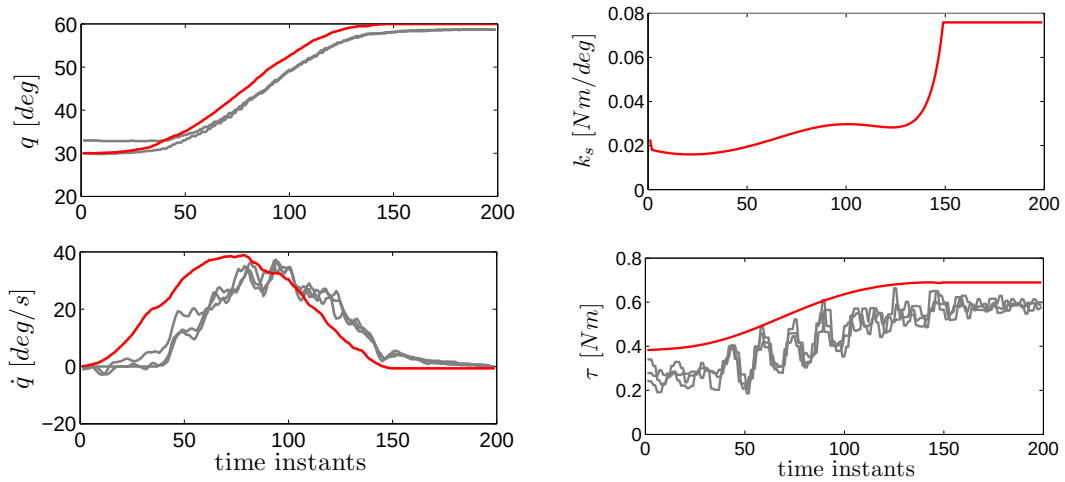
$$\tau = \tau_{\text{ff}} - k_S(q - q^*) - k_D(\dot{q} - \dot{q}^*) \tag{5.17}$$

where again k_S, k_D are positive constants representing the stiffness and damping of the elastic actuator. To simplify the problem statement, since the damping value is closely related to the stiffness value, the following relationship is assumed: $k_D = \lambda k_S$, for some $\lambda > 0$ (e.g., $\lambda = 0.2$). A stochastic optimal control problem is stated, which takes into account the time delay of the controlled system, and an additive noise e affecting the joint torque, which is generically due to uncertainties and modeling errors. The goal is to find the optimal stiffness and torque controls which make the arm perform an upward point-to-point movement while minimizing the following cost function:

$$\mathcal{J} = E\left\{w_p||q(T) - q^*(T)||^2 + w_v||\dot{q}(T)||^2 + \int_0^T ||u||_W^2 + w_p||q - q^*||^2 + w_v||\dot{q} - \dot{q}^*||^2 dt\right\}$$

with suitable weights w_p, w_v (e.g. $w_p = 10^6, w_v = 0.1w_p$). In this case, a suboptimal solution

[16] This actuator model emulates the main properties of the antagonist muscle structure in humans.



(a) (b)

Figure 5.34: An experiment consisting of multiple trials, the iCub elbow follows desired profiles (red) for stiffness and torque. Desired trajectories have been precomputed by the iLQG solving a stochastic optimal control problem. **5.34(a)** joint position and velocity. **5.34(b)** joint stiffness and torque.

has been found by means of the iLQG algorithm [Todorov and Li, 2005].[17] The following parameters, compatible with the forearm of the iCub, were used to describe the system dynamics:

- mass: $m = 0.812$kg
- link length: $a = 0.2735$m
- link COM: $l_C = 0.1033$m
- link inertia: $I = 5.0826 \times 10^{-3}$kg m^2

The desired movement was set from $q_0^* = -60$ to $q_T^* = -30$ degrees, which were remapped in the iCub elbow joint range. The movement duration T was set to 1.5s, while the control rate was set to 10ms. The additive noise e was modeled by:

$$e = (\alpha + \beta\dot{q})\eta \tag{5.18}$$

where η has a normal distribution ($\eta \approx N(0,1)$). Interestingly, the analysis has been performed on a further model, where the error is also proportional to the control torque:

$$e = (\alpha + \beta\dot{q} + \gamma\tau)\eta \tag{5.19}$$

which accounts for a control-dependent noise. In order to set reasonable values of the parameters, the modeling error has been identified: precisely, estimated joint torque has been

[17] The description of the iLQG method and its implementation is not reported here, for not being completely relevant to the experiment discussion.

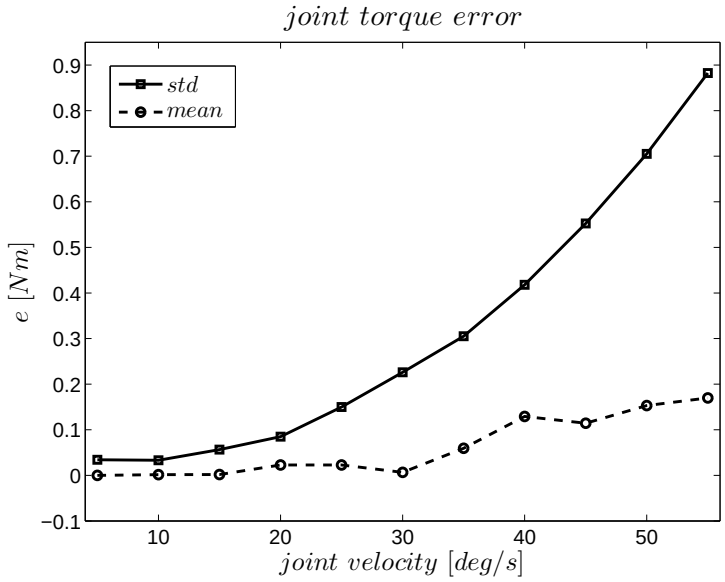


Figure 5.35: Mean $\bar{e}$ and standard deviation σ_e of the joint torque error from the elbow joint of iCub's right arm. The standard deviation is evidently bilinear with the increase of velocity, while the error mean is linearly proportional to the velocity. Polynomial fitting models are $\sigma_e = 1.254\dot{q}^2 - 0.351\dot{q} + 0.057$ and $\bar{e} = 0.213\dot{q} - 0.049$ respectively.

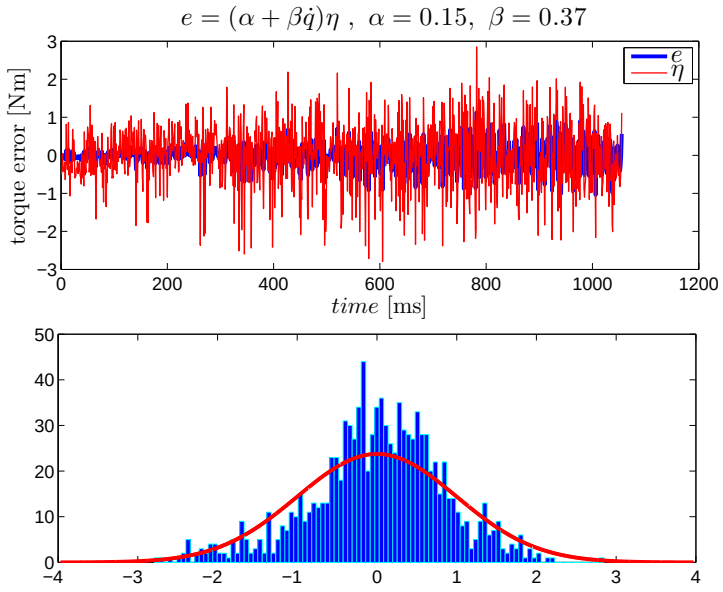
compared with the one computed by (5.16), so that $e = \tau_{\text{measured}} - \tau_{\text{model}}$. A sequence of upward movements with a minimum jerk profile was performed with the right arm of iCub, with increasing average velocity (from 0 to 55 deg/s): mean and standard deviations of the errors are reported in Figure 5.35. Then, a normality test on η was performed, exploring within reasonable accuracy the parameter space $\alpha - \beta$ over a discrete grid in the range $[0.0 : 0.001 : 1.0]$. Precisely, the Jarque-Bera test [Jarque and Bera, 1987] available in Matlab, was used. The best sets of parameters found after the exploration were $(\alpha^\circ, \beta^\circ) = (0.15, 0.37)$ for the model (5.18) and $(\alpha^\circ, \beta^\circ, \gamma^\circ) = (0.04, 0.41, 0.21)$ for (5.19): their fitting is shown in Figure 5.36.

Once the stochastic optimal control problem has been solved, the desired trajectories have been used to control the iCub elbow stiffness and torque, in a feedforward manner. To adapt the computational control (5.17) to the robot available interface (5.15), the following has been defined: $\tau_{\text{offset}} = \tau_{\text{ff}} + k_D\dot{q}^*$.

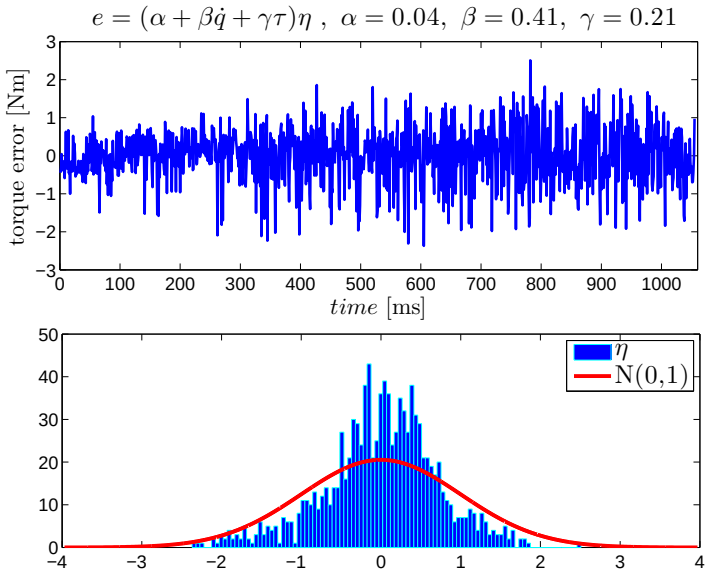
Preliminary results are shown in Figure 5.34. Some observations must be made. The damping effect is evident, introducing a small but noticeable delay in the trajectory. The stiffness trajectory features extremely low values: however, the point of this experiment is to show that the feed-forward term τ_{ff} is sufficient to achieve a desired behavior and that stiffness during trajectory must be increased only to counterbalance the uncertainties and noise in the problem. Thus, it is also correct to expect some misalignments between the desired and real trajectories. Moreover, there's a known problem with the stiffness values: the resolution of the stiffness val-

ues in the DSP is 0.01 Nm/deg, which is quite high for the trajectories we are aiming to: this means that the desired stiffness trajectory is practically "truncated" and actually is quantized. Another flaw in the experiments is that torques are estimated by the EOG on the basis of the measurements from the FTS, which have a certain drift during the use. This drift can be compensated, however it is not yet fully predictable. So it happens that a certain offset between the model and the real measurements is affecting our results per trial (for example, in the figures it is approximately 0.1Nm): this explains the differences between the desired torque and the one estimated (with a certain delay, again) by the robot dynamics module.

To overcome these issues, we are currently performing the same experiments on a prototype of the new arm of the iCub, which is equipped with a joint torque sensor at the elbow.



(a)



(b)

Figure 5.36: An experimental evaluation of the elbow joint torque error due to uncertainties in the dynamic model of the iCub arm. Examples of η (histogram) fitting a normal Gaussian (red line) are presented. **5.36(a)** shows the torque error e modeled by Eq. 5.18, while **5.36(b)** by Eq. 5.19.

Chapter 6

Conclusions

Optimal control theory has recently become of great interest for neuroscience, since it provides a framework and a rich set of tools for describing and modeling computational motor control in a convenient way. The existence of "optimality" principles in sensori-motor control in the brain and in the CNS has been demonstrated in different experiments, focusing on the precomputation of motor commands, trajectories, feedback and feed-forward commands. However, there is not a unique answer to what mechanisms are really at the basis of our motions: a variety of optimal control models has been proposed, and since different models are apparently equally effective in explaining movements in the same or a similar context, it is difficult to assess which is the most representative of human movements, and why.

Notably, by investigating human motor control it is not infrequent to address similar problems as the ones engineers usually face in robotics, and especially in humanoid robotics. By implementing what we believe the optimality principles underlying the human CNS, it is possible not only to provide an experimental verification of the proposed models, but also achieve behaviors which may outperform traditional classical controls. The point being that while classical control theory has been mainly focused on minimizing tracking errors, reject disturbances, minimize the motion duration, guarantee stability, etc., the study of human motor control may unveil other criteria as the fundamentals for achieving peculiar performances and characteristics which make our motor control so efficient and "optimal", in a sense.

But, when trying to transfer human models on a robot, one faces several limits: in terms of physical and structural impairments in the architecture, in the technology for actuation and sensing, and in the theoretical tools which can be exploited to generate controls. It is then crucial to catch the biological principles of human motor controls and understand to which extent these principles can be implemented in a robotic platform, and evaluate what is the revenue of that process. This thesis actually followed this point.

The first part of this thesis addressed the solution of stochastic optimal control problems, which can be used to state control problems to study both humans and humanoids, according to recent theories in computational motor control. The availability of powerful mathematical tools for generating suitable controls is particularly crucial: indeed, classical control theory is built on top of strong assumptions, as the known LQG conditions, which are hard to be verified in experimental platforms such as robots. With this thesis, we propose a theoretical

tool which can be used to generate optimal controls for a wider class of problems, since it does not require particular features (neither in the system nor in the cost functions) but the continuity and differentiability of their actors. Thus, it can be applied to complex systems such as humanoid without restricting assumptions. The applicability of the proposed approach has been empirically demonstrated by several numerical and experimental examples, with linear and nonlinear systems.

However, there are several issues which still need to be addressed and remain open for further research. First, the convergence for the stochastic approximation algorithm cannot be guaranteed, and in general does not always occur. A certain practical experience with the offline optimization is required. A possible solution to cut drastically the time required to train the neural approximators would be to parallelize the computations and run the optimization over a high-performance cluster grid of GPUs. This is not enough to guarantee to find the global optima of the control problem, however a considerable search in the parameter space can be performed, even if not exhaustively, and to a certain extent it can help excluding unfeasible control laws. The time constraint is the main limitation of the proposed technique, since finding a solution (global or local) requires a considerable amount of training time and samples, thus limiting the application domain of the technique to offline optimization. However, once the approximating functions are trained, even in a rough manner, the algorithm is also suited for an incremental training which can be perfectly integrated on a modular robot architecture.

Despite its flaws, results obtained applying the method in different case studies are promising: in particular, for the application in real-time domains such as robotics, since pre-computing the control laws allows saving time during online execution. The performances of the RH neural controller combined with the CLIK were impressive on James, and given the elasticity of the platform were particularly positive. It is also worth noting that the experiments performed on James were also the first (to the best of our knowledge) where the ERIM has been successfully applied for the control a real physical system.

Another interesting advantage of the proposed approach is that the formulation of the motor control problem in the stochastic optimal control framework is particularly suited for finding control laws in a modular architecture, where multiple agents coexist and in a sense cooperate to achieve the same goal. A brief hint of these future developments has been discussed in Section 3.3.4, where the main concepts of Team Theory have been introduced. Since the CA of the robot is basically a pool of modular controllers, interacting with each other regularly, it would be more than interesting to investigate a parallel between CNS and CA, and model interconnections between different controllers acting on the robot: each may have its own goal, but they all cooperate for the same task accomplishment.

The second part of the thesis has been dedicated to the development of a theoretical framework which allows integrating dynamics in motor control models, providing both an estimate of the joint torques and a detection of external forces due to contacts.

The estimation of the latter on the iCub has been particularly significant: it is clear that the proposed method is fundamental for enabling force/torque control in a platform where a direct joint torque feedback is missing. The main disadvantage of the FTS-based method is indeed the delay caused by the software estimation of the torques, which cannot be done directly on the DSP but is performed on a remote machine. In this regard, we are currently investigating the

performances of the torque estimation via EOG on the prototype of the new iCub arm, which will be equipped with direct joint torque sensing.

However, numerous experiments in iCub involving active force control showed the effectiveness of the proposed approach, which has been already integrated in other modules of the iCub software library. The current implementation of the robot dynamics paves the way for numerous developments: e.g., the integration of the dynamics computations with the tactile feedback of the artificial skin, for accurate detection of external forces; the explicit computation of the forward and inverse dynamics of the iCub limbs, to allow more advanced control schemes; the accurate estimation of the dynamical parameters of the robot (since only CAD parameters have been used so far).

The message emerging from the experimental results discussed in this thesis is that there is more than a rationale in using the formalism of stochastic optimal control to model human movements. The main benefit of such an abstract mathematical framework is that the key concepts and motion criteria can be "ported" from humans to humanoids and vice versa in a relatively easy way, i.e. a parallel between the two systems can be done. Within certain limitations, it is also possible to integrate successfully the computational motor control models provided by neuroscience into controllers for humanoid robots, which is particularly convenient since the proposed optimal control framework is also suited for continuous adaptation and incremental learning, i.e. it can be combined with a developmental approach to humanoid robotics.

Publications

Submitted / in preparation

1. S. Ivaldi, M. Fumagalli, M. Randazzo, F. Nori, G. Metta, G. Sandini. "Computing robot internal/external wrenches by means of F/T sensors: theory and implementation on the iCub humanoid."

2. M. Fumagalli, S. Ivaldi, M. Randazzo, F. Nori, G. Metta, G. Sandini. "Force feedback exploiting tactile and proximal force/torque sensing: theory and implementation on the humanoid robot iCub".

3. L. Natale, F. Nori, G. Metta, M. Fumagalli, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz and G. Sandini. "Studying developmental robotics on the iCub platform".

4. B. Berret, S. Ivaldi, F. Nori, G. Sandini. "On the role of muscle co-contraction in planning movements: implications on novel robotic actuators".

≈≈≈

Book chapters

1. M. Fumagalli, A. Gijsberts, S. Ivaldi, L. Jamone, G. Metta, L. Natale, F. Nori and G. Sandini. "Learning how to exploit proximal force sensing: a comparison approach". O. Sigaud & J. Peters (eds.), *From Motor Learning to Interaction Learning in Robots, Studies in Computational Intelligence*, vol. 264, Springer-Verlag pp.159-177, 2010.

2. S. Ivaldi, M. Baglietto, G. Metta and R. Zoppoli. "An application of receding-horizon neural control in humanoid robotics". *Assessment and Future Directions of Nonlinear Model Predictive Control, in L. Magni et al. (Eds.): Nonlinear Model Predictive Control*, LNCIS 384, Springer-Verlag Berlin Heidelberg, pp. 541-550, 2009.

International Conference papers

1. S. Ivaldi, M. Fumagalli, F. Nori, M. Baglietto, G. Metta and G. Sandini. "Approximate optimal control for reaching and trajectory planning in a humanoid robot". *Proc. of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS '10*. October 18-22 2010, Taipei, Taiwan.

2. S. Ivaldi, M. Baglietto, F. Davoli and R. Zoppoli. "Optimal control of communication in energy constrained sensor networks through team theory and Extended RItz Method". *Proc. of the 2009 Int. Joint Conference on Neural Networks - IJCNN '09*, pp. 1372-1379. June 2009, Atlanta, GA, USA.

3. S. Ivaldi, M. Baglietto, G. Metta and R. Zoppoli. "An application of receding-horizon neural control in humanoid robotics". *Proc. Int. Workshop on current research and future directions of Nonlinear Model Predictive Control - NMPC '08*, pp. 1-8. September 2008, Pavia, Italy.

4. S. Ivaldi, M. Baglietto and R. Zoppoli. "Finite and Receding Horizon Regulation of a Space Robot". *Proc. Int. Conf. on Mathematical Problems in Engineering , Aerospace and sciences - ICNPAA '08*, pp. 608-616. June 2008, Genoa, Italy.

Refereed short papers and presentations at International conferences

1. S. Ivaldi, A. Sciutti. "Aquabot". *"Affordable robots" idea context of the 2009 Int. Conf. on Advanced Robotics - ICAR '09.* June 2009, Munich, Germany. Third rank. (Project/presentation)

2. S. Ivaldi, A. Sciutti. "Night garbage collectobot". *"Affordable robots" idea context of the 2009 Int. Conf. on Advanced Robotics - ICAR '09.* June 2009, Munich, Germany. Accepted, but not selected for the final context. (Project/presentation)

3. S. Ivaldi, M. Baglietto, G. Metta, R. Zoppoli and G. Sandini. "A finite and receding horizon neural controller in humanoid robotics". *IEEE/RAS Int. Conference on Intelligent Robots and Systems - Workshop: Robotics challenges for machine learning II - IROS '08.* September 2008, Nice, France. (Short paper and poster)

Refereed abstracts at National conferences

1. S. Ivaldi, M. Baglietto, F. Davoli and R. Zoppoli. "Extended Ritz Method for optimal control of communication in energy constrained mixed (analog/digital) transmissions". *SIDRA annual conference '09.* September 2009, Siracusa, Italy. (Abstract and poster)

2. S. Ivaldi, M. Baglietto, G. Metta and R. Zoppoli. "An application of receding horizon neural control in humanoid robotics". *SIDRA annual conference '08.* September 2008, Vicenza, Italy. (Abstract and talk)

3. S. Ivaldi, M. Baglietto, R. Zoppoli. "Extended Ritz Method for optimal control in communication problems". *SIDRA Annual Conference '07.* September 2007, Genoa, Italy. (Abstract and poster)

Bibliography

[Aicardi et al., 1995] Aicardi, M., Casalino, G., Bicchi, A., and Balestrino, A. (1995). Closed loop steering of unicycle like vehicles via lyapunov techniques. *IEEE Robotics Automation Magazine*, 2(1):27–35.

[Albers, 2002] Albers, A. (2002). Conceptual design of humanoid robots. In *The Third IARP International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba Research Center, AIST, Tsukuba, Ibaraki, Japan.

[Albu-Schaffer et al., 2010] Albu-Schaffer, A., Wolf, S., Eiberger, O., Haddadin, S., Petit, F., and Chalon, M. (2010). Dynamic modelling and control of variable stiffness actuators. In *IEEE Int. Conf. on Robotics and Automation*, pages 2155–2162, Anchorage, Alaska, USA.

[Amundson et al., 2005] Amundson, K., Raade, J., Harding, N., and Kazerooni, H. (2005). Hybrid hydraulic-electric power unit for field and service robots. In *IEEE/RSJ Int Conf. on Intelligent Robots and Systems*.

[Arechavaleta et al., 2008] Arechavaleta, G., Laumond, J., Hicheur, H., and Berthoz, A. (2008). An optimality principle governing human walking. *IEEE Transactions on Robotics*, 24:5–14.

[ATI, www] ATI (www). http://www.ati-ia.com/products/ft/ft_models.aspx?id=mini45.

[Atkenson and Whitman, 2009] Atkenson, C. and Whitman, E. (2009). Dynamic programming approaches to humanoid behavior optimization. In *IEEE-RAS Humanoids 09 - Workshop: Modeling, Simulation and Optimization of Bipedal Walking*, Paris, France.

[Atkeson and Stephens, 2007] Atkeson, C. and Stephens, B. (2007). Multiple balance strategies from one optimization criterion. In *IEEE Int. Conf. on Humanoid Robots*.

[Atkeson et al., 2000] Atkeson, C. G., Hale, J. G., Pollick, F., Riley, M., Kotosaka, S., Schaul, S., Shibata, T., Tevatia, G., Ude, A., Vijayakumar, S., Kawato, E., and Kawato, M. (2000). Using humanoid robots to study human behavior. *IEEE Intelligent Systems and Their Applications*, 15(4):46–56.

[Baglietto, 1998] Baglietto, M. (1998). *Nonlinear Approximators for Team Optimal Control Problems*. PhD thesis, University of Genova.

[Baglietto et al., 2001a] Baglietto, M., Parisini, T., and Zoppoli, R. (2001a). The case of dynamic routing in traffic networks. *IEEE Transactions on Neural Networks*, 12:485–502. 2004 IEEE Transactions on Neural Networks Outstanding Paper Award.

[Baglietto et al., 2001b] Baglietto, M., Parisini, T., and Zoppoli, R. (2001b). Numerical solutions to the witsenhausen counterexample by approximating networks. *IEEE Transactions on Automatic Control*, 46(9):1471–1477.

[Barambones and Etxebarria, 2002] Barambones, O. and Etxebarria, V. (2002). Robust neural control for robotic manipulators. *Automatica*, 38:235 – 242.

[Barron, 1993] Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930 – 945.

[Barron, 1994] Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14:115 – 133.

[Bauml et al., 2010] Bauml, B., Wimbock, T., and Hirzinger, G. (2010). Kinematically optimal catching a flying ball with a hand-arm-system. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2592–2599, Taipei, Taiwan.

[Beamish et al., 2006] Beamish, D., Bhatti, S. A., MacKenzie, I. S., and Wu, J. (2006). Fifty years later: A neurodynamic explanation of fitts' law. *J R Soc Interface*, 3(10):649–654.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton Univeristy Press, Princeton, NJ.

[Ben-Itzhak and Karniel, 2008] Ben-Itzhak, S. and Karniel, A. (2008). Minimum acceleration criterion with constraints implies bang-bang control as an underlying principle for optimal trajectories of arm reaching movements. *Neural Computation*, 20(3):779–812.

[Bennequin et al., 2009] Bennequin, D., Fuchs, R., Berthoz, A., and Flash, T. (2009). Movement timing and invariance arise from several geometries. *PLoS Computational Biology*, 5(7):e1000426.

[Berret et al., 2008] Berret, B., Darlot, C., Jean, F., Pozzo, T., Papaxanthis, C., and Gauthier, J. (2008). The inactivation principle: mathematical solutions minimizing the absolute work and biological implications for the planning of arm movements. *PLoS Computational Biology*, 4(10):e1000194.

[Bertsekas and Tsitsiklis, 1996] Bertsekas, D. and Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Athena.

[Bertsekas, 1995] Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific.

[Biess et al., 2007] Biess, A., Liebermann, D. G., and Flash, T. (2007). A computational model for redundant human three-dimensional pointing movements: integration of independent

spatial and temporal motor plans simplifies movement dynamics. *The Journal of Neuroscience*, 27(48):13045–13064.

[Biess et al., 2006] Biess, A., Nagurka, M., and Flash, T. (2006). Simulating discrete and rhythmic multi-joint human arm movements by optimization of nonlinear performance indices. *Biological Cybernetics*, 95(1):31–53.

[Blair and Iwasaki, 2011] Blair, J. and Iwasaki, T. (2011). Optimal gaits for mechanical rectifier systems. *Automatic Control, IEEE Transactions on*, 56(1):59–71.

[Braganza et al., 2005] Braganza, D., W.E.Dixon, Dawson, D., and Xian, B. (2005). Tracking control for robot manipulators with kinematic and dynamic uncertainty. In *Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*.

[Braun et al., 2009] Braun, D., Aertsen, A., Wolpert, D., and Mehring, C. (2009). Learning optimal adaptation strategies in unpredictable motor tasks. *The Journal of Neuroscience*, 29(20):6472–6478.

[Brock et al., 2008] Brock, O., Kuffner, J., and Xiao, J. (2008). *Springer Handbook of Robotics*, chapter Motion for manipulation tasks, pages 615–645. Siciliano, Khatib (Eds.), Springer.

[Caccavale et al., 2005] Caccavale, F., Natale, C., Siciliano, B., and Villani, L. (2005). Integration for the next generation: embedding force control into industrial robots. *IEEE Robotics Automation Magazine*, 12(3):53–64.

[Calinon et al., 2010] Calinon, S., Sardellitti, I., and Caldwell, D. (2010). Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan.

[Cannata et al., 2008] Cannata, G., Maggiali, M., Metta, G., and Sandini, G. (2008). An embedded artificial skin for humanoid robots. In *IEEE Int. Conf. on Multisensor Fusion and Integration*, Seoul, Korea.

[Cawley, 2006] Cawley, G. C. (2006). Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *IJCNN-2006: Proceedings of the International Joint Conference on Neural Networks*, pages 1661–1668, Vancouver, BC, Canada.

[Chevallereau and Aoustin, 2001] Chevallereau, C. and Aoustin, Y. (2001). Optimal reference trajectories for walking and running of a biped robot. *Robotica*, 19:557–569.

[Chiaverini et al., 1991] Chiaverini, S., Egeland, O., and Kanestrom, R. K. (1991). Achieving user-defined accuracy with damped least-squares inverse kinematics. In *Proc. Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments', 91 ICAR*, pages 672–677.

[Chiaverini et al., 2008] Chiaverini, S., Oriolo, G., and Walker, I. (2008). *Springer Handbook of Robotics*, chapter Kinematically redundant manipulators, pages 245–268. Siciliano, Khatib (Eds.), Springer.

[Chiaverini et al., 1999] Chiaverini, S., Siciliano, B., and Villani, L. (1999). A survey of robot interaction control schemes with experimental comparison. *Mechatronics, IEEE/ASME Transactions on*, 4(3):273 –285.

[Cormen et al., 2002] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2002). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2 edition.

[De Luca, 2006] De Luca, A. (2006). Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1623–1630.

[Degallier et al., 2008] Degallier, S., Righetti, L., Natale, L., Nori, F., Metta, G., and Ijspeert, A. (2008). A modular bio-inspired architecture for movement generation for the infant-like robot icub. In *IEEE RAS / EMBS Int. Conf. on Biomedical Robotics and Biomechatronics*, Scottsdale, Arizona.

[Denavit and Hartenberg, 1955] Denavit, J. and Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 23:215–221.

[Diedrichsen et al., 2010] Diedrichsen, J., Shadmehr, R., and Ivry, R. (2010). The coordination of movement: optimal feedback control and beyond. *Trends in Cognitive Science*, 14:1.

[Diehl et al., 2006] Diehl, M., Bock, H., Diedam, H., and Wieber, P. (2006). *Fast Motions in Biomechanics and Robotics*, chapter Fast Direct Multiple Shooting algorithms for optimal robot control, pages 65–93. LNCIS 340, Springer Berlin / Heidelberg.

[Diehl et al., 2009] Diehl, M., Ferreau, H., and Haverbeke, N. (2009). *L. Magni et al. (Eds.): Nonlinear Model Predictive Control*, chapter Efficient numerical methods for nonlinear MPC and Moving Horizon estimation, pages 541–550. LNCIS 384, Springer-Verlag Berlin Heidelberg.

[Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley-Interscience.

[Dupree et al., 2009] Dupree, K., Patre, P., Johnson, M., and Dixon, W. (2009). Inverse optimal adaptive control of a nonlinear euler-lagrange system, part i: Full state feedback. In *48th IEEE Conf. on Decision and Control, held jointly with 28th Chinese Control Conference. CDC/CCC 2009*, pages 321–326.

[Eiberger et al., 2010] Eiberger, O., Haddadin, S., Weis, M., Albu-Schäffer, A., and Hirzinger, G. (2010). On joint design with intrinsic variable compliance: derivation of the DLR QA-joint. In *IEEE Int. Conf. on Robotics and Automation*, pages 1687–1694.

[Faulhaber, www] Faulhaber (www). www.faulhaber.com.

[Featherstone, 2007] Featherstone, R. (2007). *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Featherstone, 2010] Featherstone, R. (2010). Exploiting sparsity in operational-space dynamics. *International Journal of Robotics Research*, 29:1353–1368.

[Featherstone and Orin, 2008] Featherstone, R. and Orin, D. E. (2008). *Handbook of Robotics*, chapter Dynamics, pages 35–65. Springer.

[Fitts, 1954] Fitts, P. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *J. Exp. Psychol.*, 47(6):381–391.

[Fitzpatrick et al., 2008] Fitzpatrick, P., Metta, G., and Natale, L. (2008). Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56:29–45.

[Fitzpatrick et al., 2010] Fitzpatrick, P., Natale, L., and Metta, G. (2010). The Cmaking of a humanoid. *The Kitware Source: Software Developer's Quarterly*, 13:7–9.

[Flanagan et al., 2003] Flanagan, J., Vetter, P., Johansson, R., and Wolpert, D. (2003). Prediction precedes control in motor learning. *Current Biology*, 13:146–150.

[Flash and Hogan, 1985] Flash, T. and Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703.

[Franklin et al., 2008] Franklin, D., Burdet, E., Tee, K., Osu, R., Chew, C.-M., Milner, T., and Kawato, M. (2008). Cns learns stable, accurate, and efficient movements using a simple algorithm. *The Journal of Neuroscience*, 28(44):11165–11173.

[Freescale DSP, www] Freescale DSP (www). http://www.freescale.com/.

[Fumagalli et al., 2010a] Fumagalli, M., Gijsberts, A., Ivaldi, S., Jamone, L., Metta, G., Natale, L., Nori, F., and Sandini, G. (2010a). *O. Sigaud et al. (Eds.): From Motor Learning to Interaction Learning in Robots*, chapter Learning to Exploit Proximal Force Sensing: a Comparison Approach. Springer-Verlag.

[Fumagalli et al., 2009] Fumagalli, M., Jamone, L., Metta, G., Natale, L., Nori, F., Parmiggiani, A., Randazzo, M., and Sandini, G. (2009). A force sensor for the control of a human-like tendon driven neck. In *IEEE-RAS International Conference on Humanoid Robots*.

[Fumagalli et al., 2010b] Fumagalli, M., Randazzo, M., Nori, F., Natale, L., Metta, G., and Sandini, G. (2010b). Exploiting proximal F/T measurements for the iCub active compliance. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan.

[Gori and Tesi, 1992] Gori, M. and Tesi, A. (1992). On the problem of local minima in backpropagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(1):76 –86.

[Gu and Hu, 2005] Gu, D. and Hu, H. (2005). A stabilizing receding horizon regulator for nonholonomic mobile robots. *IEEE Transactions on Robotics*, 21:1022–1028.

[Guigon et al., 2008] Guigon, E., Baraduc, P., and Desmurget, M. (2008). Optimality, stochasticity and variability in motor behavior. *Journ. Computational Neuroscience*, 24(1):57–68.

[Haddadin et al., 2008a] Haddadin, S., Albu-Schaffer, A., , and Hirzinger, G. (2008a). The role of the robot mass and velocity in physical human-robot interaction - part i: Non-constrained blunt impacts. In *IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, USA.

[Haddadin et al., 2010a] Haddadin, S., Albu-Schaffer, A., Eiberger, O., and Hirzinger, G. (2010a). New insights concerning intrinsic joint elasticity for safety. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[Haddadin et al., 2008b] Haddadin, S., Albu-Schaffer, A., Frommberger, M., , and Hirzinger, G. (2008b). The role of the robot mass and velocity in physical human-robot interaction - part ii: Constrained blunt impacts. In *IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, USA.

[Haddadin et al., 2010b] Haddadin, S., Urbanek, H., Parusel, S., Burschka, D., Rossmann, J., Albu-Schaffer, A., and Hirzinger, G. (2010b). Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3109 –3116.

[Hagan and Menhaj, 1994] Hagan, M. T. and Menhaj, M. B. (1994). Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993.

[Harris and Wolpert, 1998] Harris, C. M. and Wolpert, D. M. (1998). Signal-dependent noise determines motor planning. *Nature*, 394(6695):780–784.

[He and Geng, 2007] He, G.-P. and Geng, Z.-Y. (2007). Optimal motion planning of a one-legged hopping robot. In *IEEE Int.Conf. on Robotics and Biomimetics*, pages 1178–1183, Sanya, China.

[Hersch and Billard, 2006] Hersch, M. and Billard, A. (2006). A biologically-inspired controller for reaching movements. In *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, pages 1067–1072.

[Ho and Chu, 1972] Ho, Y. and Chu, K. (1972). Team decision theory and information structures in optimal control problems - part i. *IEEE Transactions on Automatic Control*, 17:15–28.

[Hornik et al., 1989] Hornik, K., Stinchombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

[Ivaldi et al., 2009a] Ivaldi, S., Baglietto, M., Davoli, F., and Zoppoli, R. (2009a). Optimal control of communication in energy constrained sensor networks through team theory and extended ritz method. In *Int. Joint Conf. on Neural Networks*, pages 1372–1379, Atlanta, GA, USA.

[Ivaldi et al., 2008a] Ivaldi, S., Baglietto, M., Metta, G., and Zoppoli, R. (2008a). An application of receding-horizon neural control in humanoid robotics. In *Int. Workshop on Assessment and future directions of Nonlinear Model Predictive Control*, Pavia, Italy.

[Ivaldi et al., 2009b] Ivaldi, S., Baglietto, M., Metta, G., and Zoppoli, R. (2009b). *L. Magni et al. (Eds.): Nonlinear Model Predictive Control. Towards New Challenging Applications*, chapter An application of receding-horizon neural control in humanoid robotics, pages 541–550. LNCIS 384, Springer-Verlag Berlin Heidelberg.

[Ivaldi et al., 2008b] Ivaldi, S., Baglietto, M., Metta, G., Zoppoli, R., and Sandini, G. (2008b). A finite and receding horizon neural controller in humanoid robotics. In *Int. Conf. On Intelligent Robots - IROS. Workshop: Robotics Challenges for Machine Learning II*.

[Ivaldi et al., 2008c] Ivaldi, S., Baglietto, M., and Zoppoli, R. (2008c). Finite and receding horizon regulation of a space robot. In *Int. Conf. on Mathem. Probl. in Engin. Aerospace and Sciences*, pages 608–616. Cambridge Scientific Publishers, London.

[Ivaldi et al., 2010] Ivaldi, S., Fumagalli, M., Nori, F., Baglietto, M., and Metta, G. (2010). Approximate optimal control for reaching and trajectory planning in a humanoid robot. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1290–1296, Taipei, Taiwan.

[Ivaldi et al., www] Ivaldi, S., Fumagalli, M., and Pattacini, U. (www). Doxygen documentation of the iDyn library. `http://eris.liralab.it/iCub/main/dox/html/group__iDyn.html`.

[Izawa et al., 2008] Izawa, J., Rane, T., Donchin, O., and Shadmehr, R. (2008). Motor adaptation as a process of reoptimization. *The Journal of Neuroscience*, 28(11):2883–2891.

[Jamone, 2010] Jamone, L. (2010). *Autonomous sensori-motor learning in a humanoid robot*. PhD thesis, University of Genoa, Italy.

[Jamone et al., 2006] Jamone, L., Metta, G., Nori, F., and Sandini, G. (2006). James: A humanoid robot acting over an unstructured world. In *Proc. 6th IEEE-RAS International Conference on Humanoid Robots*, pages 143–150.

[Janabi-Sharifi et al., 2000] Janabi-Sharifi, F., Hayward, V., and Chen, C.-S. J. (2000). Discrete-time adaptive windowing for velocity estimation. *IEEE Transactions on Control Systems Technology*, 8(6):1003–1009.

[Jarque and Bera, 1987] Jarque, C. and Bera, A. (1987). A test for normality of observations and regression residuals. *International Statistical Review*, 55(2):163–172.

[Kaneko et al., 2005] Kaneko, Y., Nakano, E., Osu, R., Wada, Y., and Kawato, M. (2005). Trajectory formation based on the minimum commanded torque change model using euler-poisson equation. *Systems and Computers in Japan*, 36:92–103.

[Kanoun et al., 2009] Kanoun, O., Yoshida, E., and Laumond, J. (2009). An optimization formulation for footsteps planning. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 202–207, Paris, France.

[Keerthi and Gilbert, 1988] Keerthi, S. and Gilbert, E. (1988). Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57:265–293.

[Kelso, 1982] Kelso, J., editor (1982). *Human motor behavior: an introduction*. Lawrence Erlbaum Associates Inc.

[Kim et al., 2000] Kim, Y. H., Lewis, F. L., and Dawson, D. M. (2000). Intelligent optimal control of robotic manipulators using neural networks. *Automatica*, 36:1355 – 1364.

[Konczak and Dichgans, 1997] Konczak, J. and Dichgans, J. (1997). The development toward stereotypic arm kinematics during reaching in the first 3 years of life. *Experimental Brain Research*, 117:346–354.

[Konczak et al., 2010] Konczak, J., Pierscianek, D., Hirsiger, S., Bultmann, U., Schoch, B., Gizewski, E., Timmann, D., Maschke, M., and Frings, M. (2010). Recovery of upper limb function after cerebellar stroke: Lesion symptom mapping. *Stroke*, 41:2191–2200.

[Kozlowski, 1998] Kozlowski, K. (1998). *Modelling and Identification in Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Krstic, 2009] Krstic, M. (2009). Inverse optimal adaptive control : the interplay between update laws, control laws, and lyapunov functions. In *American Control Conference*, pages 1250–1255.

[Krurková, 1997] Krurková, V. (1997). *Dimension-independent rates of approximation by neural networks*, pages 261–270. "Computer-intensive methods in control and signal processing: curse of dimensionality". Warwick and Karny Eds., Boston.

[Kulic and Croft, 2007] Kulic, D. and Croft, E. (2007). Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, 22:149–164.

[Kuo, 2005] Kuo, A. (2005). An optimal state estimation model of sensory integration in human postural balance. *Journal of Neural Engineering*, 2:S235–S249.

[Kurková and Sanguineti, 2005] Kurková, V. and Sanguineti, M. (2005). Error estimates for approximate optimization by the extended ritz method. *SIAM J. on optimization*, 15:461–487.

[Kusher and Yin, 1997] Kusher, H. J. and Yin, G. G. (1997). *Stochastic approximation algorithms and applications*. Springer-Verlag, New York.

[Kushner and Yang, 1995] Kushner, H. and Yang, J. (1995). Stochastic approximation with averaging and feedback: rapidly convergent "on-line" algorithms. *IEEE Transactions on Automatic Control*, 40:24–34.

[Kwon et al., 1983] Kwon, W., Bruckstein, A., and Kailath, T. (1983). Stabilizing state-feedback design via the moving horizon method. *International Journal of Control*, 37:631–643.

[Kwon and Han, 2005] Kwon, W. and Han, S. (2005). *Receding horizon control: Model Predictive Control for State Models*. Springer-Verlag. Advanced Textbooks in Control and Signal processing. London, UK.

[Kwon and Paearson, 1978] Kwon, W. and Paearson, A. (1978). On feedback stabilization of time-varying discrete linear systems. *IEEE Transactions on Automatic Control*, 23(3):479–481.

[Lacquaniti et al., 1983] Lacquaniti, F., Terzuolo, C., and Viviani, P. (1983). The law relating kinematic and figural aspects of drawing movements. *Acta Psychologica*, 54:115–130.

[Lagarde et al., 2009] Lagarde, M., Andry, P., Gaussier, P., Boucenna, S., and Hafemeister, L. (2009). Proprioception and imitation: on the road to agent individuation. In *From Motor to Interaction Learning in Robots*. Springer.

[Lengagne et al., 2009] Lengagne, S., Ramdani, N., and Fraisse, P. (2009). Planning and fast re-planning of safe motions for humanoid robots: Application to a kicking motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 441–446, St. Louis, US.

[Levenberg, 1944] Levenberg, K. (1944). A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168.

[Lockhart and Ting, 2007] Lockhart, D. and Ting, L. (2007). Optimal sensorimotor transformations for balance. *Nature Neuroscience*, 10(10):1329–1336.

[Luh et al., 1983] Luh, J., Fisher, W., and Paul, R. (1983). Joint torque control by a direct feedback for industrial robots. *IEEE Trans. on Automatic Control*, 28(2):153–161.

[MacKenzie, 1992] MacKenzie, I. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139.

[Maggiali et al., 2008] Maggiali, M., Cannata, G., Maiolino, P., Metta, G., Randazzo, M., and Sandini, G. (2008). Embedded distributed capacitive tactile sensor. In *Mechatronics 2008*, Limerick, Ireland.

[Mandersloot et al., 2006] Mandersloot, T., Wisse, M., and Atkeson, C. (2006). Controlling velocity in bipedal walking: A dynamic programming approach. In *6th IEEE-RAS Int. Conf. on Humanoid Robots*, pages 124–130.

[Marquardt, 1963] Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441.

[Matsui, 2008] Matsui, T. (2008). A new optimal control model for reproducing two-point reaching movements of human three-joint arm with wrist joint's freezing mechanism. In *IEEE Int. Conf. on Robotics and Biomimetics*, pages 383–388.

[Matsui et al., 2006] Matsui, T., Honda, M., and Nakazawa, N. (2006). A new optimal control model for reproducing human arm's two-point reaching movements: a modified minimum torque change model. In *IEEE Int. Conf. on Robotics and Biomimetics*, pages 1541–1546.

[Matsui et al., 2009] Matsui, T., Takeshita, K., and Shibusawa, T. (2009). Effectiveness of human three-joint arm's optimal control model characterized by hand-joint's freezing mechanism in reproducing constrained reaching movement characteristics. In *ICROS-SICE Int. Joint Conf. (ICCAS-SICE)*, pages 1206 –1211.

[Mayne and Michalska, 1990] Mayne, D. and Michalska, H. (1990). Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633.

[Metta et al., 2006] Metta, G., Fitzpatrick, P., and Natale, L. (2006). Yarp: yet another robot platform. *International Journal on Advanced Robotics Systems. Special Issue on Software Development and Integration in Robotics*, 1(8/9):975–997.

[Metta et al., 2010] Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Santos-Victor, J., Bernardino, A., and Montesano, L. (2010). The icub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks, special issue on Social Cognition: From Babies to Robots*, 23:1125–1134.

[Metta et al., 1999] Metta, G., Sandini, G., and Konczak, J. (1999). A developmental approach to visually-guided reaching in artificial systems. *Neural Networks*, 12(10):1413–1427.

[Mettin et al., 2010] Mettin, U., Shiriaev, A., Freidovich, L., and Sampei, M. (2010). Optimal ball pitching with an underactuated model of a human arm. In *IEEE Int. Conf. on Robotics and Automation*, Anchorage, Alaska, USA.

[Michalska and Mayne, 1993] Michalska, H. and Mayne, D. (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. on Automatic Control*, 38:1623–1633.

[Minguez et al., 2008] Minguez, J., Lamiraux, F., and Laumond, J.-P. (2008). *Handbook of Robotics*, chapter Motion planning and obstacle avoidance, pages 827–852. Springer.

[Mistry et al., 2008] Mistry, M., A.Theodorou, E., Liaw, G., Yoshioka, T., Schaal, S., and Kawato, M. (2008). Adaptation to a sub-optimal desired trajectory. In *Society for Neuroscience - Symposium on Advances in Computational Motor Control*, Washington DC, USA.

[Mistry et al., 2010] Mistry, M., Buchli, J., and Schaal, S. (2010). Inverse dynamics control of floating base systems using orthogonal decomposition. In *IEEE Int. Conf. on Robotics and Automation*.

[Mitrovic et al., 2010] Mitrovic, D., Klanke, S., and Vijayakumar, S. (2010). *O. Sigaud, J. Peters (Eds.): From Motor to Interaction Learning in Robotics*, chapter Adaptive Optimal Feedback Control with Learned Internal Dynamics Models, pages 65–84. SCI 264 Springer-Verlag Berlin Heidelberg.

[Mombaur et al., 2008] Mombaur, K., Laumond, J.-P., and Yoshida, E. (2008). An optimal control model unifying holonomic and nonholonomic walking. In *8th IEEE-RAS Int. Conf. on Humanoid Robots*, Daejeon, Korea.

[Mombaur et al., 2010] Mombaur, K., Truong, A., and Laumond, J.-P. (2010). From human to humanoid locomotionan inverse optimal control approach. *Autonomous Robots*, 28:369–383.

[Morasso, 1983] Morasso, P. (1983). Three dimensional arm trajectories. *Biological Cybernetics*, 48:187–194.

[Morel and Dubowsky, 1996] Morel, G. and Dubowsky, S. (1996). The precise control of manipulators with joint friction: A base force/torque sensor method. In *IEEE Int. Conf. on Robotics and Automation*, pages 360–365.

[Morel et al., 2000] Morel, G., Iagnemma, K., and Dubowsky, S. (2000). The precise control of manipulators with high joint friction using base force/torque sensing. *Automatica*, 36(7):931–941.

[Murray et al., 1994] Murray, R. M., Sastry, S. S., and Zexiang, L. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA.

[Nagengast et al., 2011] Nagengast, A., Braun, D., and Wolpert, D. (2011). Risk-sensitivity and the mean-variance trade-off: decision making in sensorimotor control. *Proceedings of The Royal Society B*, doi: 10.1098/rspb.2010.2518:1–8.

[Nakano et al., 1999] Nakano, E., Imamizu, H., Osu, R., Uno, Y., Gomi, H., Yoshioka, T., and Kawato, M. (1999). Quantitative examinations of internal representations for arm trajectory planning: Minimum commanded torque change model. *Journal of Neurophysiology*, 81:2140–2155.

[Nelson, 1983] Nelson, W. (1983). Physical principles for economies of skilled movements. *Biological Cybernetics*, 46:135–147.

[Nguyen and Widrow, 1990] Nguyen, D. and Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *IJCNN International Joint Conference on Neural Networks*, volume 3, pages 21–26, San Diego, CA, USA.

[Niyogi and Girosi, 1996] Niyogi, P. and Girosi, F. (1996). On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. *Neural Computation*, 8:819 – 842.

[Nori et al., 2007a] Nori, F., Jamone, L., Sandini, G., and Metta, G. (2007a). Accurate control of a human-like tendon-driven neck. In *Proc. 7th IEEE-RAS International Conference on Humanoid Robots*, pages 371–378.

[Nori et al., 2007b] Nori, F., Natale, L., Sandini, G., and Metta, G. (2007b). Autonomous learning of 3d reaching in a humanoid robot. In *IEEE/RSJ International Conference of Intelligent Robots and Systems*.

[Parisini and Zoppoli, 1995] Parisini, T. and Zoppoli, R. (1995). A receding horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31:1443–1451.

[Parmiggiani et al., 2009] Parmiggiani, A., Randazzo, M., Natale, L., Metta, G., and Sandini, G. (2009). Joint torque sensing for the upper-body of the iCub humanoid robot. In *Int. Conf. on Humanoid Robotics*, Paris, France.

[Pattacini, www] Pattacini, U. (www). Doxygen documentation of the iKyn library. `http://eris.liralab.it/iCub/main/dox/html/group__iKin.html`.

[Pattacini et al., 2010] Pattacini, U., Nori, F., Natale, L., Metta, G., and Sandini, G. (2010). An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan.

[Pianosi, 2008] Pianosi, F. (2008). *Novel methods for water reservoir management. Advisor: prof. Rodolfo Soncini-Sessa*. PhD thesis, Politecnico di Milano.

[Pianosi and Soncini-Sessa, 2008] Pianosi, F. and Soncini-Sessa, R. (2008). Extended ritz method for reservoir management over an infinite horizon. In *17th IFAC World Congress*.

[Pozzo et al., 1990] Pozzo, T., Berthoz, A., and Lefort, L. (1990). Head stabilisation during various locomotor tasks in humans. i. normal subjects. *Experimental Brain Research*, 82:97–106.

[Pratt and Williamson, 1995] Pratt, G. and Williamson, M. (1995). Series elastic actuators. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 399–406, Los Alamitos, CA, USA.

[Radner, 1962] Radner, R. (1962). Team decision problems. *Ann. Math. Statist.*, AC-17:15–21.

[Richardson and Flash, 2000] Richardson, M. and Flash, T. (2000). On the emulation of natural movements by humanoid robots. In *Int. Conf. on Humanoids Robots*.

[Richardson and Flash, 2002] Richardson, M. and Flash, T. (2002). Comparing smooth arm movements with the two-thirds power law and the related segmented-control hypothesis. *J Neurosci*, 22(18):8201–8211.

[Ritz, 1909] Ritz, W. (1909). Über eine neue methode zur lösung gewisser variationsprobleme der mathematischen physik. *J. für die reine und angewandte mathematik*, 135:1–61.

[Roboskin Project, www] Roboskin Project (www). `http://www.roboskin.eu`.

[RobotCub Project, www] RobotCub Project (www). http://www.robotcub.org.

[Sandini et al., 2004] Sandini, G., Metta, G., and Vernon, D. (2004). Robotcub: an open framework for research in embodied cognition. In *Proc. 4th IEEE/RAS International Conference on Humanoid Robots*, volume 1, pages 13–32.

[Sandini et al., 2007] Sandini, G., Metta, G., and Vernon, D. (2007). The iCub cognitive humanoid robot: An open-system research platform for enactive cognition. *50 Years of AI*, LNAI 4850:359–370.

[Sastry and Bodson, 1994] Sastry, S. and Bodson, M. (1994). *Adaptive Control: Stability, Convergence, and Robustness*. Advanced Reference Series (Engineering). Prentice-Hall.

[Schaal and Schweighofer, 2005] Schaal, S. and Schweighofer, N. (2005). Computational motor control in humans and robots. *Current Opinion in Neurobiology*, 15:675–682.

[Scheidt et al., 2000] Scheidt, R., Reinkensmeyer, D., Conditt, M., Rymer, W., and Mussa-Ivaldi, F. (2000). Persistence of motor adaptation during constrained, multi-joint, arm movements. *Journal of Neurophysiology*, 84(2):853–862.

[Schmidt et al., 1979] Schmidt, R., Zelaznik, H., Hawkins, B., Frank, J., and Quinn, J. (1979). Motoroutput variability: a theory for the accuracy of rapid motor acts. *Psychol Rev*, 47:415–51.

[Schmitz et al., 2010] Schmitz, A., Maggiali, M., Natale, L., Bonino, B., and Metta, G. (2010). A tactile sensor for the fingertips of the humanoid robot icub. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 18-22, 2010.

[Schultz and Mombaur, 2010] Schultz, G. and Mombaur, K. (2010). Modeling and optimal control of human-like running. *IEEE/ASME Trans. on Mechatronics*, 15(5):783–792.

[Sciavicco and Siciliano, 2000] Sciavicco, L. and Siciliano, B. (2000). *Robotica industriale: modellistica e controllo di manipolatori*. McGraw-Hill.

[Sciavicco and Siciliano, 2005] Sciavicco, L. and Siciliano, B. (2005). *Modelling and Control of Robot Manipulators*. Adv. textbooks in Control and Signal Processing. Springer, 2nd edition.

[Scott, 2004] Scott, S. (2004). Optimal feedback control and the neural basis of volitional motor control. *Nature Reviews Neuroscience*, 5:532–546.

[Seki and Tadakuma, 2004] Seki, H. and Tadakuma, S. (2004). Minimum jerk control of power assisting robot based on human arm behavior characteristics. In *IEEE Int. Conf. on System, Man and Cybernetics*, volume 1, pages 722–727.

[Seong and Widrow, 2001a] Seong, C.-Y. and Widrow, B. (2001a). Neural dynamic optimization for control systems. i. background. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31(4):482–489.

[Seong and Widrow, 2001b] Seong, C.-Y. and Widrow, B. (2001b). Neural dynamic optimization for control systems.ii. theory. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31(4):490–501.

[Seong and Widrow, 2001c] Seong, C.-Y. and Widrow, B. (2001c). Neural dynamic optimization for control systems.iii. applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31(4):502–513.

[Shadmehr et al., 2010a] Shadmehr, R., de Xivry, J., Xu-Wilson, M., and Shih, T.-Y. (2010a). Temporal discounting of reward and the cost of time in motor control. *The Journal of Neuroscience*, 30(31):10507–10516.

[Shadmehr et al., 2010b] Shadmehr, R., Smith, M., and Krakauer, J. (2010b). Error correction, sensory prediction, and adaptation in motor control. *Annual Review of Neuroscience*, 33:89–108.

[Shadmehr and Wise, 2005] Shadmehr, R. and Wise, S. (2005). *The Computational Neurobiology of Reaching and Pointing: a foundation for Motor Learning*. MIT Press.

[Shiller and Dubowsky, 1991] Shiller, Z. and Dubowsky, S. (1991). On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6):785–797.

[Siciliano and Villani, 1996] Siciliano, B. and Villani, L. (1996). A passivity-based approach to force regulation and motion control of robot manipulators. *Automatica*, 32(3):443 – 447.

[Siciliano and Villani, 2000] Siciliano, B. and Villani, L. (2000). *Robot Force Control*. Kluwer Academic Publishers, Norwell, MA, USA.

[Simmons and Demiris, 2005] Simmons, G. and Demiris, Y. (2005). Optimal robot arm control using the minimum variance model. *Journal of Robotic Systems*, 22(11):677–690.

[Sisbot et al., 2010] Sisbot, E., Marin-Urias, L., Broqure, X., Sidobre, D., and Alami, R. (2010). Synthesizing robot motions adapted to human presence. *Int. Journal of Social Robotics*, 2:329–343.

[Spall, 2003] Spall, J. (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, Hoboken, NJ.

[Sugihara, 2009] Sugihara, T. (2009). Solvability-unconcerned inverse kinematics based on levenberg-marquardt method with robust damping. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 555–560, Paris, France.

[Sun et al., 2002] Sun, F., Li, H., and Li, L. (2002). Robot discrete adaptive control based on dynamic inversion using dynamical neural networks. *Automatica*, 38:1977 – 1983.

[Suykens et al., 2002] Suykens, J., Gestel, T. V., Brabanter, J. D., Moor, B. D., and Vandewalle, J. (2002). *Least Squares Support Vector Machines*. World Scientific Publishing Co. Pte Ltd., Singapore.

[Tanaka et al., 2006] Tanaka, H., Krakauer, J., and Qian, N. (2006). An optimization principle for determining movement duration. *J Neurophysiol*, 95:3875–3886.

[Tlalolini et al., 2011] Tlalolini, D., Chevallereau, C., and Aoustin, Y. (2011). Human-like walking: Optimal motion of a bipedal robot with toe-rotation motion. *IEEE/ASME Transactions on Mechatronics*, 16(2):310–320.

[Todorov, 2004] Todorov, E. (2004). Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):907–915.

[Todorov, 2005] Todorov, E. (2005). Stochastic optimal control and estimationmethods adapted to the noise characteristics of the sensorimotor system. *Neural Computation*, 17:1084–1108.

[Todorov and Jordan, 2002] Todorov, E. and Jordan, M. (2002). Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235.

[Todorov and Li, 2005] Todorov, E. and Li, W. (2005). A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Americal Control Conference*, pages 300–306.

[Tsagarakis et al., 2007] Tsagarakis, N., Metta, G., Sandini, G., Vernon, D., Beira, R., Santos-Victor, J., Carrazzo, M., Becchi, F., and Caldwell, D. (2007). iCub - the design and realization of an open humanoid platform for cognitive and neuroscience research. *Journal of Advanced Robotics*, 21(10):1151–1175.

[Tsagarakis et al., 2009] Tsagarakis, N. G., Vanderborght, B., Laffranchi, M., and Caldwell, D. G. (2009). The mechanical design of the new lower body for the child humanoid robot 'icub'. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4962–4968, St. Louis, MO, USA.

[Tuan et al., 2008] Tuan, T., Souères, P., Taïx, M., and Guigon, E. (2008). A principled approach to biological motor control for generating humanoid robot reaching movements. In *IEEE Int. Conf. Biomedical Robotics and Biomechatronics*, pages 783–788.

[Uno et al., 1989] Uno, Y., Kawato, M., and Suzuki, R. (1989). Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61:89–101.

[Vernon, 2010] Vernon, D. (2010). Enaction as a conceptual framework for developmental cognitive robotics. *Paladyn Journal of Behavioral Robotics*, 1:89–98.

[Vernon et al., 2007a] Vernon, D., Metta, G., and Sandini, G. (2007a). The icub cognitive architecture: Interactive development in a humanoid robot. In *Proc. IEEE 6th International Conference on Development and Learning ICDL 2007*, pages 122–127.

[Vernon et al., 2007b] Vernon, D., Metta, G., and Sandini, G. (2007b). A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Transactions on Evolutionary Computation, Special Issue on Autonomous Mental Development*, 11(2):151 – 180.

[Vicon, www] Vicon (www). http://www.vicon.com/.

[Vidyasagar, 1987] Vidyasagar, M. (1987). *Control System Synthesis: A factorization approach*. MIT Press.

[Viviani, 1986] Viviani, P. (1986). *Generation and modulation of action patterns, H. Heuer & C. Fromm (Eds.)*, chapter Do units of motor action really exist?, pages 201–216. Berlin: Springer-Verlag.

[Viviani and Flash, 1995] Viviani, P. and Flash, T. (1995). Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. *Journal of Experimental Psychology: Human Perception and Performance*, 21:32–53.

[Viviani and Stucchi, 1992] Viviani, P. and Stucchi, N. (1992). Biological movements look constant: Evidence of motor perceptual interactions. *Journal of Experimental Psychology: Human Perception and Performance*, 18:603–623.

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. (2006). On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57.

[Wada et al., 2001] Wada, Y., Kaneko, Y., Nakano, E., Osu, R., and Kawato, M. (2001). Quantitative examinations for multi joint arm trajectory planning–using a robust calculation algorithm of the minimum commanded torque change trajectory. *Neural Networks*, 14(4-5):381–393.

[Whitman and Atkeson, 2009] Whitman, E. and Atkeson, C. (2009). Control of a walking biped using a combination of simple policies. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 520–527, Paris, France.

[Wittenburg, 1994] Wittenburg, J. (1994). Topological description of articulated systems. *Computer-Aided Analysis of Rigid and Flexible Mechanical Systems*, pages 159–196.

[Wolpert et al., 1998] Wolpert, D., Miall, R., and Kawato, M. (1998). Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347.

[Wolpert et al., 1995] Wolpert, D. M., Ghahramani, Z., and Jordan, M. I. (1995). Are arm trajectories planned in kinematic or dynamic coordinates? an adaptation study. *Experimental Brain Research*, 103:460–470.

[Xsens, www] Xsens (www). *The MTx orientation tracker*. `http://www.xsens.com/en/general/mtx`.

[Zhao and Chen, 1996] Zhao, H. and Chen, D. (1996). Optimal motion planning for flexible space robots. In *IEEE Int. Conf. on Robotics and Automation*, pages 393–398, Minneapolis, Minnesota, USA.

[Zoppoli et al., 2001] Zoppoli, R., Sanguineti, M., and Parisini, T. (2001). Can we cope with the curse of dimensionality in optimal control by using neural approximators? In *Proc. 40th IEEE Conference on Decision and Control*, volume 4, pages 3540–3545.

[Zoppoli et al., 2002] Zoppoli, R., Sanguineti, M., and Parisini, T. (2002). Approximating networks and extended ritz method for the solution of functional optimization problems. *Journal of Optimization Theory and Applications*, 112:403–439.

[Zoppoli et al., 2011] Zoppoli, R., Sanguineti, M., Parisini, T., and Baglietto, M. (2011). *Neural Approximations for Optimal Control and Decision*. Control and Communications Systems Series. Springer-Verlag.