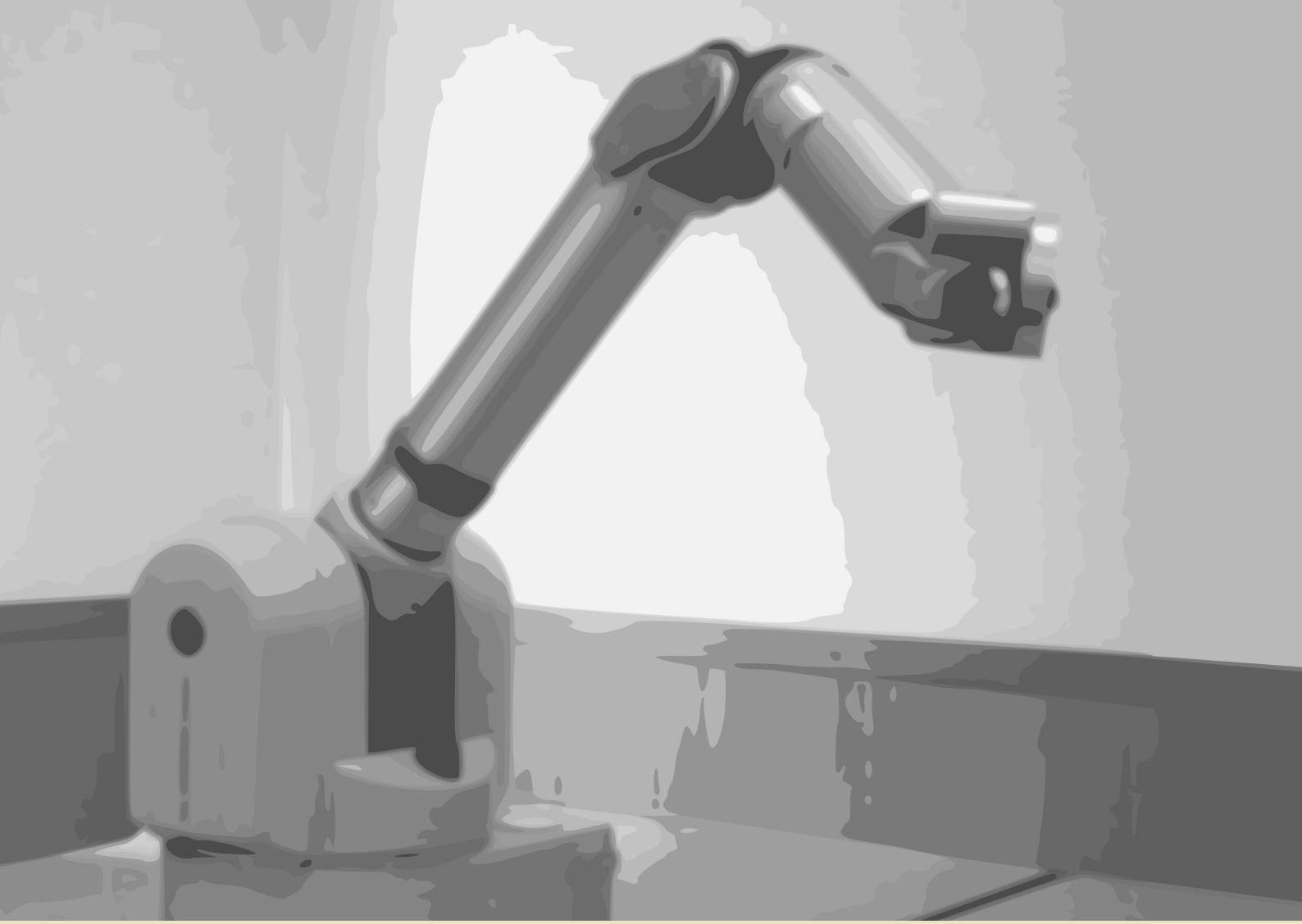

Cristóvão Jorge da Silva Duarte e Sousa

DYNAMIC MODEL IDENTIFICATION OF ROBOT MANIPULATORS: SOLVING THE PHYSICAL FEASIBILITY PROBLEM

Tese de doutoramento em Engenharia Electrotécnica e de Computadores, Ramo de Especialização em Automação e Robótica, orientada pelo Senhor Professor Doutor Rui Pedro Duarte Cortesão e apresentada ao Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Setembro de 2014

· U C ·

Universidade de Coimbra

UNIVERSITY OF COIMBRA

Dynamic model identification of robot manipulators: Solving the physical feasibility problem

Ph.D. Thesis

Cristóvão Jorge Silva Duarte Sousa

September 2014

Advisor: Prof. Rui Pedro Duarte Cortesão, University of Coimbra

Dynamic model identification of robot manipulators: Solving the physical feasibility problem

Cristóvão Jorge Silva Duarte Sousa

September 2014

ABSTRACT

Nowadays, novel robotic applications are appearing, where humans and robots collaborate in multiple tasks, ranging from house hold duties to surgical interventions. These applications deal with complex human and environment interactions, requiring advanced modeling and control techniques to boost performance. The identification of robot dynamics, which includes the dynamic parameters, is a key issue for model-based controllers, having also a key role in realistic robot simulation. The dynamic parameters are constants related to robot link inertias, joint frictions, and other dynamic aspects. In order to identify the dynamic parameters, regression methods based on recorded position and force based data are usually the only viable option. However, there is a problem which often appears in the regression of dynamic parameters: it is the so-called physical feasibility issue. Robot dynamic parameters have physical meaning, they represent physical quantities, and thereby the parameter estimations must correspond to physically feasible values. If the parameters are not feasible, i.e., not consistent from the physical point of view, then they will render an unrealistic and unstable dynamic model. A simple example of a not physically feasible parameter is a mass with negative value: its use in simulation or in a model-based controller entails positive feedback and divergent behavior. It is straightforward to verify whether a complete set of standard dynamic parameters comply with the physical feasibility conditions. Nevertheless, when identifying dynamic parameters though regression, it is only possible to identify linear combinations of parameters, the so-called base parameters. Moreover, the direct feasibility verification method for the standard parameters is not applicable to the base parameters. Although physically infeasible estimations are intrinsically unstable, there are cases where they are so close to the feasible region that the effects can remain hidden until the robot preforms specific motions entailing dangerous behaviors. An effective and efficient method for base parameter feasibility test had yet not been devised, hence posing an open problem.

This thesis presents a novel approach that describes the physical feasibility conditions as a linear matrix inequality (LMI). This approach enables easy representation of the feasible region either using the standard or the base parameter space. With this representation it is possible to devise methods for feasibility test and correction, rooted in recent mathematical formulations and tools related to semidefinite programming (SDP). Furthermore, a novel method which finds the optimal feasible parameter solution that best fits a given regression data set is also presented. These methods are experimentally tested in the real case of the parameter identification of a WAM robot, a seven-link manipulator widely used in applications that require both motion and contact. The

manipulator model is devised from software tools developed within this work, which also implement feasibility methods. The software stack is available for free and is completely open, relying only on open-source languages and libraries. Experimental validation has shown to be effective, and the physically feasible parameter identification of the 7-link WAM robot is absolutely necessary when tested in simulation and model-based control applications. This novel approach is compared with previous works addressing feasibility issues. Although previous works solve part of the problem, all of them present several drawbacks when compared to the methods presented here. The approach based on linear matrix inequalities and semidefinite programming addresses the problem from an elegant mathematical perspective. The methods provide optimal solutions and are very efficient, even when the problem size scales up.

RESUMO

Presentemente, estão a surgir novas aplicações robóticas, onde humanos e robôs colaboram em múltiplas tarefas, que vão desde trabalhos domésticos a intervenções cirúrgicas. Estas aplicações lidam com complexas interacções com o ambiente, e requerem técnicas avançadas de modelação e controlo para melhor resultados. A identificação da dinâmica de robôs, que inclui os parâmetros dinâmicos, é uma peça fundamental para os controladores baseados no modelo, e é também fundamental para a simulação realística de robôs. Os parâmetros dinâmicos são constantes relacionadas com a inercia dos elos do robô, fricções das juntas e outros aspectos dinâmicos. Para se poder identificar os parâmetros dinâmicos, métodos de regressão baseados em dados de posição e força são normalmente a única opção viável. No entanto, há um problema que aparece muitas vezes na regressão do parâmetros dinâmicos: é o problema da factibilidade física. Os parâmetros dinâmicos dos robôs têm significado físico, representam quantidades físicas, e por isso os parâmetros estimados devem corresponder a valores fisicamente fazíveis. Se os parâmetros não forem fazíveis, isto é, não consistentes de um ponto de vista físico, então eles irão tornar o modelo dinâmico irrealista e instável. Um exemplo simples de um parâmetro fisicamente não fazível é o de uma massa com valor negativo: o seu uso em simulação ou num controlador baseado no modelo implica realimentação positiva e um comportamento divergente. É possível verificar directamente se um conjunto completo de parâmetros dinâmicos obedece às condições de factibilidade física. No entanto, quando se identificam os parâmetros dinâmicos através de regressão, é apenas possível identificar combinações lineares dos mesmos, os chamados parâmetros base. Além disso, o método de verificação de factibilidade dos parâmetros completos não é aplicável aos os parâmetros base. Embora as estimações fisicamente não fazíveis sejam intrinsecamente instáveis, há casos em que elas estão tão próximas da região de factibilidade que os efeitos permanecem escondidos até que o robô faça determinados movimentos que levam a comportamentos perigosos. Um método efectivo e eficiente para testar a factibilidade de parâmetros base não tinha sido ainda concebido, sendo por isso um problema em aberto.

Esta tese apresenta uma nova abordagem que descreve as condições de factibilidade física através de uma inequação linear de matrizes. Tal abordagem permite a fácil representação da região fazível usando o espaço dos parâmetros completos ou dos parâmetros base. Com esta representação é possível conceber métodos para teste e correcção de factibilidade, baseados em formulações e ferramentas matemáticas recentes, relacionadas com programação semidefinida. Além disso, é também apresentado um novo método que encontra a solução óptima dos parâmetros fazíveis que melhor se adaptam a um dado conjunto de dados de

regressão. Este métodos são testados experimentalmente no caso real da identificação dos parâmetros de um robô WAM, um manipulador com sete juntas, muito usado em aplicações que requerem movimentação e contacto. O modelo do manipulador é obtido com ferramentas de software desenvolvidas neste trabalho, que também incluem métodos de factibilidade. Tal software está disponível gratuitamente, e é completamente aberto, sendo baseado apenas em linguagens e bibliotecas abertas. A validação experimental mostra ser efectiva, e a identificação fisicamente fazível dos parâmetros do robô WAM é absolutamente necessária quando testada em aplicações de simulação e de controlo baseado no modelo. Esta nova abordagem é comparada com trabalhos anteriores que também visam problemas de factibilidade. Embora os trabalhos anteriores tenham resolvido parte do problema, todos eles apresentam várias desvantagens quando comparados com os métodos aqui apresentados. A abordagem baseada em inequações lineares de matrizes e programação semidefinida faz uma aproximação ao problema de uma perspectiva matemática elegante. Os métodos fornecem soluções óptimas e são bastante eficientes, mesmo quando o tamanho do problema cresce.

AGRADECIMENTOS

Em primeiro lugar, quero mostrar o meu maior agradecimento ao meu orientador, o Professor Doutor Rui Cortesão. Agradeço-lhe pelo suporte e orientação nestes anos de doutoramento, assim como pelos anos anteriores, durante a licenciatura, o mestrado e a bolsa de investigação. Agradeço, sobretudo, pela motivação que soube transmitir, por mostrar que o caminho académico se faz de altos e baixos; agradeço por sempre dizer que só não falha quem não tenta. Agradeço a sua dedicação constante, e por ser um orientador presente mas que ao mesmo tempo me deu a liberdade de seguir a minha linha de investigação. Agradeço-lho toda a orientação e ajuda no meu trabalho. Pelo seu profissionalismo, e também pela sua amizade, o meu maior obrigado.

Aos meus colegas de doutoramento. Obrigado pelo bom ambiente, pelas discussões de ideias, pela amizade. Um obrigado especial ao Luís Santos, à Fernanda Coutinho e ao Michel Dominici pela partilha deste caminho. Ao Michel, obrigado também por me receber na sua casa em Montpellier e me possibilitar contactar com outros trabalhos na área da robótica médica. Agradeço a Alex Becker e a *BS.* pelas suas preciosas repostas no sítio MathOverflow.net que me permitiram reformular o problema investigado nas áreas matemáticas adequadas e abriram caminho a novos desenvolvimentos. Agradeço a toda a comunidade de software aberto, que me possibilitou ter ferramentas da mais alta qualidade para fazer o meu trabalho. A toda a comunidade científica e técnica do campo da robótica, obrigado pelos "ombros de gigantes".

Agradeço ao Instituto de Sistemas e Robótica de Coimbra, que me acolheu e me proporcionou os meios para realizar a minha investigação. À Universidade de Coimbra, à Faculdade de Ciências e Tecnologia, ao Departamento de Engenharia Electrotécnica e de Computadores, e a todos aqueles que foram meus professores, agradeço a formação, a investigação e desenvolvimento de qualidade e prestigio que fomentam. Agradeço à Fundação para a Ciência e a Tecnologia pelo suporte ao meu trabalho através do projecto PTDC/EEA-CRO/110008/2009 e da bolsa de doutoramento SFRH/BD/61553/2009.

No espaço da vida pessoal, agradeço a toda a minha família pelo amor que me têm dado e pelo suporte sem o qual eu não

poderia fazer este caminho. Em particular agradeço aos meus pais, por tudo o que deixaram de ter e fazer para eles, por mim. Ao meu pai António devo aquele gosto especial por máquinas, por ferramentas, por processos, controladores, por arranjar coisas, pela invenção, enfim, tudo aquilo que é, na verdade, engenharia. À minha mãe Maria Helena devo o gosto pelo conhecimento, pela sabedoria, não só intelectual, mas sobretudo pessoal e espiritual, e agradeço por sempre me mostrar que a Deus tudo devo. Agradeço aos meus familiares mais próximos, os meus tios Carlos e Ilda, as minhas primas e primos, e a minha tia Encarnação. Agradeço à minha sogra Ângela, agradeço ao Tó, ao Joca e à Tita. Agradeço a todos os meus amigos, sobretudo os que me ajudaram quando precisei de tempo para trabalhar. Ao Jorge e à Joana, à Vânia e ao Eduardo, à Maria João e família, ao João Marcos, obrigado. À minha esposa, Cláudia, obrigado por tudo.

Cristóvão Duarte Sousa
Setembro 2014

Dedico esta tese aos meus pais, António e Maria Helena,
à minha esposa, Cláudia,
e às minhas filhas, nascidas durante os anos de doutoramento,
Madalena, Leonor e Alice.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AOB active observer.

CAD computer-aided design.

CADec cylindrical algebraic decomposition.

COM center of mass.

DH Denavit–Hartenberg.

DOF degrees of freedom.

EL Euler–Lagrange.

LMI linear matrix inequality.

LS least squares.

ODE ordinary differential equation.

OLS ordinary least squares.

PD proportional–derivative.

QP quadratic programming.

RNE recursive Newton–Euler.

SDP semidefinite programming.

SQP sequential quadratic programming.

SVD singular value decomposition.

WLS weighted least squares.

NOTATION

$\boldsymbol{A} \succ 0$	$\boldsymbol{A}$ is positive definite	
$\boldsymbol{A} \succeq 0$	$\boldsymbol{A}$ is positive semidefinite	
$\dot{a}$	first derivative of a in order to time	
$\ddot{a}$	second derivative of a in order to time	
$\boldsymbol{A}^{-1}$	inverse of $\boldsymbol{A}$	
$\boldsymbol{A}^{T}$	transpose of $\boldsymbol{A}$	
$\emptyset$	empty set	
$\mathbf{0}_{m \times n}$	zero matrix of size $m \times n$	
$\mathbf{1}_{m}$	identity matrix of size m	
$\boldsymbol{\beta}$	base dynamic parameters	(2.53)
$\mathcal{B}$	set of physically feasible base dynamic parameters	(2.67), (3.16), (3.33)
$\boldsymbol{C}$	Coriolis and centripetal forces matrix	(2.7)
$\boldsymbol{c}$	Coriolis and centripetal forces	(2.8, 2.9)
$\boldsymbol{D}$	matrix from the linear matrix inequality (LMI) of the feasible standard parameter set	(3.25)
$\bar{\boldsymbol{D}}$	matrix from the non-strict LMI of the feasible standard parameter set	(3.29)
$\boldsymbol{\delta}$	standard (barycentric) dynamic parameters	(2.20)
$\boldsymbol{\delta}_{\mathrm{A}}$	additional dynamic parameters	(2.19)
$\boldsymbol{\delta}_{\mathrm{b}}$	independent standard dynamic parameters	(2.49)
$\boldsymbol{\delta}_{\mathrm{d}}$	dependent standard dynamic parameters	(2.49)
$\boldsymbol{\delta}_{\mathrm{L}}$	link inertial dynamic parameters	(2.18)
$\mathcal{D}$	set of physically feasible standard dynamic parameters	(2.65), (3.30)
$\boldsymbol{D}_{\mathrm{A}}$	matrix from the LMI of the additional dynamic parameter feasibility conditions	(3.23)
$\mathcal{D}_{\mathrm{A}}$	set of physically feasible additional dynamic parameters	(2.64)

$\bar{\boldsymbol{D}}_\beta$	matrix from the non-strict LMI of the feasible base–dependent parameter set	(3.32)
$\mathcal{D}_\beta$	set of physically feasible base–dependent dynamic parameters	(3.31)
Diag$(\cdot)$	function to create a diagonal matrix with the vector argument elements	
diag$(\cdot)$	function to extract the diagonal elements from the argument matrix	
$\boldsymbol{D}_\mathrm{L}$	generalized inertia matrix; matrix from the LMI of inertial parameter feasibility conditions	(3.20)
$\mathcal{D}_\mathrm{L}$	set of physically feasible link inertial parameters	(2.63)
$\boldsymbol{\epsilon}$	vector of regression/prediction residuals	(2.44), (2.56)
F_c	Coulomb friction constant	(2.19)
F_o	Coulomb friction offset (can include motor torque offset too)	(2.19)
F_v	viscous friction constant	(2.19)
$\boldsymbol{g}$	gravity force term	(2.7)
$\boldsymbol{h}$	non-conservative forces affecting robot dynamics	(2.9)
$\boldsymbol{H}$	regressor matrix function for the dynamics	(2.22)
$\boldsymbol{H}_\mathrm{b}$	independent parameter regressor function	(2.49)
$\boldsymbol{H}_\mathrm{d}$	dependent parameter regressor function	(2.47), (2.49)
$\boldsymbol{H}_S$	regression matrix	(2.40)
$\boldsymbol{I}$	inertia tensor about the center of mass	(2.12)
I_a	motor rotor inertia	(2.19)
$\boldsymbol{J}$	Jacobian relating joint and task space velocities	(2.2, 2.3)
k	link/joint index	

$\boldsymbol{K}$	standard to base parameter combination matrix	(2.54)
$\boldsymbol{K}_\mathrm{d}$	matrix of linear dependencies of dependent parameters with respect to independent parameters	(2.47)
$\boldsymbol{l}$	first moment of inertia	(2.13)
$\boldsymbol{L}$	inertia tensor about the link frame	(2.14)
m	mass	
$\mathfrak{m}$	linear mapping from standard to base parameters	(3.9)
$\mathfrak{m}_p$	projection from base–dependent to base parameters	(3.12)
$\mathfrak{m}_t$	bijective mapping from standard to base–dependent parameters	(3.11)
$\boldsymbol{M}$	inertia matrix	(2.7)
$\boldsymbol{\mu}$	end-effector forces in task space	(2.3)
$\boldsymbol{\mu}_\mathrm{c}$	joint actuator forces seen in task space	(2.29)
$\boldsymbol{\mu}_\mathrm{e}$	task space environment forces	(2.29)
N	robot number of links/joints; the number of degrees of freedom (DOF)	
n	total number of dynamic parameters	
n_b	number of base dynamic parameters; number of independent dynamic parameters	(2.46)
n_d	number of dependent dynamic parameters	(2.46)
$\boldsymbol{\omega}$	regression regressand vector	(2.41)
$\boldsymbol{P}$	permutation matrix for dynamic parameter ordering	(2.48)
$\boldsymbol{P}_\mathrm{b}$	independent parameter slice of permutation matrix	(2.48)
$\boldsymbol{P}_\mathrm{d}$	dependent parameter slice of permutation matrix	(2.48)

$\boldsymbol{q}$	joint positions	(2.4)
$\dot{\boldsymbol{q}}$	joint velocities	(2.4)
$\ddot{\boldsymbol{q}}$	joint accelerations	(2.4)
$\dot{\boldsymbol{q}}_\mathrm{m}$	motor velocities	(4.2)
$\mathbb{R}$	set of real numbers	
$\boldsymbol{R}_1$	base reduced regression matrix	(3.46)
$\boldsymbol{\rho}_1$	reduced regressand vector	(3.49)
$\boldsymbol{r}$	center of mass position (relative to link frame)	(2.11)
S	number of excitation trajectory measurements	
$\mathrm{sign}(\cdot)$	signal function	
$\mathbf{S}(\cdot)$	skew-symmetric matrix operator	(2.17)
$\boldsymbol{T}$	joint-motor coupling matrix	(4.1, 4.2)
$\boldsymbol{\tau}$	generic joint forces	(2.4)
$\boldsymbol{\tau}_\mathrm{c}$	actuator joint forces	(2.9)
$\boldsymbol{\tau}_\mathrm{e}$	environment contact forces seen in joint space	(2.9)
$\boldsymbol{\tau}_\mathrm{m}$	motor forces	(4.1)
t	time	
$\boldsymbol{U}$	matrices from the LMIs of least squares (LS) minimizations	(3.39), (3.43), (3.52), (3.57)
$\mathcal{V}_{\hat{\boldsymbol{\beta}}}$	virtual parameter set of a base parameter vector $\hat{\boldsymbol{\beta}}$	(2.60)
$\boldsymbol{W}$	base regression matrix	(2.57)
$\boldsymbol{x}$	end-effector position in task space	(2.1)

Essentially, all models are wrong, but some are useful.

— George E. P. Box

1 INTRODUCTION

1.1 Motivation

Well established in industry contexts, manipulation robots have been slowly entering new application domains in the last decades. Although the first thought of a manipulator robot may be an image of a bulky robot working in an assembly line of some factory, far from any human contact, the truth is that they are already much closer than that. In fact, robots are already being used close to humans, even inside them, in surgical applications.

The present work is in the context of a research group focused on medical robotic applications where advanced dynamic robot control for human–robot contact is essential. In industrial applications, robots often perform repetitive tasks, with predefined trajectories, both in space and time, and high precision. On the other hand, robots interacting with persons, as in medical applications, have to perform much less structured tasks, usually having humans not only in the task space but also in the control loop. Actual and potential medical robot applications include surgical procedures, rehabilitation, tele-echography and other remote procedures. Robots present a huge potential for these tasks. They are not meant to completely replace physicians, as many repetitive tasks were replaced by robots in industry, but rather to enhance and extend their skills. The best capabilities both from humans and robots are merged together creating a system with advantages for physicians and ultimately for patients. On one hand, robots are great at geometrical accuracy and fine movements, and they are able to work endlessly. Humans, on the other hand, have much higher capacity to take decisions, and to learn and deal with unpredictable environments. Medical robots must be seen as "smart" instruments which enlarge medical procedures effectiveness (Taylor 2006). Moreover, robots also have the potential to physically decouple the human operator from the task, enabling humans to perform remote operations.

A classic example of medical assistance robot is the *da Vinci* system[1], used for minimally invasive surgeries where medical tools enter into the patient body through small incisions. In such systems, the physician controls the

[1] *da Vinci Surgical System*: http://www.davincisurgery.com/.

robot that holds surgical instruments. This approach helps mitigating some of the major drawbacks of minimally invasive surgical procedures (Palep 2009). The surgeon is at a master station, sending commands to the slave robot from where he receives visual feedback. One of the most appointed disadvantages of this kind of systems, however, has been the lack of haptic feedback, i.e., force feedback, and thus this has been an emerging field with growing number of research works (van der Meijden and Schijven 2009). The introduction of force feedback in the human–robot loop dramatically changes the system dynamics. Although the visual feedback also closes the loop, the motion–force loop create a bilateral tele-manipulation scheme, introducing new big challenges to the system dynamics modeling and control. Robot, operator, environment, master station and transmission channel, all those system sub-parts have to be carefully studied and modeled. Moreover, force feedback is a key research not only in robotic-assisted surgery but in general tele-manipulation and other medical fields too. For example, robot-assisted tele-echography, i.e., echographic exams executed by a remotely commanded robot (Sousa et al. 2010; Santos and Cortesão 2014), can take advantage of force feedback. The intrinsic nature of master–slave systems enable remote manipulation through the physical separation of master and slave stations. In surgery, such possibility was realized for the first time on an intercontinental operation, which became known as the Lindbergh Operation (Marescaux et al. 2001), using an adapted *da Vinci* surgery system.

While robot dynamic modeling is fundamental for tele-manipulation control, it is also a key issue in other control schemes, both medical and non-medical related. For example, it is essential for co-manipulation, where robot and human operator share the control of the end-effector[2] (Lamy et al. 2009; Cortesão et al. 2010). It has also been more and more important in industry applications, where new high-speed and compliance-force requirements are becoming important.

1.1.1 Dynamic model-based control and simulation

Initially, the control of robots was a matter of moving the end-effector to a desired position, following predetermined trajectories. This is accomplished by the knowledge of the geometric model which relates robot joint positions

[2]The robot tip; the tool performing the task.

(can be angles) with the end-effector position in the environment. The geometric model can be derived to obtain the kinematic model, which relates velocities in joint and task spaces. These models enable the design of motions with controlled end-effector position and velocity, often used in pick-and-place tasks, and are only dependent on the robot geometry, i.e., the way robot links are positioned relative to each other.

For more advanced tasks, with higher motion dynamics and unstructured contacts with the environment, dynamic model knowledge and dynamic model-based controllers are needed. The dynamic model of a robot relates joint accelerations to forces, which is an extension of Newton's second law (and Euler's laws of motion),

$$\text{force} = \text{mass} \times \text{acceleration} ,$$

to the system formed by robot links. Force and acceleration are vectors, with a value for each joint, and the mass is a matrix, the inertia matrix. These terms are non-linear functions dependent on robot geometry and posture, i.e., joint positions at a given time. The inertia matrix is also dependent on robot dynamic parameters: masses, centers of mass, and inertia tensors. The forces, also dependent on dynamic parameters, refer to both translational forces and rotational torques, and include robot actuator forces, environment contact forces and other forces like weight and frictions. The knowledge of the dynamic model enables advanced control techniques such as

computed torque where the robot low-level command is done in force or torque;

resolved acceleration where a trajectory can be designed and controlled by the desired acceleration;

feedback linearization where the non-linear terms of the model are canceled by non-linear feedback computed torque, entailing a linear plant for control design;

gravity compensation an example of feedback linearization, where the gravity forces actuating on the robots are canceled by commanding equal but symmetric forces to the joint actuators;

impedance shaping where the robot stiffness, damping and inertia felt by an external interactive agent is imposed by the robot controller.

Model-based control allows advanced robot applications necessary in medical fields (bilateral tele-manipulation, co-manipulation, etc.) and also in new industrial fields. Furthermore, the dynamic model is the cornerstone of robot simulation. The simulation of the robot behavior is very important in control and application design phases as it enables safe and early testing. For applications with a human in the control loop, simulation enables safe training with higher system availability. That is specially relevant for medical applications. Additionally, robot simulation can be important even for application where no model-based control is used.

The quality of model-based control and simulation depends directly on the accuracy of the available robot model, hence the need for good modeling techniques.

1.2 Problem definition

A robot can be modeled employing non-parametric and semi-parametric models (for example, Peters (2010)). However, as a robot is a structured system of links, fully parametric models are much more common in article literature, as well as in robotics textbooks (for instance, Yoshikawa (1990), Murray et al. (1994), Siciliano and Villani (1999), Khalil and Dombre (2002), Craig (2004), Spong et al. (2005), Siciliano and Khatib (2008), and Siciliano et al. (2009)). The dynamic model of a robot can be described by expanding the equation presented above into

$$(\text{inertia} \times \text{acceleration}) - \text{internal forces} = \text{actuator forces} + \text{external forces}\ .$$

Actuator forces are the forces produced by the motors or other actuation mechanism which are commanded by the controller. External forces come from the robot–environment contact. Internal forces are due to joint frictions, weight and other link-to-link forces. Robot inertia and internal forces are dependent on robot posture, as well as on dynamic parameters.

Model formulation in terms of equations is a well studied subject. Two main approaches for equation derivation exist: the Euler–Lagrange (EL) formulation, based on the derivation of the system energy as a whole, and the recursive Newton–Euler (RNE) formulation, based on the application of Newton-Euler equations to link interactions. While being equivalent, the RNE formulation has easier computational implementation since it includes no differ-

ential terms and has a direct relation to programming language loops. Although generic RNE implementations run on current computers with enough performance, in the late eighties there was the need to generate smaller and faster custom code for each robot. Such code is obtained by running the formulations within symbolic algebra systems and taking advantage of each robot particularities to generate code with less operations. These optimizations are still very important nowadays in applications where the dynamic model has to be computed in real-time, several times per cycle. Currently, there are some implementations of robot custom code generation available, however, all of them rely on proprietary and closed software.

Besides model equations, dynamic model parameters must also be known. These parameters are constant and must be identified (estimation of constants) to fully model the robot. Among few others, the usual process for parameter identification refers to regression techniques applied to force and position based data. By measuring a large number of motion points — joint positions, velocities and accelerations — and the respective actuator and environment forces, it is possible to infer the dynamic parameters which produce such data set. Although being composed by large expressions, the robot dynamic model is linear with respect to the dynamic parameters, thus allowing simple linear regression techniques like the classical ordinary least squares (OLS). On the other hand, it is usually not possible to identify all parameters, and many of them can only be obtained in linear combinations, leading to the so-called base parameters. As in any estimation process, the error from measurements and errors in modelling propagate to the dynamic parameter estimation. There is a countless number of parameter identification techniques which aim at reducing such error, be it by employing better ways to collect the data set, be it by using regression techniques more meaningful in terms of statistics theory.

Often, measurement and model errors lead to what is know as physically infeasible, or physically inconsistent, parameter identifications. These estimations, which were studied in deep for the first time by Yoshida and Khalil (2000), are comparable to a negative mass, i.e., they are not possible in a real physical system. The arise of this kind of issues in robot model identification is not necessarily associated with high regression error, and many times they can appear in well modeled robots with low optimal regression error. One of the major problems of mitigating infeasible estimations is that while it is

straightforward to check the physical feasibility of a complete set of all dynamics parameters, that is far from being true for the identifiable parameters (the base parameters). The effect of using infeasible estimations in model-based controllers or in simulation is comparable to the effect of a negative mass in the Newton's second law: they lead to intrinsically unstable models. Furthermore, in robot modeling, the effects of theses problematic estimations are not always obvious, and may show up only at few robot postures, leading to potentially dangerous applications. Although more accurate models and more accurate regression data are more likely to provide physically feasible estimations, that is not by itself a guarantee of feasibility. Although there have been some works in this field, proposed solutions tend to be not effective or not efficient. There is a lack for a common framework, effective and efficient, and with simple implementation to deal with robot parameter feasibility checking and identification. This problem is addressed in this thesis.

1.3 Objectives

Quoting Box and Draper (1987, 74),

> "Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful."

In the case of robot modeling, while an estimation can entail very low regression and prediction errors, if it is physically infeasible it can be considered completely useless for model-based control and simulation applications. In this work we seek to study the physical feasibility of the parameters, trying to answer the following questions:

1. Is it possible to tell solely from a given base parameter estimation if it is physically feasible or not?

2. If so, how to efficiently and effectively check the feasibility of any given estimation?

3. How to obtain an estimation from a regression data set that is guaranteed to be physically feasible and, at the same time, minimizes the regression error?

Recalling to the quote of Box and Draper, it must be stressed out that this work is not about quantitative results, but rather about qualitative ones. Furthermore, as this work integrates a research group which uses a WAM™ robot, it is also an objective to model and identify it, taking physical feasibility issues into account.

1.4 Contributions

This thesis presents a novel theory for dynamic parameter identification taking into account physical feasibility. Key contributions address the questions stated above. This work picks from recent mathematical theory and control tools and combines them to solve the physical feasibility problem in an unprecedented way. A complete framework based on linear matrix inequality (LMI) and semidefinite programming (SDP) is proposed to deal with the feasibility issue, and three LMI–SDP main methods are devised: a method for base parameter feasibility test, a method for base parameters feasibility correction, and a method for optimal and feasible base parameter estimation. Those methods show to be very effective and efficient. Moreover, an elaborated model of the WAM robot is devised and a feasible set of dynamic parameters is identified. All the robot model and identification software is made from ground up based only on open source software. The modeling software employs custom code generation techniques, providing very efficient code, on par with other solutions. Since it is based only on open source, the developed software is the first, up to this date, to be completely free and openly available.

1.5 Thesis structure

The thesis is organized as follows. Chapter 2 introduces the robot modeling fundamentals. The dynamic model is formulated and the dynamic parameters are presented along classical techniques for their identification. The physical feasibility problem is introduced and analyzed, and state of the art works on the subject are presented. The main theoretical contribution of the thesis is in chapter 3. There, the structure of physical feasibility conditions is explored and properly identified in terms of mathematical nomenclature, first as semialgebraic sets and second as LMIs. Then, SDP methods are de-

vised to deal with the problems of feasibility testing and guaranteed feasible parameter identification. In chapter 4, the complete identification process of the WAM robot is presented. The proposed LMI–SDP methods are applied to identification and compared to the classical approach. Chapter 5 presents real applications of the identified WAM model, showing the importance of physically feasible estimations. Further discussion on the importance of the LMI–SDP approach and on its advantages over previous works is addressed in chapter 6. Chapter 7 presents a description of the practical implementation of the robot modeling and feasibility methods, as well as an overview to the software developed within this work. Chapter 8 concludes the thesis and presents future work directions.

2 ROBOT MODELING AND IDENTIFICATION

2.1 Serial robot manipulators

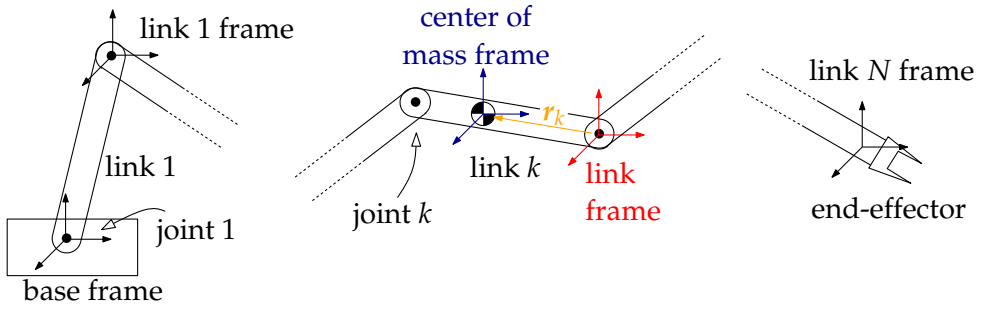

Figure 2.1: Links, joints and frames of a serial manipulator.

A serial robot manipulator is a set of N rigid bodies[1] — the so-called links — connected by movable joints which form a chain from the robot base to the robot end-effector as present in figure 2.1. To each link k, for $k = 1, \ldots, N$, there is an associated joint whose position is denoted by q_k. In serial robots, the number of links/joints usually entails the same number of degrees of freedom (DOF). Each joint is subject to a force (or torque) τ_k resulting from the sum of all forces applied by external contacts, inter links contacts and joint actuator. The ordered set of all joint positions forms the joint position vector $\boldsymbol{q}$, which fully describes the robot posture, whereas the set of joint forces (or torques) gives the joint force vector $\boldsymbol{\tau}$.

Since the robot end-effector is the effective tool which performs the task, the relation between joint positions and end-effector position in the task space forms the basic modeling problem, the geometric model. Computing the end-effector position and orientation, denoted by $\boldsymbol{x}$, from a set of joint positions values is usually done by referring to inter-link matrix transformations. Such transformation matrices are obtained from the known robot link geometry, which is often described by the Denavit–Hartenberg (DH) notation (Harten-

[1] There is also a class of robots whose links are flexible. These are not directly addressed in this thesis, but the presented formulation is extensible to them.

berg and Denavit 1955; Craig 2004). That is called the direct geometric model, a usually nonlinear but always unique transformation from $\boldsymbol{q}$ to $\boldsymbol{x}$:

$$\boldsymbol{x} = \mathbf{DirGeo}(\boldsymbol{q}) \; . \tag{2.1}$$

On the other hand, the computation of how to position the robot joints in order to obtain a desired end-effector posture usually poses a harder problem: the inverse geometry problem. While the geometric model relates positions, the relation between joint velocities, $\dot{\boldsymbol{q}}$, and end-effector velocities, $\dot{\boldsymbol{x}}$, is described by the kinematic model which is given by the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q}) \, \dot{\boldsymbol{q}} \; . \tag{2.2}$$

Due to the velocity–force duality, the Jacobian also relates the forces in the joints, $\boldsymbol{\tau}$, with the forces in the end-effector, $\boldsymbol{\mu}$, by

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q})^T \, \boldsymbol{\mu} \; . \tag{2.3}$$

Geometric and kinematic models are traditionally used in pick-and-place tasks where the controllers make the robot follow predefined position trajectories. Whenever a task requires higly unpredictable trajectories or higher accelerations, a dynamic model-based controller is additionally needed.

2.2 The dynamic model

The dynamic model of a robot relates the full motion of its joints — positions, velocities and accelerations ($\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$) — with the forces ($\boldsymbol{\tau}$) being applied to those joints.

The relation can be written as an inverse dynamic equation, where the joint force is defined as a function of the joint motion:

$$\mathbf{InvDyn}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{\tau} \; . \tag{2.4}$$

The dynamics is dependent on the robot geometry itself and robot dynamic parameters, which are constant. A classical way to devise the inverse dynamics equation is the Euler–Lagrange (EL) formulation, a differential equation of the system Lagrangian. The Lagrangian is the difference between the system kinetic and potential energies, E_k and E_p receptively,

$$L = E_k - E_p \; . \tag{2.5}$$

The EL differential equation relates the Lagrangian to the non-conservative joint forces, $\boldsymbol{\tau}$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right)^T - \left(\frac{\partial L}{\partial \boldsymbol{q}}\right)^T = \boldsymbol{\tau} \ . \tag{2.6}$$

It is common to split such differential equation into three terms, one for each differential order:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \ , \tag{2.7}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix (aforementioned in section §1.1), $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the Coriolis and centripetal forces matrix and $\boldsymbol{g}(\boldsymbol{q})$ is the term of forces due to gravity. The matrix $\boldsymbol{C}$ is non-unique, although the Coriolis and centripetal forces term given by

$$\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \equiv \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{2.8}$$

is. In (2.8), all forces not related to link inertia, such as drive chain forces, are included in $\boldsymbol{\tau}$. Equation (2.8) can be further split into additional terms as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{\tau}_\mathrm{c} + \boldsymbol{\tau}_\mathrm{e} \ , \tag{2.9}$$

where $\boldsymbol{\tau}_\mathrm{c}$ represents the joint actuator forces, $\boldsymbol{\tau}_\mathrm{e}$ represents the forces due to contact with the environment, and $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ represents any other forces related to additionally modeled dynamics related to other joint dynamic effects. For instance, a classical formulation of $\boldsymbol{h}$, often presented in textbooks, includes the joint friction forces as given given by

$$\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{F}_\mathrm{v}\dot{\boldsymbol{q}} + \boldsymbol{F}_\mathrm{c}\mathbf{sign}(\dot{\boldsymbol{q}}) \ , \tag{2.10}$$

where $\boldsymbol{F}_\mathrm{v}$ and $\boldsymbol{F}_\mathrm{c}$ are diagonal matrices of viscous and Coulomb friction constants, respectively. Often, motor rotor inertias are also included to the model.

2.3 Dynamic parameters

Besides robot geometry, dynamics is also dependent on the set of parameters related to link inertias and to the additional dynamic constants — the dynamic parameters. Each robot link k inertia can be fully characterized by the mass, m_k, the center of mass (COM) position relatively to the link frame (see figure 2.1),

$$\boldsymbol{r}_k \equiv \begin{bmatrix} r_{xk} \\ r_{yk} \\ r_{zk} \end{bmatrix} \ , \tag{2.11}$$

and the inertia tensor about the COM,

$$\boldsymbol{I}_k \equiv \begin{bmatrix} I_{xxk} & I_{xyk} & I_{xzk} \\ I_{xyk} & I_{yyk} & I_{yzk} \\ I_{xzk} & I_{yzk} & I_{zzk} \end{bmatrix} . \tag{2.12}$$

However, for identification purposes it is common to use the so-called barycentric parameters (Maes et al. 1989). By such formulation each link inertia is parameterized by the mass, m_k, the first moment of inertia,

$$\boldsymbol{l}_k \equiv \begin{bmatrix} l_{xk} \\ l_{yk} \\ l_{zk} \end{bmatrix} , \tag{2.13}$$

and the inertia tensor about the link frame,

$$\boldsymbol{L}_k \equiv \begin{bmatrix} L_{xxk} & L_{xyk} & L_{xzk} \\ L_{xyk} & L_{yyk} & L_{yzk} \\ L_{xzk} & L_{yzk} & L_{zzk} \end{bmatrix} . \tag{2.14}$$

The first moment of inertia is equal to the mass times the COM position vector,

$$\boldsymbol{l}_k = m_k \boldsymbol{r}_k = \begin{bmatrix} m_k r_{xk} \\ m_k r_{yk} \\ m_k r_{zk} \end{bmatrix} , \tag{2.15}$$

and the inertia tensor about the link frame is related to the tensor about the COM by the Huygens–Steiner theorem (also known as parallel axis theorem):

$$\boldsymbol{L}_k = \boldsymbol{I}_k + m_k \, \mathbf{S}(\boldsymbol{r}_k)^T \, \mathbf{S}(\boldsymbol{r}_k) , \tag{2.16}$$

where $\mathbf{S}(\cdot)$ is the skew-symmetric matrix operator,

$$\mathbf{S}(\boldsymbol{v}) \equiv \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \quad \text{for} \quad \boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} . \tag{2.17}$$

All inertial parameters of link k can then be grouped into a parameter vector $\boldsymbol{\delta}_{\mathrm{L}k}$ as

$$\boldsymbol{\delta}_{\mathrm{L}k} \equiv \begin{bmatrix} L_{xxk} & L_{xyk} & L_{xzk} & L_{yyk} & L_{yzk} & L_{zzk} & l_{xk} & l_{yk} & l_{zk} & m_k \end{bmatrix}^T . \tag{2.18}$$

Besides link inertia, robot dynamics is also dependent on the parameters related to the additionally modeled dynamics, which can include drive chain

friction and inertia constants, as mentioned above. Depending on the robot characteristics and on the modeling purposes one can opt to use more or less detailed models, with more or less parameters. Throughout this document, the set of the joint k additional dynamic parameters will be generally denoted by $\boldsymbol{\delta}_{\mathrm{A}k}$. A possible set of additional parameters related to joint k is, for example,

$$\boldsymbol{\delta}_{\mathrm{A}k} = \begin{bmatrix} F_{\mathrm{v}k} & F_{\mathrm{c}k} & F_{\mathrm{o}k} & I_{\mathrm{a}k} \end{bmatrix}^T , \tag{2.19}$$

where $F_{\mathrm{v}k}$ and $F_{\mathrm{c}k}$ are viscous and Coulomb friction constants, $F_{\mathrm{o}k}$ is a constant offset including the Coulomb friction offset and the motor torque offset (due to current offset), and $I_{\mathrm{a}k}$ is the motor rotation inertia seen by the joint.

The complete set of robot dynamic parameters, of all links, forms the *standard dynamic parameter* vector $\boldsymbol{\delta}$ given by

$$\boldsymbol{\delta} \equiv \begin{bmatrix} \boldsymbol{\delta}_{\mathrm{L}1} \\ \boldsymbol{\delta}_{\mathrm{A}1} \\ \vdots \\ \boldsymbol{\delta}_{\mathrm{L}k} \\ \boldsymbol{\delta}_{\mathrm{A}k} \\ \vdots \\ \boldsymbol{\delta}_{\mathrm{L}N} \\ \boldsymbol{\delta}_{\mathrm{A}N} \end{bmatrix} , \tag{2.20}$$

which length will be denoted by n. The equation of inverse dynamics can in fact be written as

$$\mathbf{InvDyn}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\delta}) = \boldsymbol{\tau}_{\mathrm{c}} . \tag{2.21}$$

The rational of not using $\boldsymbol{r}$ and $\boldsymbol{I}$ but rather $\boldsymbol{l}$ and $\boldsymbol{L}$ to characterize link inertias is the fact that the dynamic model is linear to the latter (Atkeson et al. 1986; Maes et al. 1989). Considering that the model is also linear to the additional dynamic parameters, which happens most of the times, the whole dynamics can be rewritten in the linear form as

$$\boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\, \boldsymbol{\delta} = \boldsymbol{\tau}_{\mathrm{c}} , \tag{2.22}$$

where the matrix $\boldsymbol{H}$, function of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$, is called the regressor.

2.4 Dynamic model-based technique examples

To better understand the usefulness of the dynamic model, some model-based techniques are presented in the sequel.

2.4.1 Model linearization by nonlinear feedback

Equation (2.9) describes a nonlinear system plant by the fact that it is nonlinearly dependent on the robot posture and velocity ($\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$). It is possible to apply linearization by nonlinear feedback, canceling nonlinear effects with a symmetric joint torque command. If the command torque $\boldsymbol{\tau}_c$ is computed as

$$\boldsymbol{\tau}_c = \hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \hat{\boldsymbol{g}}(\boldsymbol{q}) + \hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) + \boldsymbol{\tau}_c{}' \,, \tag{2.23}$$

where $\hat{\boldsymbol{c}}$, $\hat{\boldsymbol{g}}$ and $\hat{\boldsymbol{h}}$, are good estimations of their respective real values, and $\boldsymbol{\tau}_c{}'$ is the new command, the system plant becomes

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} = \boldsymbol{\tau}_e + \boldsymbol{\tau}_c{}' \,. \tag{2.24}$$

Having this, it is possible to perform resolved acceleration control, for instance, by doing

$$\boldsymbol{\tau}_c{}' = \hat{\boldsymbol{M}}(\boldsymbol{q})\, \boldsymbol{a}_c - \hat{\boldsymbol{\tau}}_e \,, \tag{2.25}$$

where $\hat{\boldsymbol{M}}$ is the inertia matrix estimation and $\hat{\boldsymbol{\tau}}_e$ is the estimation of external forces obtained, for example, from a force sensor. Vector $\boldsymbol{a}_c$ becomes the new control command and the new system plant is then given by

$$\ddot{\boldsymbol{q}} = \boldsymbol{a}_c \,, \tag{2.26}$$

which is equivalent to a simple double integrator, described in Laplace space representation (s domain) by

$$\frac{\boldsymbol{q}(s)}{\boldsymbol{a}_c(s)} = \frac{1}{s^2} \,. \tag{2.27}$$

The new plant is not only independent from the robot posture and velocity, but it is also decoupled at joint level. This technique is know as model linearization through non-linear feedback, and the computed command $\boldsymbol{\tau}_c$ is the so-called computed torque.

2.4.2 Task space

Computed torque control can also be design in task space, i.e., in a referential other than the joint space, more appropriated for task design and control. The task space is often the Cartesian space associated with the robot environment. The task space version of (2.9) is given by (from Khatib 1987)

$$\boldsymbol{A}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{c}_x(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}_x(\boldsymbol{q}) + \boldsymbol{h}_x(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\nu}_x(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\mu}_\mathrm{c} + \boldsymbol{\mu}_\mathrm{e} \ , \tag{2.28}$$

which relates to (2.9) through the following equations[2]:

$$\begin{aligned} \boldsymbol{A} &= \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1} \\ \boldsymbol{c}_x &= \boldsymbol{J}^{+T}\boldsymbol{c} \\ \boldsymbol{g}_x &= \boldsymbol{J}^{+T}\boldsymbol{g} \\ \boldsymbol{\nu}_x &= -\dot{\boldsymbol{J}}\dot{\boldsymbol{q}} \\ \boldsymbol{h}_x &= \boldsymbol{J}^{+T}\boldsymbol{h} \\ \boldsymbol{\tau}_\mathrm{e} &= \boldsymbol{J}^T\boldsymbol{\mu}_\mathrm{e} \\ \boldsymbol{\tau}_\mathrm{c} &= \boldsymbol{J}^T\boldsymbol{\mu}_\mathrm{c} \ , \end{aligned} \tag{2.29}$$

where $\boldsymbol{J}^+$ is the generic inverse of the Jacobian $\boldsymbol{J}$, the solution that minimizes the robot instantaneous kinetic energy (Khatib 1987), given by

$$\boldsymbol{J}^+ = \boldsymbol{M}^{-1}\boldsymbol{J}^T\boldsymbol{A} \ . \tag{2.30}$$

Linearization is also possible in task space through the feedback of task space term estimations, doing

$$\boldsymbol{\mu}_\mathrm{c} = \hat{\boldsymbol{c}}_x(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \hat{\boldsymbol{g}}_x(\boldsymbol{q}) + \hat{\boldsymbol{h}}_x(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \hat{\boldsymbol{\nu}}_x(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\mu}'_\mathrm{c} \ , \tag{2.31}$$

where $\boldsymbol{\mu}'_\mathrm{c}$ is the new command, thus leading to the system plant

$$\boldsymbol{A}(\boldsymbol{q})\ddot{\boldsymbol{x}} = \boldsymbol{\mu}_\mathrm{e} + \boldsymbol{\mu}'_\mathrm{c} \ . \tag{2.32}$$

Doing

$$\boldsymbol{\mu}'_\mathrm{c} = \hat{\boldsymbol{A}}(\boldsymbol{q})\, \boldsymbol{a}_{x\mathrm{c}} - \hat{\boldsymbol{\mu}}_\mathrm{e} \ , \tag{2.33}$$

decoupled resolved acceleration in task space is obtained:

$$\frac{\ddot{\boldsymbol{x}}(s)}{\boldsymbol{a}_{x\mathrm{c}}(s)} = \frac{1}{s^2} \ . \tag{2.34}$$

[2]Dependencies on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ were omitted in the equations.

2.4.3 Impedance model

Dynamic model enables impedance shaping, i.e., the design of robot force output as a function of motion input, also know as impedance control (Hogan 1985). The impedance $\boldsymbol{Z}$ is given, in Laplace domain, by

$$\boldsymbol{Z}(s) = \frac{\boldsymbol{\mu}(s)}{\dot{\boldsymbol{x}}(s)} = \frac{\boldsymbol{\mu}(s)}{s\boldsymbol{x}(s)} \,. \tag{2.35}$$

For instance, the impedance of a typical mass-damper-spring system with mass $\boldsymbol{A}$, damping gain $\boldsymbol{D}$, and spring gain $\boldsymbol{K}$ is

$$\boldsymbol{Z}(s) = \frac{\boldsymbol{A}s^2 + \boldsymbol{D}s + \boldsymbol{K}}{s} \,, \tag{2.36}$$

since in time domain the system dynamics is given by

$$\boldsymbol{A}\ddot{\boldsymbol{x}} + \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = \boldsymbol{\mu} \,. \tag{2.37}$$

In fact (2.28) is in the form of (2.37). Impedance control enables the robot to have a designed compliant behavior, like a mass-damper-spring system, instead of being a rigid positioning system. With the knowledge of the dynamic model, a robot can be controlled in such a way that, for example, the impedance seen from the environment,

$$\boldsymbol{Z}_{\mathrm{e}}(s) = \frac{\boldsymbol{\mu}_{\mathrm{e}}(s)}{s\boldsymbol{x}(s)} \,, \tag{2.38}$$

is shaped as desired. This technique is know as impedance shaping, and is very useful in co-manipulation applications.

Additionally, a system can also be described by its admittance $\boldsymbol{Z}^{-1}$, the inverse of impedance, which relates the response motion to the input force:

$$\boldsymbol{Z}^{-1}(s) = \frac{\dot{\boldsymbol{x}}(s)}{\boldsymbol{\mu}(s)} \,. \tag{2.39}$$

2.4.4 Adaptive control with AOB

Cortesão et al. (2006) presented a bilateral tele-manipulation architecture where force is used both for robot command and for feedback to the operator. Such force is computed through a virtual coupling spring dependent on position tracking error. The architecture features an adaptive force controller in the robot side and makes use of computed torque, feedback linearization and other model-based techniques. Furthermore, the control scheme includes a task environment stiffness estimator and an active observer (AOB) for force estimation. The AOB is a reformulation of the Kalman filter which uses:

- closed loop plant model (instead of open loop);
- n augmented states to model disturbances of order n;
- stochastic information of measures and model (including disturbances) for design.

Figure 2.2 schematizes the robot plant and force control architecture. The

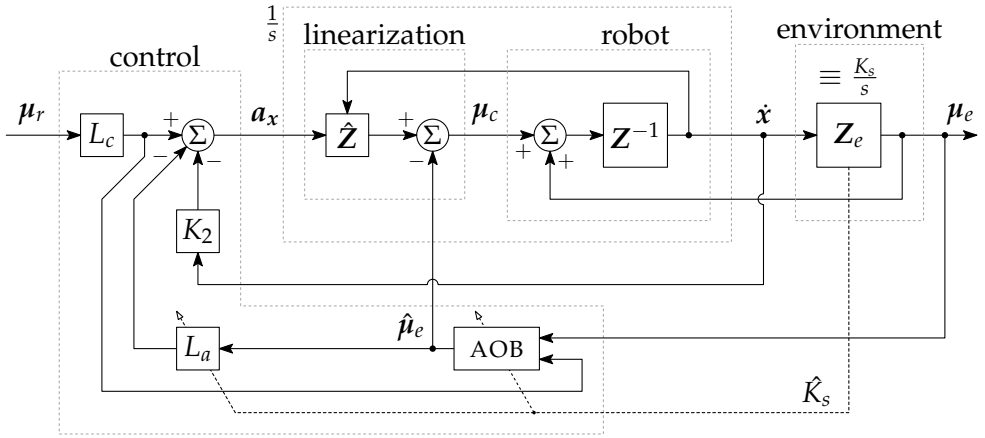

Figure 2.2: Adaptive force control with AOB. μ_r is the reference force; $\hat{Z}$ is the slave robot estimated impedance; L_c, L_a and K_2 are control gains. The environment is modeled by a spring K_s, whose estimation is $\hat{K}_s$.

control gains and the AOB are real-time adaptive to estimated environment stiffness. Only force data is used in the robot side of the controller: reference force, μ_r, environment contact measured force, μ_e, and its estimation, $\hat{\mu}_e$. This architecture is a good example of an advanced controller based on the robot dynamic model. This and other model-based controllers must take possible modeling errors into account, however, the better the model is, the better the controller performance will be.

2.5 Dynamic model identification

The complete knowledge of a robot dynamic model includes the knowledge of the dynamic parameters. The identification of the dynamic parameters is a common problem in robotics. Classically, three major techniques exist:

1. identification through computer-aided design (CAD) software;

2. identification through experimental measurements of individual robot parts;

3. identification through dynamic model regression methods.

The CAD estimation is known to not provide the most accurate results (Khalil and Dombre 2002; Wu et al. 2010). On the other hand, the measurement of individual robot parts, although accurate, is often impracticable. The use of regression techniques is more common, well studied and there is a high quantity of literature about it. See for example Atkeson et al. (1986), Gautier (1986), and Swevers et al. (2007) for articles, and Khalil and Dombre (2002), Siciliano and Khatib (2008), and Siciliano et al. (2009) for books on the basics of dynamic parameter identification by regression. As seen in section §2.2, the robot dynamic model can be formulated linearly to the dynamic parameters, thus allowing linear regression techniques to be applied.

The classical regression methods involve the collection of a large amount data. Usually, joint position and force measurements are collected by moving the robot along an excitation trajectory. Doing S measurements (with $S \gg N$), one can construct a regression matrix

$$\boldsymbol{H}_S = \begin{bmatrix} \boldsymbol{H}(\boldsymbol{q}_1, \dot{\boldsymbol{q}}_1, \ddot{\boldsymbol{q}}_1) \\ \boldsymbol{H}(\boldsymbol{q}_2, \dot{\boldsymbol{q}}_2, \ddot{\boldsymbol{q}}_2) \\ \vdots \\ \boldsymbol{H}(\boldsymbol{q}_S, \dot{\boldsymbol{q}}_S, \ddot{\boldsymbol{q}}_S) \end{bmatrix} , \tag{2.40}$$

and a regressand vector

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\tau}_{\mathrm{c}1} \\ \boldsymbol{\tau}_{\mathrm{c}2} \\ \vdots \\ \boldsymbol{\tau}_{\mathrm{c}S} \end{bmatrix} , \tag{2.41}$$

where $\boldsymbol{q}_1$ to $\boldsymbol{q}_S$ are the S measured joint position vectors, and $\boldsymbol{\tau}_{\mathrm{c}1}$ to $\boldsymbol{\tau}_{\mathrm{c}S}$ are the S measured force vectors. Depending on the robot sensors, $\dot{\boldsymbol{q}}_1$ to $\dot{\boldsymbol{q}}_S$ and $\ddot{\boldsymbol{q}}_1$ to $\ddot{\boldsymbol{q}}_S$ are obtained either by direct measurement or by numerical differentiation. The identified dynamic parameter vector, $\hat{\boldsymbol{\delta}}$, will then be the one which best fits

$$\boldsymbol{H}_S \hat{\boldsymbol{\delta}} \approx \boldsymbol{\omega} , \tag{2.42}$$

where $\approx$ represents an approximation. One of the simplest regression technique is the ordinary least squares (OLS), whose solution, $\hat{\boldsymbol{\delta}}_{\mathrm{OLS}}$, minimizes the

sum of the squared residuals,

$$\hat{\boldsymbol{\delta}}_{\text{OLS}} = \arg\min_{\boldsymbol{\delta}} \ \|\boldsymbol{\epsilon}\|^2 \ , \tag{2.43}$$

where

$$\boldsymbol{\epsilon} = \boldsymbol{\omega} - \boldsymbol{H}_S \boldsymbol{\delta} \ , \tag{2.44}$$

is the residuals vector.

2.5.1 Base dynamic parameters

When the rank of the regression matrix is equal to number of estimation variables (n in the present problem) the OLS has an analytic solution. However, with some very special exceptions, all robots have regressor functions with some null columns and some columns linearly dependent between them. Those columns correspond to standard dynamic parameters which have no effect on the dynamics, and to parameters which have proportional contributions, respectively. This implies that no matter how many or how exciting the measurement points are, the regression matrix $\boldsymbol{H}_S$ will always have rank lower than n. To overcome this issue, a set of *base dynamic parameters* (also know as minimal parameters) can be found through the elimination of the unidentifiable parameters and through the grouping of the dependent ones (Khalil and Kleinfinger 1987; Gautier and Khalil 1990; Mayeda et al. 1990). Equation (2.22) can be rewritten, by reordering, as

$$\begin{bmatrix} \boldsymbol{H}_\text{b} & \boldsymbol{H}_\text{d} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_\text{b} \\ \boldsymbol{\delta}_\text{d} \end{bmatrix} = \boldsymbol{\tau}_\text{c} \ , \tag{2.45}$$

where $\boldsymbol{H}_\text{b}$ are a maximum number n_b of linearly independent columns of $\boldsymbol{H}$, and $\boldsymbol{H}_\text{b}$ are the remain n_d null and dependent columns, with

$$n = n_\text{b} + n_\text{d} \ . \tag{2.46}$$

Since the columns of $\boldsymbol{H}_\text{b}$ form a base, the columns of $\boldsymbol{H}_\text{d}$ can be written as a linear combinations of such base:

$$\boldsymbol{H}_\text{d} = \boldsymbol{H}_\text{b} \boldsymbol{K}_\text{d} \ , \tag{2.47}$$

where $\boldsymbol{K}_\text{d}$ is the matrix encoding the linear combination. The vectors $\boldsymbol{\delta}_\text{b}$ and $\boldsymbol{\delta}_\text{d}$ are the standard dynamic parameters reordered according to the same

reorder of $\boldsymbol{H}$. The vector δ_b includes the parameters corresponding to the base columns of $\boldsymbol{H}$; the vector δ_d includes the remaining parameters, which are related to the dependent columns of $\boldsymbol{H}$. The reordering can be described by a permutation matrix $\boldsymbol{P}$,

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_\mathrm{b} & \boldsymbol{P}_\mathrm{d} \end{bmatrix} , \tag{2.48}$$

where $\boldsymbol{P}_\mathrm{b}$ and $\boldsymbol{P}_\mathrm{d}$ are, respectively, the n_b first columns and the n_d last columns of $\boldsymbol{P}$, which verify

$$\begin{aligned} \boldsymbol{H}_\mathrm{b} &= \boldsymbol{H}\boldsymbol{P}_\mathrm{b} , \\ \boldsymbol{H}_\mathrm{d} &= \boldsymbol{H}\boldsymbol{P}_\mathrm{d} , \\ \delta_\mathrm{b} &= \boldsymbol{P}_\mathrm{b}^T \delta , \\ \delta_\mathrm{d} &= \boldsymbol{P}_\mathrm{d}^T \delta . \end{aligned} \tag{2.49}$$

From (2.45) and (2.47),

$$\boldsymbol{H}_\mathrm{b} \begin{bmatrix} \mathbf{1} & \boldsymbol{K}_\mathrm{d} \end{bmatrix} \begin{bmatrix} \delta_\mathrm{b} \\ \delta_\mathrm{d} \end{bmatrix} = \boldsymbol{\tau}_\mathrm{c} , \tag{2.50}$$

therefore

$$\boldsymbol{H}_\mathrm{b}(\delta_\mathrm{b} + \boldsymbol{K}_\mathrm{d}\delta_\mathrm{d}) = \boldsymbol{\tau}_\mathrm{c} . \tag{2.51}$$

The inverse dynamic equation, (2.22), can then be written in terms of base parameters by

$$\boldsymbol{H}_\mathrm{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\, \boldsymbol{\beta} = \boldsymbol{\tau}_\mathrm{c} , \tag{2.52}$$

where $\boldsymbol{H}_\mathrm{b}$ is the base regressor, given in (2.49), and $\boldsymbol{\beta}$ is the base dynamic parameter vector of size n_b, given by the linear combination of the δ dynamic parameters:

$$\boldsymbol{\beta} = \boldsymbol{K}\delta , \tag{2.53}$$

with

$$\boldsymbol{K} = \boldsymbol{P}_\mathrm{b}^T + \boldsymbol{K}_\mathrm{d}\boldsymbol{P}_\mathrm{d}^T . \tag{2.54}$$

In practice, matrices $\boldsymbol{P}_\mathrm{b}$, $\boldsymbol{P}_\mathrm{d}$ and $\boldsymbol{K}_\mathrm{d}$ can be obtained either by rule-based (Gautier and Khalil 1990; Mayeda et al. 1990; Khalil et al. 1994; Khalil and Bennis 1995) or numerical-based methods (Gautier 1990, 1991). The linear combinations, and therefore these matrices, are not unique. However, it is common to choose the base by selecting the first linearly independent columns starting

from the left of $\boldsymbol{H}$. The linear regression can then be performed with respect to base parameters whose OLS solution is given by

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \arg\min_{\boldsymbol{\beta}} \ \|\boldsymbol{\epsilon}\|^2 \,, \tag{2.55}$$

with the residuals given by

$$\boldsymbol{\epsilon} = \boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta} \,, \tag{2.56}$$

and the regression matrix given by

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{H}_{\text{b}}(\boldsymbol{q}_1, \dot{\boldsymbol{q}}_1, \ddot{\boldsymbol{q}}_1) \\ \boldsymbol{H}_{\text{b}}(\boldsymbol{q}_2, \dot{\boldsymbol{q}}_2, \ddot{\boldsymbol{q}}_2) \\ \vdots \\ \boldsymbol{H}_{\text{b}}(\boldsymbol{q}_S, \dot{\boldsymbol{q}}_S, \ddot{\boldsymbol{q}}_S) \end{bmatrix} \,. \tag{2.57}$$

Providing that the measurement points excite well all parameters and lead to a $\boldsymbol{W}$ matrix with rank equal to n_{b}, the optimal solution $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ is unique and can be analytically computed by

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = (\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{W}^T\boldsymbol{\omega} \,. \tag{2.58}$$

All the non-unique optimal solutions of (2.43) map to the same base parameter solution:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \boldsymbol{K}\hat{\boldsymbol{\delta}}_{\text{OLS}} \,. \tag{2.59}$$

In fact, for each $\hat{\boldsymbol{\beta}}$ vector picked from the $\boldsymbol{\beta}$ space there is more than one vector in the $\boldsymbol{\delta}$ space mapping into it. Yoshida and Khalil (2000) called the set of all $\boldsymbol{\delta}$ vectors that map to the same $\hat{\boldsymbol{\beta}}$, the *virtual parameters* of $\hat{\boldsymbol{\beta}}$, defined by $\mathcal{V}_{\hat{\boldsymbol{\beta}}}$ as:

$$\mathcal{V}_{\hat{\boldsymbol{\beta}}} \equiv \{\boldsymbol{\delta} \in \mathbb{R}^n : \boldsymbol{K}\,\boldsymbol{\delta} = \hat{\boldsymbol{\beta}}\} \,. \tag{2.60}$$

There is a rich literature on robot dynamic parameter identification, highlighting many different aspects. Some literature focus the regression data generation, by studding techniques to create trajectories which excite well the parameters and also by discussing how such parameter excitation can be measured. See, for example, the works of Armstrong (1989), Gautier and Khalil (1992), Vandanjon et al. (1995), Swevers et al. (1997), Calafiore et al. (2001), and Park (2006). Some works study how to better deal with the data noise and bias, proposing methods more advanced than the classical OLS,

taking into account the uncertainties from not only the output variable $\boldsymbol{\tau}$ but also from the input ones, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ (using maximum-likelihood approaches for example). See, for instance, the works of Swevers et al. (1997), Gautier and Poignet (2001), Calafiore et al. (2001), Olsen and Petersen (2001), Olsen et al. (2002), Ting et al. (2006), Ting et al. (2011), Janot et al. (2009), and Janot et al. (2014). Other literature explore the dynamic model itself, proposing alternative models (e.g., the power model, filtered models) which can provide some advantages in the parameter identification. See, for example, the works of Prufer et al. (1994), Gautier (1996), Gautier (1997), Gautier and Poignet (2001), and Gautier et al. (2013b). A good overview on the state of the art of dynamic parameter identification techniques has been compiled by Wu et al. (2010).

2.6 The physical feasibility problem

A drawback of dynamic parameter identification through regression is the fact that the estimation solution can be not physically feasible. An estimated solution is physically feasible when it is possible to exist in real world, since the dynamic parameters express quantities of physical properties which are bounded to values with meaning. The most straight example of such quantity is the mass: it is always positive.

The physical feasibility of identified robot dynamic parameters was first studied in a seminal work by Yoshida et al. in the nineties (Yoshida et al. 1994; Yoshida 1995; Yoshida and Khalil 2000). They studied how the estimated parameters affect the positive definiteness of the inertia matrix ($\boldsymbol{M}$ in (2.9)). This matrix is know to only present positive semidefinite values in practice, which are associated with always positive energy. Yoshida et al. showed that some dynamic parameter solutions lead to non-positive inertia matrices at some (or even all) robot postures. The fact that some estimations present these issues can be related not only to errors and lack of richness in the regression data, but also to model accuracy itself (Yoshida et al. 1994; Farhat et al. 2008). Moreover, regression methods may be used considering some assumptions about the model that are seldom meet, for example, the assumption of determinism in OLS methods. On the other hand, the robot model itself is always an approximation to an infinitely more complex system. For example, while its almost a standard procedure to consider the robot dynamic model as being

linear in the parameters, it is also true that joint frictions can present highly nonlinear behaviors. The extent to which each of these factors contribute to obtaining infeasible solutions has not yet been studied.

While the problem of checking the feasibility of standard dynamic parameters can be solved by verifying a set of conditional inequalities, the same does not applies to the case of a base parameter estimation. Adding to that, the feasibility issues appear in contexts of identification by regression where only base parameter can be uniquely identified. The problem of having infeasible estimations that entail inertia matrix not always positive definite is directly related with the usefulness of such matrix itself. Non-positive matrices are intrinsically unstable when used in model-based controllers and entail unrealistic results when used in robot simulations (Yoshida and Khalil 2000), thus they have limited usefulness. Although some estimations are "so infeasible" that their application is clearly problematic, a bigger problem can come from solutions which entail non-positive definite inertia matrices at very few robot postures. For the latter case, an identification can be being used without visible problems until the robot performs an uncommon trajectory where the inertia matrix becomes not positive definite and unexpected and dangerous behavior occurs.

Although the physically consistency of identifications has paramount importance, few are the published works on dynamic identification which do not disregard this aspect. Nevertheless, there are some articles whose authors have acknowledged the feasibility issue. Besides the seminal work of Yoshida et al. the first works to acknowledge the issue were done by Calafiore et al. (2001), Benimeli et al. (2003), Mata et al. (2005), Ting et al. (2006), and Radkhah et al. (2007). In many of these works, the obtained estimations are clearly inconsistent, and some propose techniques to overcome the problem. The feasibility issue seems to arise even more often in works addressing the identification of parallel arms (Farhat et al. 2007) and humans/humanoids (Venture et al. 2009; Ayusawa and Nakamura 2010), which pose parameter excitation difficulties and large number of DOF. To the extent of our knowledge, up to this date, the physical feasibility subject has never been treated in any textbook on robotics.

It can be argued that adequate experiments can lead to estimations that are accurate enough to be physically feasible. However, no matter how good the parameter excitation and regression techniques are, a mathematical proof

that guarantees physical feasibility of robot base parameter estimation is necessary. Moreover, a regression method constrained to physically feasible solutions can be of interest. In the sequel, the feasibility conditions, following the base work of Yoshida et al., are introduced. Then, the various approaches to the problem which have been proposed up to now are presented. It has to be noted that several terms have been used among authors to refer to the same problem, hence the terms "physical impossibility", "physical inconsistency", and "physical infeasibility" (or "unfeasibility") must be taken as synonyms.

2.6.1 Feasibility of inertial parameters

Yoshida and Khalil (2000) defined that inertial parameters are feasible if the following conditions are verified:

$$\begin{cases} m_k > 0 \\ \boldsymbol{I}_k \succ 0 \end{cases} \quad \text{for} \quad k = 1, \ldots, N \,, \tag{2.61}$$

where the $\succ 0$ notation means that the left-hand side is a positive definite matrix. Through (2.16), (2.61) can be rewritten in terms of link inertial parameters as

$$\begin{cases} m_k > 0 \\ \boldsymbol{L}_k - \dfrac{1}{m_k} \mathbf{S}(\boldsymbol{l}_k)^T \mathbf{S}(\boldsymbol{l}_k) \succ 0 \end{cases} \quad \text{for} \quad k = 1, \ldots, N \,. \tag{2.62}$$

The set of all dynamic parameter vectors which are physically feasible, with respect to inertial parameters, can be defined as

$$\mathcal{D}_{\mathrm{L}} = \left\{ \boldsymbol{\delta} \in \mathbb{R}^n : \; m_k > 0, \; \boldsymbol{L}_k - \frac{1}{m_k} \mathbf{S}(\boldsymbol{l}_k)^T \mathbf{S}(\boldsymbol{l}_k) \succ 0 \mid k = 1, \ldots, N \right\} . \tag{2.63}$$

2.6.2 Feasibility of additional dynamic parameters

Besides the physical conditions on the link inertial parameters, any other additional parameter included in the model must be conditioned to the physical meaning it represents. The set of dynamic parameter vectors which verify those additional conditions will be denoted by $\mathcal{D}_{\mathrm{A}}$. For example, the joint viscous and Coulomb frictions and the reflected motor inertia parameters, as given in (2.19), must be positive in order to be physically feasible. Hence, for that example, $\mathcal{D}_{\mathrm{A}}$ is defined as

$$\mathcal{D}_{\mathrm{A}} = \left\{ \boldsymbol{\delta} \in \mathbb{R}^n : \; F_{\mathrm{v}k} > 0, \; F_{\mathrm{c}k} > 0, \; I_{\mathrm{a}k} > 0 \mid k = 1, \ldots, N \right\} . \tag{2.64}$$

Considering the conditions on all parameters, the set of physically feasible dynamic parameter vectors is given by

$$\mathcal{D} = \mathcal{D}_{\mathrm{L}} \cap \mathcal{D}_{\mathrm{A}} \,, \tag{2.65}$$

i.e., it is the set defined by the intersection of all conditions.

2.6.3 Feasibility conditions defined by additional robot information

The physical feasibility conditions presented above are valid for any robot. However, additional information about a given robot can be used to better define the physical feasibility of the dynamic parameters. For instance, if it is possible to detach a group of links, from $k = a$ to $k = b$, and measure their total weight $m_{a,b}$, an additional condition can be added to the feasibility conditions:

$$\sum_{k=a}^{b} m_k = m_{a,b} \,. \tag{2.66}$$

The position of the centers of mass may also be used to refine the feasibility conditions. The center of mass of a link is always positioned within the link body convex hull, i.e. the smallest convex volume which contains the link. Hence, if additional information about the geometry of a given robot links is available, additional conditions on centers of mass can be devised. Interval of confidence from other estimation procedures may also be used as additional feasibility conditions. Additional conditions can be appended to the definition of (2.65) and many can be integrated into the methods that will be presented in the sequel.

2.6.4 Feasibility of standard dynamic parameters

Given a standard dynamic parameter estimation $\hat{\delta}$, it is straightforward to verify its feasibility by checking if it verifies the conditions of $\mathcal{D}$. In order to verify the positive definiteness conditions, several simple techniques can be applied. They are the the Sylvester's criterion (Gilbert 1991), the verification of eigenvalues positiveness, and the Cholesky decomposition which shows to be impossible when the matrix is not positive definite.

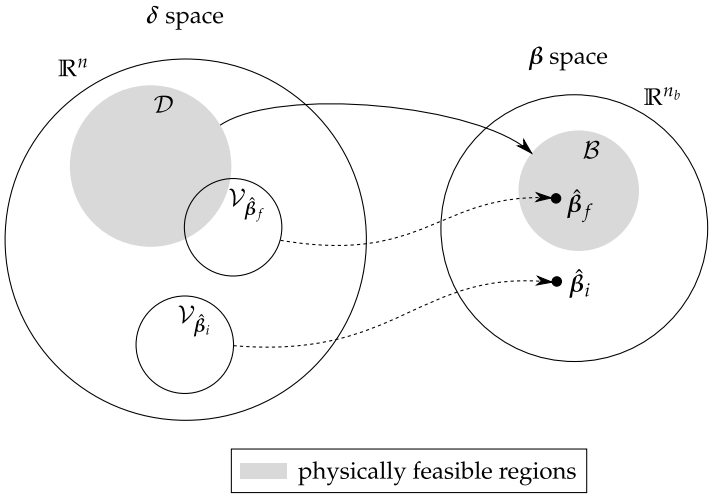

Figure 2.3: Relation between feasible sets in δ and β spaces (2D illustration). $\hat{\beta}_f$ is a feasible parameter vector and $\hat{\beta}_i$ is an infeasible one.

2.6.5 Feasibility of base dynamic parameters

A given base dynamic parameter vector, $\hat{\beta}$, is said to be physical feasible if there is at least one feasible standard dynamic parameter vector mapping into it (Yoshida and Khalil 2000). Therefore, merging (2.60) and (2.63), the set of all base dynamic parameter vectors which are physically feasible, can be defined by

$$\mathcal{B} = \{\beta \in \mathbb{R}^{n_b} : \exists\, \delta \in \mathcal{D} \mid K\delta = \beta\}\,. \tag{2.67}$$

Since the set of all standard dynamic parameters mapping into a given $\hat{\beta}$ is its virtual parameter set, $\mathcal{V}_{\hat{\beta}}$, one can say that such $\hat{\beta}$ is feasible if, and only if, $\mathcal{V}_{\hat{\beta}}$ intersects $\mathcal{D}$:

$$\hat{\beta} \in \mathcal{B} \quad \Leftrightarrow \quad \mathcal{V}_{\hat{\beta}} \cap \mathcal{D} \neq \emptyset\,. \tag{2.68}$$

The relation between this sets in both δ and β spaces is illustrated Figure 2.3. The transformation from δ to β space, performed by the matrix K, is surjective since each δ maps to a single β while the opposite is not true, thus entailing a non-bijective transformation. Due to that fact, the definition given in equation (2.67) cannot be used straightforwardly to verify if a $\hat{\beta}$ if physically feasible. This raises the problem: how to verify whether a $\hat{\beta}$ solution obtained from a regression method is feasible?

2.6.6 Previous works on the physical feasibility issue

Along with the feasibility/infeasibility condition definition, Yoshida et al. proposed a method to check if a given base parameter estimation is physically possible (Yoshida et al. 1994; Yoshida 1995; Yoshida and Khalil 2000). The method is recursive, starting from the end-effector and then iterating over each link. At each link one must try to find a vector of standard inertial parameters which verify the feasibility constraints given by (2.61) and at the same time correspond it to the identified base parameters of that link. Whenever a solution is found, one moves to the next link towards the robot base, and repeats the procedure taking the previous standard parameter solutions into account. The search for each solution is performed within regions with at most 3 degrees of freedom, which are defined and reduced by the very same rules used to group parameters into the base by Mayeda et al. (1990) and Gautier and Khalil (1990). The choice of values in the regions is done recurring to visual graphs. If it is possible to complete the process up to the base link, then the found standard inertial parameter solution is a feasible virtual parameter of the given base parameter estimation, thus proving its feasibility. If at some link, a solution cannot be found, the method must move back to the previous link and try again with a valid but different solution. Hence, although the method is systematic, it is in fact a trial and error procedure. While the feasibility can be proved when a valid solution is found, the proof of infeasibility seems to be harder to obtain since the search region of each link depends on the solutions chosen at the others. That makes the method impracticable for larger DOF robots.

Constrained regression methods

Mata et al. (2005) proposed a method for parameter identification taking physical feasibility into account. The method extends the classical OLS regression into a nonlinear optimization with nonlinear constrains. The optimization objective is to find a standard parameter estimation which minimizes the regression error and the distance to a prior base estimation, as given by the function

$$f(\delta) = \lambda_\omega \|\omega - \boldsymbol{W}\boldsymbol{K}\delta\|^2 + \lambda_\beta \|\hat{\boldsymbol{\beta}} - \boldsymbol{K}\delta\|^2 \,, \tag{2.69}$$

where $\hat{\boldsymbol{\beta}}$ is the prior estimation to be fitted, and λ_ω and λ_β are weighting factors. The optimization is performed referring to sequential quadratic pro-

gramming (SQP) methods which enable the inclusion of nonlinear constrains in the space of δ to ensure feasibility. The constrains impose that all masses, m_k, and all the eigenvalues of all COM inertia tensors, $\boldsymbol{I}_k$, are positive, which effectively is equivalent to (2.61). Since the convergence of this optimization was not studied and not guaranteed, Mata et al. opted to use a two-step formulation and recurred to parameter estimations taken from literature as initial guess. In the first step, only mass and inertia tensor diagonal positiveness is constrained. The solution of the first step is then the initial guess of a second step where the eigenvalues positiveness is take into account too. The selection of the prior solution $\hat{\boldsymbol{\beta}}$ of (2.69) and the tuning of the weighting factors λ_ω and λ_β are not discussed. In a work with a 3-DOF parallel robot by Farhat et al. (2007, 2008), nonlinear optimization constrained to feasible solutions is also performed. Although no details are given about the optimization formulation, that work seems to be related to the one of Mata et al. (2005), with the difference that the Sylvester's criterion, rather than the eigenvalues positiveness, is used for the feasibility constraints. It is worth mentioning that in the latter work, nonlinear friction models are also used, hence the requirement for a nonlinear optimization processes does not comes exclusively from the feasibility constrains.

Ting et al. proposed a nonlinear Bayesian parameter identification method with physical feasibility assessment (Ting et al. 2006; Ting et al. 2011). Rather than being done in the standard or base parameter space, the regression optimization step of the identification method is done in a new parameter space, the "virtual parameters" $\boldsymbol{\theta}$ (not related to (2.60)) which has a nonlinear mapping to feasible standard parameters only. The space of such virtual parameters is defined for each link k by the vector $\boldsymbol{\theta}_k$,

$$\boldsymbol{\theta}_k = \begin{bmatrix} \theta_{1k} & \theta_{2k} & \cdots & \theta_{11k} \end{bmatrix}^T , \tag{2.70}$$

which maps to the standard inertial parameters $\delta_{\mathrm{L}k}$ (plus the viscous friction

constant) through the following equations:

$$\begin{aligned}
m_k &= \theta_{1k}^2 \\
l_{xk} &= \theta_{1k}^2\theta_{2k} \\
l_{yk} &= \theta_{1k}^2\theta_{3k} \\
l_{zk} &= \theta_{1k}^2\theta_{4k} \\
L_{xxk} &= \theta_{1k}^2\left(\theta_{3k}^2+\theta_{4k}^2\right)+\theta_{5k}^2 \\
L_{xyk} &= -\theta_{1k}^2\theta_{2k}\theta_{3k}+\theta_{5k}\theta_{6k} \\
L_{xzk} &= -\theta_{1k}^2\theta_{2k}\theta_{4k}+\theta_{5k}\theta_{7k} \\
L_{yyk} &= \theta_{1k}^2\left(\theta_{2k}^2+\theta_{4k}^2\right)+\theta_{6k}^2+\theta_{8k}^2 \\
L_{yzk} &= -\theta_{1k}^2\theta_{3k}\theta_{4k}+\theta_{6k}\theta_{7k}+\theta_{8k}\theta_{9k} \\
L_{zzk} &= \theta_{10k}^2+\theta_{1k}^2\left(\theta_{2k}^2+\theta_{3k}^2\right)+\theta_{7k}^2+\theta_{9k}^2 \\
F_{\mathrm{v}k} &= \theta_{11}^2 \,.
\end{aligned} \tag{2.71}$$

The parameters $\boldsymbol{\theta}$ correspond to the mass square root, the COM position ($\boldsymbol{r}$), a said "Cholesky decomposition of the DOF's inertia matrix at the center of gravity", and the viscous friction square root, which according to Ting et al. (2011) encode the feasibility conditions in such a way that every possible $\boldsymbol{\theta}$ vector maps to a physically feasibility standard parameter vector. Due to the difficulty in including the nonlinear mapping into the Bayesian identification formulation, a two-step solution is proposed. Firstly, an unconstrained solution $\hat{\boldsymbol{\delta}}_{uc}$ in the standard parameter space is found by the Bayesian method (or possibly other identification method). Then, a nonlinear least squares optimization is performed in the $\boldsymbol{\theta}$ parameters space, seeking to minimize the distance to the unconstrained solution under a metric given by the regression matrix $\boldsymbol{H}$:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\| \boldsymbol{H}\hat{\boldsymbol{\delta}}_{uc} - \boldsymbol{H}\boldsymbol{\delta}_{\theta} \right\| , \tag{2.72}$$

with $\boldsymbol{\delta}_{\theta}$ being given by the function f_{θ},

$$\boldsymbol{\delta}_{\theta} = f_{\theta}(\boldsymbol{\theta}) \,, \tag{2.73}$$

which performs the mapping formulated by (2.71) for all links. Since the least squares minimization is weighted by the regression matrix $\boldsymbol{H}$, the optimal solution $\hat{\boldsymbol{\delta}} = f_{\theta}(\hat{\boldsymbol{\theta}})$ will be the one that best fits the regression data. The same physical feasibility assessment technique also appear in the work Nakanishi et al. (2008), where the prior unconstrained solution is obtained by classical

least squares methods. While claiming that the presented nonlinear mapping guarantee always feasible standard parameters, no further details on the proof of such claim and no details on the nonlinear optimization implementation are given in the cited works. It is also not discussed if every feasible vector in δ has a corresponding solution in the space of θ, i.e., if the best θ solution really corresponds to the best feasible δ solution.

Ayusawa and Nakamura (2010) proposed an identification technique enabling physically feasible parameter identification for robots with a high number of DOF. The work extends the formulation of Mata et al. (2005) introducing two major differences to mitigate two major problems of the formulation on large DOF robots. First, the feasibility constraints are approximated by linear inequalities which helps reducing the complexity of the optimization and improve its performance and effectiveness — in fact, Ayusawa et al. were unable to obtain a converged solution in their 34-DOF identification experiment when using the nonlinear constrain formulation. Second, a prior solution taken from CAD data is used, adding stability to the identification when it is not possible to successfully excite all identifiable dynamic parameters — which is the common case for large DOF robots. To obtain linear conditions, instead of considering the links as generic rigid bodies, the links are considered to be composed by a finite number of mass points at fixed positions, leading to a set of parameters only including masses. Being $\boldsymbol{p}$ the vector of mass parameters for all mass points grouped by link, the link inertial parameter vector is given by

$$\delta = \boldsymbol{R}\boldsymbol{p} \,, \tag{2.74}$$

where $\boldsymbol{R}$ is a block-diagonal matrix (a block per link) encoding the position of the mass points and thereby the contribution of each to the link inertia. With such formulation, the physical feasibility conditions are reduced to the parameters positiveness, which can be defined by simple linear inequalities:

$$\begin{cases} p_1 > 0 \\ p_2 > 0 \\ p_3 > 0 \\ \quad \vdots \end{cases} . \tag{2.75}$$

The regression optimization problem is then formulated by

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{minimize}} \quad & \|\boldsymbol{G}_f(\boldsymbol{\omega} - \boldsymbol{H}_S\boldsymbol{R}\boldsymbol{p})\|^2 + \|\boldsymbol{G}_p(\hat{\boldsymbol{p}} - \boldsymbol{p})\|^2 \\ \text{subject to} \quad & \boldsymbol{p} > \boldsymbol{0}\,, \end{aligned} \tag{2.76}$$

where $\hat{\boldsymbol{p}}$ is a prior estimation obtained from CAD data, and $\boldsymbol{G}_f$ and $\boldsymbol{G}_p$ are weighting matrices. If $\boldsymbol{R}$ is full rank, which can be achieved by having a minimum of 10 well distributed mass points per each link, then any $\boldsymbol{\delta}$ vector has at least one corresponding $\boldsymbol{p}$ solution. Hence, if the feasibility inequalities and the second term of (2.76) are discarded, the problem will be equivalent the classical OLS one. However, when considering positive $\boldsymbol{p}$ solutions only, it is not possible to fully map the $\boldsymbol{\delta}$ space into the $\boldsymbol{p}$ (positive) space. The set of $\boldsymbol{\delta}$ vectors given by all possible feasible $\boldsymbol{p}$ estimations, defined as

$$\mathcal{D}_{\boldsymbol{p}} = \{\boldsymbol{\delta} \in \mathbb{R}^n : \forall \boldsymbol{p} > \boldsymbol{0} \mid \boldsymbol{\delta} = \boldsymbol{R}\boldsymbol{p}\}\,, \tag{2.77}$$

is a polyhedral convex cone contained by the feasibility convex cone defined by (2.63) (Ayusawa and Nakamura 2010),

$$\mathcal{D}_{\boldsymbol{p}} \subset \mathcal{D}_{\mathrm{L}}\,. \tag{2.78}$$

The higher the number of distinct mass points used per link, the better the approximation of $\mathcal{D}_{\boldsymbol{p}}$ to $\mathcal{D}_{\mathrm{L}}$ will be. Ayusawa et al. proposes the use of $3^3 = 27$ mass points per link placed at the center, at the vertexes, at the centers of the edges, and at the centers of the faces of the bounding box enclosing the respective link (obtained from CAD data too). This approach guarantees that the link centers of mass are inside their bounding boxes, which indeed is known to be true. While the simplification entails a error inverse to the number of mass points, such simplification enables faster optimization processes, up to the point that this formulation enabled the identification of human body segments in real-time (Ayusawa et al. 2011).

Model reduction methods

Díaz-Rodríguez et al. (2010) proposed an identification method which also considers parameter physical feasibility but differing from previous approaches both in the way the feasibility is checked and in the way the infeasibility is mitigated. Such identification method has an iterative process where the regression is performed in the base parameters space by a classical weighted

least squares (WLS) technique at each step. After each step, if the solution is feasible the process stops. If not, the model is reduced by removing the base parameter which presented the higher relative standard deviation in the previous solution (Pham and Gautier; Khalil and Dombre 2002) and the process repeats. To verify the feasibility of a given base estimation $\hat{\boldsymbol{\beta}}$, a feasible solution is searched for in the standard parameter space. Since through (2.53), (2.54) and (2.49)

$$\boldsymbol{\delta}_\mathrm{b} = \hat{\boldsymbol{\beta}} - \boldsymbol{K}_\mathrm{d}\boldsymbol{\delta}_\mathrm{d} \,, \tag{2.79}$$

Díaz-Rodríguez et al. search for a $\boldsymbol{\delta}_\mathrm{d}$ that in conjunction with the computed $\boldsymbol{\delta}_\mathrm{b}$ entails a $\boldsymbol{\delta}$,

$$\boldsymbol{\delta} = \boldsymbol{P}\begin{bmatrix}\boldsymbol{\delta}_\mathrm{b}\\ \boldsymbol{\delta}_\mathrm{d}\end{bmatrix} \,, \tag{2.80}$$

which verifies the feasibility conditions. The proposed procedure performs a discrete and bounded brute-force search in the space of $\boldsymbol{\delta}_\mathrm{b}$. If at some iteration a feasible solution is found, the remaining base parameters are considered to be the "relevant" parameters for that model. Nevertheless, Díaz-Rodríguez et al. acknowledge that in some cases it may not be possible to find feasible solutions for any level of model reduction.

Gautier et al. (2013a) presents a method where "essential" parameters are also used to tackle the physical infeasibility problem. In this method, unlike in the one of Díaz-Rodríguez et al. (2010), the essential parameters are devised one time only by the techniques of Pham and Gautier. The regression is then performed in the standard parameter space minimizing the torque prediction square error through singular value decomposition (SVD) techniques. Additionally, taking advantage the of standard parameter null space, the the solution is pulled towards a prior feasible CAD estimation. Hence, in case the least squares solution allows it, it is more probable for the final standard parameter solution to be feasible. In the experiments present in the work, the estimation in the essential parameters shows to be feasible, while no feasible estimation can be found when considering all parameters. Similar results are also found in the work of Gautier and Venture (2013). In the experiments of Venture and Gautier (2012), these techniques are applied to the identification of human body parameters, however no feasible solution was found at all. In all these related works, the feasibility is checked using the Cholesky factorization whose tolerance is used to infer the degree of infeasibility.

3

SOLVING THE PHYSICAL FEASIBILITY PROBLEM

3.1 Semialgebraic set approach

Physical feasibility conditions define the set $\mathcal{D}$ in the standard dynamic parameter space. For any given point in this space, it is possible to verify if it is in the feasibility region by checking the conditions in a closed-form approach. In this section, an attempt to obtain a closed-form expression for the base feasibility set $\mathcal{B}$ is presented.

3.1.1 Physically feasible dynamic parameter set as a semialgebraic set

It is known through the Sylvester's criterion, that a positive definite matrix has positive leading principal minors, i.e., all the leading sub-matrices have positive determinants. Applying this necessary and sufficient criterion, the positive definiteness condition on $\boldsymbol{I}_k$ in (2.61), $\boldsymbol{I}_k \succ 0$, can be rewritten as

$$\begin{cases} I_{xxk} > 0 \\ \det\left(\begin{bmatrix} I_{xxk} & I_{xyk} \\ I_{xyk} & I_{yyk} \end{bmatrix}\right) > 0 \\ \det\left(\boldsymbol{I}_k\right) > 0\,. \end{cases} \tag{3.1}$$

Mapping to link inertial parameters through the Huygens–Steiner theorem (2.16), expanding the determinants into closed-form formulas, and then multiplying by m_k^2 (entailing equivalent inequalities), equation (3.1) can be written as

$$\begin{cases} P_{1k}(\boldsymbol{\delta}_{\mathrm{L}k}) > 0 \\ P_{2k}(\boldsymbol{\delta}_{\mathrm{L}k}) > 0 \\ P_{3k}(\boldsymbol{\delta}_{\mathrm{L}k}) > 0 \end{cases}, \tag{3.2}$$

where P_{1k}, P_{2k} and P_{3k} are polynomials in the variables of vector $\boldsymbol{\delta}_{\mathrm{L}k}$ defined by (2.18):

$$P_{1k}(\boldsymbol{\delta}_{\mathrm{L}k}) = L_{xxk} m_k^2 - l_{yk}^2 m_k - l_{zk}^2 m_k\,, \tag{3.3}$$

$$\begin{aligned} P_{2k}(\delta_{\mathrm{L}k}) = {} & L_{xxk}L_{yyk}m_k^2 - L_{xxk}l_{xk}^2m_k - L_{xxk}l_{zk}^2m_k - L_{xyk}^2m_k^2 - 2L_{xyk}l_{xk}l_{yk}m_k \\ & - L_{yyk}l_{yk}^2m_k - L_{yyk}l_{zk}^2m_k + l_{xk}^2l_{zk}^2 + l_{yk}^2l_{zk}^2 + l_{zk}^4 \ , \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} P_{3k}(\delta_{\mathrm{L}k}) = {} & L_{xxk}L_{yyk}L_{zzk}m_k^2 - L_{xxk}L_{yyk}l_{xk}^2m_k - L_{xxk}L_{yyk}l_{yk}^2m_k - L_{xxk}L_{yzk}^2m_k^2 \\ & - 2L_{xxk}L_{yzk}l_{yk}l_{zk}m_k - L_{xxk}L_{zzk}l_{xk}^2m_k - L_{xxk}L_{zzk}l_{zk}^2m_k + L_{xxk}l_{xk}^4 \\ & + L_{xxk}l_{xk}^2l_{yk}^2 + L_{xxk}l_{xk}^2l_{zk}^2 - L_{xyk}^2L_{zzk}m_k^2 + L_{xyk}^2l_{xk}^2m_k + L_{xyk}^2l_{yk}^2m_k \\ & + 2L_{xyk}L_{xzk}L_{yzk}m_k^2 + 2L_{xyk}L_{xzk}l_{yk}l_{zk}m_k + 2L_{xyk}L_{yzk}l_{xk}l_{zk}m_k \\ & - 2L_{xyk}L_{zzk}l_{xk}l_{yk}m_k + 2L_{xyk}l_{xk}^3l_{yk} + 2L_{xyk}l_{xk}l_{yk}^3 + 2L_{xyk}l_{xk}l_{yk}l_{zk}^2 \\ & - L_{xzk}^2L_{yyk}m_k^2 + L_{xzk}^2l_{xk}^2m_k + L_{xzk}^2l_{zk}^2m_k - 2L_{xzk}L_{yyk}l_{xk}l_{zk}m_k \\ & + 2L_{xzk}L_{yzk}l_{xk}l_{yk}m_k + 2L_{xzk}l_{xk}^3l_{zk} + 2L_{xzk}l_{xk}l_{yk}^2l_{zk} + 2L_{xzk}l_{xk}l_{zk}^3 \\ & - L_{yyk}L_{zzk}l_{yk}^2m_k - L_{yyk}L_{zzk}l_{zk}^2m_k + L_{yyk}l_{xk}^2l_{yk}^2 + L_{yyk}l_{yk}^4 \\ & + L_{yyk}l_{yk}^2l_{zk}^2 + L_{yzk}^2l_{yk}^2m_k + L_{yzk}^2l_{zk}^2m_k + 2L_{yzk}l_{xk}^2l_{yk}l_{zk} \\ & + 2L_{yzk}l_{yk}^3l_{zk} + 2L_{yzk}l_{yk}l_{zk}^3 + L_{zzk}l_{xk}^2l_{zk}^2 + L_{zzk}l_{yk}^2l_{zk}^2 + L_{zzk}l_{zk}^4 \ . \end{aligned} \tag{3.5}$$

It follows that the feasibility set $\mathcal{D}$ can then be rewritten as

$$\begin{aligned} \mathcal{D} = \{ \delta \in \mathbb{R}^n : {} & m_k > 0, \\ & P_{1k}(\delta_{\mathrm{L}k}) > 0, \ P_{2k}(\delta_{\mathrm{L}k}) > 0, \ P_{3k}(\delta_{\mathrm{L}k}) > 0, \\ & F_{\mathrm{v}k} > 0, \ F_{\mathrm{c}k} > 0, \ I_{\mathrm{a}k} > 0 \\ & \mid k = 1, \ldots, N \} \ . \end{aligned} \tag{3.6}$$

Therefore, the set $\mathcal{D}$ is as subset of $\mathbb{R}^n$ defined by M polynomial inequalities,

$$\mathcal{D} = \{ \delta \in \mathbb{R}^n : Q_i(\delta) > 0 \mid i = 1, \ldots, M \} \tag{3.7}$$

where $Q_i(\delta)$ are polynomials, which proves that $\mathcal{D}$ is in fact a semialgebraic set.

3.1.2 Base–dependent parameter space

The transformation from the δ space to the β space can be defined by a map $\mathfrak{m}$,

$$\begin{array}{rccc} \mathfrak{m}: & \mathbb{R}^n & \longmapsto & \mathbb{R}^{n_{\mathrm{b}}} \\ & \delta & \longmapsto & \beta \end{array} \ , \tag{3.8}$$

with

$$\mathfrak{m}(\boldsymbol{\delta}) \equiv \boldsymbol{K}\boldsymbol{\delta} \,. \tag{3.9}$$

Such mapping can be decomposed into two mappings, $\mathfrak{m}_t$ and $\mathfrak{m}_p$,

$$\begin{array}{rccccc} \mathfrak{m} = \mathfrak{m}_t \circ \mathfrak{m}_p : \mathbb{R}^n & \xmapsto{\mathfrak{m}_t} & \mathbb{R}^n & \xmapsto{\mathfrak{m}_p} & \mathbb{R}^{n_\mathrm{b}} \\ \boldsymbol{\delta} & \longmapsto & (\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) & \longmapsto & \boldsymbol{\beta} \end{array} \,, \tag{3.10}$$

where $\mathfrak{m}_t$ is a bijective linear transformation defined by

$$\mathfrak{m}_t(\boldsymbol{\delta}) \equiv \begin{bmatrix} \mathbf{1}_{n_\mathrm{b}} & \boldsymbol{K}_\mathrm{d} \\ \mathbf{0} & \mathbf{1}_{n_\mathrm{d}} \end{bmatrix} \boldsymbol{P}^T \boldsymbol{\delta} \tag{3.11}$$

and $\mathfrak{m}_p$ is a projection defined by

$$\mathfrak{m}_p(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \equiv \begin{bmatrix} \mathbf{1}_{n_\mathrm{b}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\delta}_\mathrm{d} \end{bmatrix} \,. \tag{3.12}$$

The mapping $\mathfrak{m}_t$ is bijective since it has inverse, which is given by

$$\mathfrak{m}_t^{-1}(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \equiv \boldsymbol{P} \begin{bmatrix} \mathbf{1}_{n_\mathrm{b}} & -\boldsymbol{K}_\mathrm{d} \\ \mathbf{0} & \mathbf{1}_{n_\mathrm{d}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\delta}_\mathrm{d} \end{bmatrix} \,. \tag{3.13}$$

Therefore, there is a one-to-one correspondence between points in the space $\boldsymbol{\delta}$ and points in the space formed by $(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$ parameters. Henceforward, the $(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$ parameter space will be referred as *base–dependent* parameter space. One advantage of the base–dependent parameter space it that $(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$ vectors have the identifiable and the unidentifiable terms split into $\boldsymbol{\beta}$ and $\boldsymbol{\delta}_\mathrm{d}$, respectively, while $\boldsymbol{\delta}$ vectors are a mix of identifiable and unidentifiable values.

The set of feasible parameters can be mapped into the base–dependent parameter space, defining the set of feasible base–dependent parameters, $\mathcal{D}_\beta$, as

$$\mathcal{D}_\beta = \{(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \in \mathbb{R}^n : \; Q_{i\beta}(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) > 0 \mid i = 1, \ldots, M\} \,, \tag{3.14}$$

where $Q_{i\beta}$ are the polynomials of (3.7) rewritten in terms of $(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$:

$$Q_{i\beta}(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) = Q_i(\mathfrak{m}_t^{-1}(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})) \,. \tag{3.15}$$

Since $\mathfrak{m}_t^{-1}$ is a linear mapping, $Q_{i\beta}$ are still polynomials, and the set $\mathcal{D}_\beta$ is still a semialgebraic set. Figure 3.1 extends the example given in figure 2.3 to include the new base–dependent parameter space.

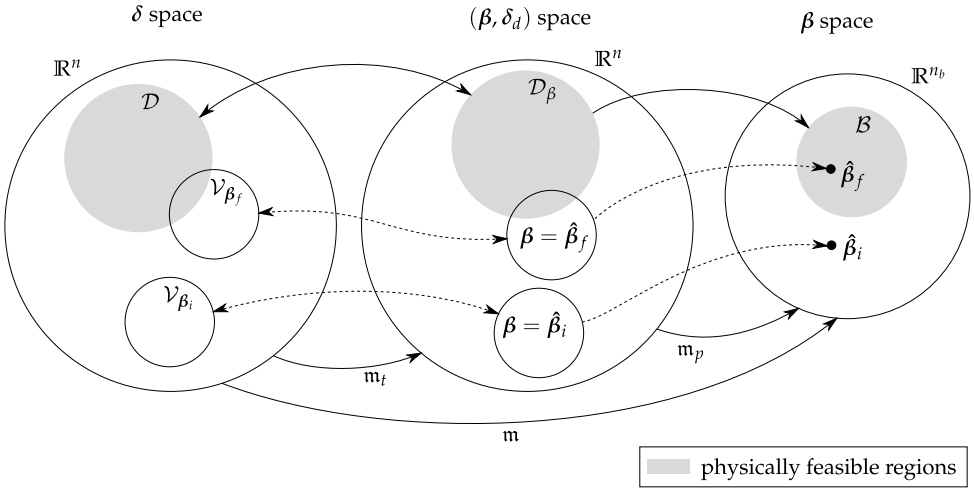

Figure 3.1: Relation between feasible sets in $\boldsymbol{\delta}$, $(\boldsymbol{\delta}, \boldsymbol{\beta})$, and $\boldsymbol{\beta}$ spaces (2D illustration). $\hat{\boldsymbol{\beta}}_f$ is a feasible parameter vector and $\hat{\boldsymbol{\beta}}_i$ is an infeasible one.

3.1.3 Feasible base parameter set as a semialgebraic set

The feasible base parameter set can be rewritten in terms of $\mathcal{D}_\beta$ as

$$\mathcal{B} = \{\boldsymbol{\beta} \in \mathbb{R}^{n_\mathrm{b}} : \exists\, \boldsymbol{\delta}_\mathrm{d} \in \mathbb{R}^{n_\mathrm{d}} \mid (\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \in \mathcal{D}_\beta\}\,. \tag{3.16}$$

which in fact represents a projection of $\mathcal{D}_\beta$ onto the $\mathbb{R}^{n_\mathrm{b}}$ subspace defined by the dimensions of $\boldsymbol{\beta}$. By the Tarski–Seidenberg theorem (Bierstone and Milman 1988), it is known that a projection of a semialgebraic set onto some of its dimensions is still a semialgebraic set. Therefore, an important conclusion is devised:

- the set $\mathcal{B}$ is a semialgebraic set, and its closed-form representation, free of existential quantifier ($\exists$) and definable in terms of polynomial identities and inequalities, exists.

Being quantifier free means that that the set $\mathcal{B}$ can be formulated with conditions expressed in terms of $\boldsymbol{\beta}$ variables only. Hence, such formulation can be directly used to check whether a given $\hat{\boldsymbol{\beta}}$ solution is physically feasible.

In order to obtain such quantifier free expressions, a quantifier elimination algorithm is needed, being the cylindrical algebraic decomposition (CADec) algorithm developed by Collins (1975) the most effective one. Unfortunately, while the CADec can be effectively implemented in a computer, it has a dou-

ble exponential complexity associated to the space dimension. Therefore, although it has been proven that a closed-form semialgebraic representation of $\mathcal{B}$ exists, in practice it may be impossible to find such representation. For small planar robots, where the number of inertial parameters can be cut down to 4 per link, the CADec algorithm may be capable of finding the closed-form conditions. However, for a common robot manipulator, the number of parameters grows to several dozens, rendering a computationally intractable CADec problem.

3.2 Linear matrix inequality and semidefinite programming approach

It has been shown that physical feasibility conditions define a semialgebraic set. In this section, the positive definite structure of the feasibility conditions will be further exploited, referring to linear matrix inequalities and semidefinite programming techniques.

3.2.1 Physically feasible dynamic parameter set as an LMI

The second condition of (2.62), which defines the positive definiteness of a given link inertia, can be written in a quadratic matrix inequality form as

$$\boldsymbol{L}_k - \mathbf{S}(\boldsymbol{l}_k)^T \, (m_k \mathbf{1})^{-1} \, \mathbf{S}(\boldsymbol{l}_k) \succ 0 \, . \tag{3.17}$$

Since $\boldsymbol{L}_k$ is an inertia tensor, it must be definite positive, and since m_k is positive (from the feasibility condition itself), $(m_k \mathbf{1})$ is definite positive too. These assertions enable the application of the Schur complement condition for positive definiteness (Boyd et al. 1994; Goldman and Ramana 1995) which entails that (3.17) is equivalent to

$$\begin{bmatrix} \boldsymbol{L}_k & \mathbf{S}(\boldsymbol{l}_k)^T \\ \mathbf{S}(\boldsymbol{l}_k) & m_k \mathbf{1} \end{bmatrix} \succ 0 \, . \tag{3.18}$$

The left-hand side block matrix is a function of $\delta_{\mathrm{L}k}$, which can be defined as

$$\boldsymbol{D}_{\mathrm{L}k}(\delta_{\mathrm{L}k}) \equiv \begin{bmatrix} \boldsymbol{L}_k & \mathbf{S}(\boldsymbol{l}_k)^T \\ \mathbf{S}(\boldsymbol{l}_k) & m_k \mathbf{1} \end{bmatrix} \, , \tag{3.19}$$

and in expanded form is given by

$$\boldsymbol{D}_{\mathrm{L}k}(\boldsymbol{\delta}_{\mathrm{L}k}) = \begin{bmatrix} L_{xxk} & L_{xyk} & L_{xzk} & 0 & l_{zk} & -l_{yk} \\ L_{xyk} & L_{yyk} & L_{yzk} & -l_{zk} & 0 & l_{xk} \\ L_{xzk} & L_{yzk} & L_{zzk} & l_{yk} & -l_{xk} & 0 \\ 0 & -l_{zk} & l_{yk} & m_k & 0 & 0 \\ l_{zk} & 0 & -l_{xk} & 0 & m_k & 0 \\ -l_{yk} & l_{xk} & 0 & 0 & 0 & m_k \end{bmatrix} . \tag{3.20}$$

The two conditions given in (2.62) are fully encoded in the single inequality

$$\boldsymbol{D}_{\mathrm{L}k}(\boldsymbol{\delta}_{\mathrm{L}k}) \succ 0 \tag{3.21}$$

since such inequality also implies $m_k > 0$. In fact, (3.21) is a linear matrix inequality (LMI), since it can be decomposed as

$$\boldsymbol{D}_{\mathrm{L}k}(\boldsymbol{\delta}_{\mathrm{L}k}) = \boldsymbol{D}_{\mathrm{L}k0} + \sum_{i=1}^{10} \boldsymbol{D}_{\mathrm{L}ki}\, \boldsymbol{\delta}_{\mathrm{L}ki} , \tag{3.22}$$

where $\boldsymbol{D}_{\mathrm{L}ki}$, for $i = 0, \ldots, 10$, are constant symmetric matrices, and $\boldsymbol{\delta}_{\mathrm{L}ki}$, for $i = 1, \ldots, 10$, are the variables composing $\boldsymbol{\delta}_{\mathrm{L}k}$ as given in (2.18), i.e., the standard inertial parameters. Indeed, the structure of the matrix $\boldsymbol{D}_{\mathrm{L}k}$ is often found in many robot and rigid-body dynamic formulations. This matrix, under the name of *generalized inertia*, is used in the recursive formulation of robot dynamics using Lie Groups proposed by Park et al. (1995). It also appears in some dynamic formulations related to screw theory and using 6D vector representations of motions and forces, as, for example, the spatial vectors developed by Featherstone (2010).

Simple physical feasibility conditions on additional dynamic parameters, as exemplified in (2.64), can also be easily defined as an linear matrix inequality (LMI) by

$$\boldsymbol{D}_{\mathrm{A}k}(\boldsymbol{\delta}_{\mathrm{A}k}) \succ 0 \tag{3.23}$$

with

$$\begin{bmatrix} F_{\mathrm{v}k} & 0 & 0 \\ 0 & F_{\mathrm{c}k} & 0 \\ 0 & 0 & I_{\mathrm{a}k} \end{bmatrix} \succ 0 . \tag{3.24}$$

Finally, it is possible to combine the feasibility conditions for the parameters of all links and joints by using a block-diagonal matrix $\boldsymbol{D}$,

$$\boldsymbol{D}(\delta) \equiv \begin{bmatrix} \boldsymbol{D}_{\mathrm{L}1} & & & & & & & \\ & \boldsymbol{D}_{\mathrm{A}1} & & & & & & \\ & & \ddots & & & & 0 & \\ & & & \boldsymbol{D}_{\mathrm{L}k} & & & & \\ & & & & \boldsymbol{D}_{\mathrm{A}k} & & & \\ & 0 & & & & \ddots & & \\ & & & & & & \boldsymbol{D}_{\mathrm{L}N} & \\ & & & & & & & \boldsymbol{D}_{\mathrm{A}N} \end{bmatrix} , \tag{3.25}$$

into a single LMI:

$$\boldsymbol{D}(\delta) \succ 0 \, . \tag{3.26}$$

This LMI block alone is enough to define the physically feasible parameter set (2.65) as

$$\mathcal{D} = \left\{ \delta \in \mathbb{R}^n : \, \boldsymbol{D}(\delta) \succ 0 \right\} . \tag{3.27}$$

The fact that it is possible to define the set $\mathcal{D}$ as an LMI implies and proves that it is a convex set.

LMI formulations are well known in the field of system and control theory, and have been around since Lyapunov's control stability formulations (Boyd et al. 1994). In this fields, both strict and non-strict LMIs, i.e., definite and semi-definite inequalities, have been well studied. Nevertheless, in recent years, mathematical formalization has been focusing non-strict/semi-definite LMIs, leading to the concept of spectrahedral set and to the emergence of the so-called semidefinite programming (SDP) field.

While the LMI of (3.27) is strict, a relaxed, non-strict, version can also be formulated. The difference would relate to whether absolute zero valued dynamic parameters must be considered physically feasible or not. While the validity of absolutely zero friction constants is perfectly acceptable, the existence of links with zero masses or null inertia tensors may be more questionable.

A way to solve this question can be the definition of a new LMI as

$$\bar{\boldsymbol{D}}(\delta) \succeq 0 \, , \tag{3.28}$$

for

$$\bar{\boldsymbol{D}}(\delta) = \boldsymbol{D}(\delta) - \varepsilon \mathbf{1} \, , \tag{3.29}$$

where ε is a positive scalar sufficiently small to consider

$$\mathcal{D} = \{\boldsymbol{\delta} \in \mathbb{R}^n : \bar{\boldsymbol{D}}(\boldsymbol{\delta}) \succeq 0\} \tag{3.30}$$

equivalent to (3.27). In order to adequate the physical feasibility problem into the semidefinite programming (SDP) mathematical formulation, and benefit from its developments and tools, the non-strict definition of $\mathcal{D}$, (3.30), will be used henceforth. From a practical implementation perspective, (3.29) can be easily addressed taking the numerical precision of computers and SDP solvers into account. A practical example of this will be shown in section §4.5.

The set $\mathcal{D}$, as defined in (3.30), is mathematically called a spectrahedron. Spectrahedra are semialgebraic convex shapes defined by the intersection of positive definite cones with linear affine subspaces, and are the solution spaces of SDP problems. SDP optimization techniques have been under heavy research and are remarkably known for being effective and guaranteeing convergence to the global optimum very efficiently (Vandenberghe and Boyd 1996). SDP unifies and generalizes linear and quadratic programming while maintaining easy solving.

3.2.2 SDP-representable feasible base parameter set

Since spectrahedral sets remain spectrahedral under bijective affine transformations (Goldman and Ramana 1995), the feasible base–dependent parameter set $\mathcal{D}_\beta$ also has an LMI representation. The set $\mathcal{D}_\beta$ can be defined by

$$\mathcal{D}_\beta = \{(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \in \mathbb{R}^n : \bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \succeq 0\}\ , \tag{3.31}$$

where the matrix $\bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$ is obtained by applying a change in variable from $\boldsymbol{\delta}$ to $(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$ to the matrix $\bar{\boldsymbol{D}}(\boldsymbol{\delta})$,

$$\bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) = \bar{\boldsymbol{D}}(\mathfrak{m}_t^{-1}(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}))\ . \tag{3.32}$$

Since $\mathcal{D}_\beta$ is a spectrahedron, the feasible base parameter set $\mathcal{B}$ is a projection of a spectrahedron which can be defined as

$$\mathcal{B} = \{\boldsymbol{\beta} \in \mathbb{R}^{n_\mathrm{b}} : \exists\, \boldsymbol{\delta}_\mathrm{d} \in \mathbb{R}^{n_\mathrm{d}} \mid \bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \succeq 0\}\ . \tag{3.33}$$

Although remaining convex and semialgebraic, projections of spectrahedral sets do not necessarily conserve the spectrahedral form (Goldman and Ramana 1995). Nonetheless, $\mathcal{B}$ is said to be *semidefinite representable* (or SDP representable), being (3.33) its semidefinite representation. In this context, $\mathcal{D}_\beta$ is

said to be the SDP *lift* of $\mathcal{B}$, and $\bar{\boldsymbol{D}}_\beta$ its *lifted* LMI. The fact that the feasible base parameter set is semidefinite representable has great importance since it is still possible to take advantage of SDP techniques (Nesterov and Nemirovskii 1987).

3.3 Base parameters feasibility test through SDP

Although the definition (3.33) does not provides a closed-form test to the feasibility of $\hat{\boldsymbol{\beta}}$ vector, it can be used to devise a feasibility test using SDP. A given $\hat{\boldsymbol{\beta}}$ vector is physically feasible if the LMI $\bar{\boldsymbol{D}}_\beta \succ 0$ partially evaluated at $\hat{\boldsymbol{\beta}}$ defines a non-empty set over $\boldsymbol{\delta}_\mathrm{d}$ space:

$$\hat{\boldsymbol{\beta}} \in \mathcal{B} \quad \Leftrightarrow \quad \{\boldsymbol{\delta}_\mathrm{d} \in \mathbb{R}^{n_\mathrm{d}} : \ \bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})\Big|_{\beta=\hat{\beta}} \succeq 0\} \neq \emptyset \,. \tag{3.34}$$

This is a typical feasibility SDP problem which can be defined as

$$\begin{aligned} \text{find} \quad & \boldsymbol{\delta}_\mathrm{d} \\ \text{subject to} \quad & \bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})\Big|_{\beta=\hat{\beta}} \succeq 0 \,. \end{aligned} \tag{3.35}$$

This kind of problem can be seen as a SDP optimization problem where the cost function is not relevant. If it is possible to find a solution to this problem, then the given $\hat{\boldsymbol{\beta}}$ is physically feasible.

3.4 Distance to the physical feasible parameter set

Another method to verify the physical feasibility of a given $\hat{\boldsymbol{\beta}}$ is to search for a feasible $(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$ solution whose $\boldsymbol{\beta}$ part has minimal distance to $\hat{\boldsymbol{\beta}}$. If the distance can be minimized down to zero then it means that the $\hat{\boldsymbol{\beta}}$ is feasible. The distance minimization problem can be defined as

$$\begin{aligned} \underset{(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})}{\text{minimize}} \quad & \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\| \\ \text{subject to} \quad & \bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \succeq 0 \,, \end{aligned} \tag{3.36}$$

for which an SDP problem form can be devised. First, the problem is defined as minimization of a slack variable u by

$$\begin{aligned} \underset{(u, \boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})}{\text{minimize}} \quad & u \\ \text{subject to} \quad & u \geq \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|^2 \\ & \bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d}) \succeq 0 \,. \end{aligned} \tag{3.37}$$

Then, the condition $u \geq \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|^2$ is rewritten as an LMI:

$$\begin{aligned} & \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|^2 \leq u \\ \Leftrightarrow \quad & u - (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \geq 0 \\ \Leftrightarrow \quad & \boldsymbol{U}_{\hat{\boldsymbol{\beta}}}(u, \boldsymbol{\beta}) \succeq 0 \,, \end{aligned} \tag{3.38}$$

where the matrix $\boldsymbol{U}_{\hat{\boldsymbol{\beta}}}(u, \boldsymbol{\beta})$,

$$\boldsymbol{U}_{\hat{\boldsymbol{\beta}}}(u, \boldsymbol{\beta}) \equiv \begin{bmatrix} u & (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T \\ \hat{\boldsymbol{\beta}} - \boldsymbol{\beta} & \mathbf{1}_{n_\mathrm{b}} \end{bmatrix} \,, \tag{3.39}$$

is obtained through the application of Schur complement conditions. Hence, problem (3.37) can be formulated as an SDP problem by

$$\begin{aligned} \underset{(u, \boldsymbol{\beta}, \delta_\mathrm{d})}{\text{minimize}} \quad & u \\ \text{subject to} \quad & \boldsymbol{U}_{\hat{\boldsymbol{\beta}}}(u, \boldsymbol{\beta}) \succeq 0 \\ & \bar{\boldsymbol{D}}_\beta(\boldsymbol{\beta}, \delta_\mathrm{d}) \succeq 0 \,. \end{aligned} \tag{3.40}$$

Being $(u', \boldsymbol{\beta}', \delta_\mathrm{d}')$ the solution to (3.40), δ_d' is the feasible base parameter vector closest to $\hat{\boldsymbol{\beta}}$, and u' is the respective distance. In the case u' is zero, then $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}'$ and $\hat{\boldsymbol{\beta}}$ is a physically feasible vector. This method can be used to correct an infeasible base parameter estimation $\hat{\boldsymbol{\beta}}$ into a feasible $\boldsymbol{\beta}'$, while not relying on any regression data. The value of u' provides a measure for the infeasibility of $\hat{\boldsymbol{\beta}}$. However, it must be taken into account that the distance is dependent on the chosen base parameter grouping ($\boldsymbol{K}$), since distances are not linearly mapped between different base spaces.

3.5 Base parameter regression constrained to physically feasible solutions

The SDP methods devised above can be further extended to include the dynamic parameter regression itself. If the regression minimization is performed inside the feasible parameters set only, then the obtained solution will be feasible.

3.5.1 OLS regression as an SDP problem

In a similar fashion to the distance minimization of (3.38), the regression by OLS, given by (2.55), can be formulated a an SDP problem. Therefore (2.55)

can be written as

$$\begin{aligned} \underset{(u,\boldsymbol{\beta})}{\text{minimize}} \quad & u \\ \text{subject to} \quad & u \geq \|\boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta}\|^2 \,. \end{aligned} \tag{3.41}$$

Applying the Schur complement condition for positive definiteness once again,

$$\begin{aligned} & \|\boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta}\|^2 \leq u \\ \Leftrightarrow \quad & u - (\boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta})^T(\boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta}) \geq 0 \\ \Leftrightarrow \quad & \boldsymbol{U}_{u,\boldsymbol{\omega}}(\boldsymbol{\beta}) \succeq 0 \,, \end{aligned} \tag{3.42}$$

with

$$\boldsymbol{U}_{\boldsymbol{\omega}}(u, \boldsymbol{\beta}) \equiv \begin{bmatrix} u & (\boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta})^T \\ \boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta} & \mathbf{1}_{NS} \end{bmatrix} \,. \tag{3.43}$$

Hence, the solution of the SDP minimization problem

$$\begin{aligned} \underset{(u,\boldsymbol{\beta})}{\text{minimize}} \quad & u \\ \text{subject to} \quad & \boldsymbol{U}_{\boldsymbol{\omega}}(u, \boldsymbol{\beta}) \succeq 0 \,. \end{aligned} \tag{3.44}$$

is the same $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ solution given by the analytic method (2.58).

3.5.2 LMI size reduction

Although this is a valid SDP problem, its LMI matrix has a size of $NS{+}1 \times NS{+}1$, a very high number which can lead to slow solving times. Nevertheless, this issue can be tackled through a prior decomposition of the matrix $\boldsymbol{W}$. Applying a QR orthogonal-triangular decomposition to the regression matrix $\boldsymbol{W}$, we obtain

$$\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{R} \,, \tag{3.45}$$

which, since $\boldsymbol{W}$ has more rows than columns, can be split into

$$\boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \mathbf{0} \end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}_1 \,, \tag{3.46}$$

where $\boldsymbol{R}_1$ is a $n_{\text{b}} \times n_{\text{b}}$ upper-triangular matrix. Since matrix $\boldsymbol{Q}$ is orthogonal ($\boldsymbol{Q}^T\boldsymbol{Q} = \mathbf{1}$), vectors maintain their norm when left-multiplied by its transpose.

Therefore the regression error $\|\boldsymbol{\epsilon}\|^2$ can be rewritten as

$$\begin{aligned}\|\boldsymbol{\epsilon}\|^2 &= \|\boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta}\|^2 \\ &= \|\boldsymbol{Q}^T(\boldsymbol{\omega} - \boldsymbol{W}\boldsymbol{\beta})\|^2 \\ &= \|\boldsymbol{Q}^T\boldsymbol{\omega} - \boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}\boldsymbol{\beta}\|^2 \\ &= \left\| \begin{bmatrix} \boldsymbol{Q}_1^T \\ \boldsymbol{Q}_2^T \end{bmatrix} \boldsymbol{\omega} - \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\beta} \right\|^2 \\ &= \left\| \begin{bmatrix} \boldsymbol{Q}_1^T\boldsymbol{\omega} - \boldsymbol{R}_1\boldsymbol{\beta} \\ \boldsymbol{Q}_2^T\boldsymbol{\omega} \end{bmatrix} \right\|^2 ,\end{aligned} \tag{3.47}$$

and finally as

$$\|\boldsymbol{\epsilon}\|^2 = \|\boldsymbol{\rho}_2\|^2 + \|\boldsymbol{\rho}_1 - \boldsymbol{R}_1\boldsymbol{\beta}\|^2 \tag{3.48}$$

with

$$\begin{aligned}\boldsymbol{\rho}_1 &\equiv \boldsymbol{Q}_1^T\boldsymbol{\omega} \\ \boldsymbol{\rho}_2 &\equiv \boldsymbol{Q}_2^T\boldsymbol{\omega} .\end{aligned} \tag{3.49}$$

The term $\|\boldsymbol{\rho}_2\|^2$ equals the minimal regression error, and it is constant and not dependent on $\boldsymbol{\beta}$. From (3.41) and (3.48),

$$u - \|\boldsymbol{\rho}_2\|^2 \geq \|\boldsymbol{\rho}_1 - \boldsymbol{R}_1\boldsymbol{\beta}\|^2 , \tag{3.50}$$

hence, the SDP problem from (3.44) can be reformulated as

$$\begin{aligned}&\underset{(u,\boldsymbol{\beta})}{\text{minimize}} \quad u \\ &\text{subject to} \quad \boldsymbol{U}_{\rho_1}(u,\boldsymbol{\beta}) \succeq 0 ,\end{aligned} \tag{3.51}$$

where

$$\boldsymbol{U}_{\rho_1}(u,\boldsymbol{\beta}) \equiv \begin{bmatrix} u - \|\boldsymbol{\rho}_2\|^2 & (\boldsymbol{\rho}_1 - \boldsymbol{R}_1\boldsymbol{\beta})^T \\ \boldsymbol{\rho}_1 - \boldsymbol{R}_1\boldsymbol{\beta} & \boldsymbol{1}_N \end{bmatrix} . \tag{3.52}$$

In this case, the LMI matrix $\boldsymbol{U}_{\rho_1}$ has a size of $N{+}1 \times N{+}1$ only, being much more SDP solver friendly than $\boldsymbol{U}_{\omega}$.

3.5.3 SDP regression constrained to the physically feasible space

A new regression SDP problem can now be defined by including the feasible base–dependent parameter LMI to (3.51):

$$\begin{aligned}&\underset{(u,\boldsymbol{\beta},\delta_d)}{\text{minimize}} \quad u \\ &\text{subject to} \quad \boldsymbol{U}_{\rho_1}(u,\boldsymbol{\beta}) \succeq 0 \\ &\qquad\qquad \bar{\boldsymbol{D}}_{\beta}(\boldsymbol{\beta},\delta_d) \succeq 0 .\end{aligned} \tag{3.53}$$

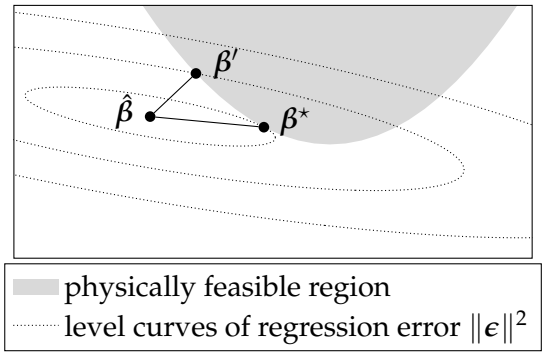

Figure 3.2: Regression with a non-feasible $\hat{\beta}$ (2D illustration). The always feasible and optimal solution $\beta^\star$ has lower regression error than the feasible solution β' closer to $\hat{\beta}$.

This problem searches for (u, β, δ_d) solutions which minimize the regression error, however, such solutions will be constrained to the physically feasible parameter set $\mathcal{D}_\beta$.

Being $(u^\star, \beta^\star, \delta_d^\star)$ the optimal solution to (3.53), the vector $\beta^\star$ is the physically feasible base parameters which best fits the regression data. The value $u^\star$ is the respective regression error,

$$u^\star = \|\omega - W\beta^\star\|^2 . \tag{3.54}$$

If $\beta^\star$ is equal to the unconstrained OLS estimation, $\hat{\beta}_{\text{OLS}}$, given by (2.58), then it means that $\hat{\beta}_{\text{OLS}}$ is feasible and that the regression data is well conditioned with respect to feasibility. In this case, being β'_{OLS} the solution to the problem (3.40) with the $\hat{\beta}$ input equal to $\hat{\beta}_{\text{OLS}}$, then $\beta'_{\text{OLS}} = \hat{\beta}_{\text{OLS}}$ will also be verified. If the solution $\hat{\beta}_{\text{OLS}}$ is not feasible, $\beta^\star_{\text{OLS}}$ is not necessarily equal to β'_{OLS}, as figure 3.2 illustrates.

3.6 Implementation in the standard parameter space

Although the above methods are written in the base–dependent space, it is possible to rewrite them in the standard parameter space.

The verification of the physical feasibility of a given standard parameter estimation $\hat{\delta}$ is directly performed by verifying the positive definiteness of the matrix $D(\delta)$ (see section §3.2.1). However, it may be possible to an infeasible solution to still entail a dynamic model with a "feasible behavior".

This happens when that infeasible solution belongs to a virtual parameter set (see (2.60)) that partially intersects the physically feasible set (for example, the virtual parameter set $\mathcal{V}_{\hat{\beta}_f}$ of figure 2.3). The model behavior is dictated by the feasibility of the effective part of the parameters, i.e., the base parameters, which in such cases is feasible. We can then say that $\hat{\delta}$ estimations can be physically infeasible, feasible, or "virtually feasible". The method for base parameter feasibility test (section §3.3) can be modified to check if a given standard parameter estimation $\hat{\delta}$ is feasible or "virtually feasible", or if it is completely infeasible, by simply modifying the problem of (3.35) to

$$\begin{aligned} \text{find} \quad & \delta_{\mathrm{d}} \\ \text{subject to} \quad & \bar{\boldsymbol{D}}_{\beta}(\boldsymbol{K}\hat{\delta}, \delta_{\mathrm{d}}) \succeq 0 \,. \end{aligned} \tag{3.55}$$

If some δ_{d} can be found, then $\hat{\delta}$ is either feasible or "virtually feasible".

The method to find the closest feasible parameter solution devised in section §3.4 can also be modified to find the closest feasible standard parameters to some given standard parameter estimation $\hat{\delta}$. For that, the problem of (3.40) can be rewritten as

$$\begin{aligned} \underset{(u,\delta)}{\text{minimize}} \quad & u \\ \text{subject to} \quad & \boldsymbol{U}_{\hat{\delta}}(\delta) \succeq 0 \\ & \bar{\boldsymbol{D}}(\delta) \succeq 0 \,, \end{aligned} \tag{3.56}$$

with $\bar{\boldsymbol{D}}(\delta)$ given by (3.29), and

$$\boldsymbol{U}_{\hat{\delta}}(\delta) \equiv \begin{bmatrix} u & (\hat{\delta} - \delta)^T \\ \hat{\delta} - \delta & \mathbf{1}_n \end{bmatrix} \,. \tag{3.57}$$

A u solution higher than zero means that $\hat{\delta}$ is not feasible, however, it can be "virtually feasible". This method can be used to find a feasible standard parameter vector which entails the same model of a given "virtually feasible" estimation.

The feasible regression problem of section §3.5 can be formulated in the standard parameter space as

$$\begin{aligned} \underset{(u,\delta)}{\text{minimize}} \quad & u \\ \text{subject to} \quad & \boldsymbol{U}_{\rho_1}(u, \boldsymbol{K}\delta) \succeq 0 \\ & \bar{\boldsymbol{D}}(\delta) \succeq 0 \,, \end{aligned} \tag{3.58}$$

with $\boldsymbol{U}_{\rho_1}$ given by (3.52). For this version of the regression, the optimal solution $\delta^\star$ will be not unique. Nevertheless, it will be equivalent to the unique optimal base parameter solution $\boldsymbol{\beta}^\star$, since $\boldsymbol{\beta}^\star = \boldsymbol{K}\delta^\star$. Being the solutions equivalent, the choice of parameter space for this method becomes a matter of how one can turn to be more practical than the other.

4 WAM ROBOT DYNAMIC MODEL IDENTIFICATION

4.1 The seven-link WAM robot

The WAM^{TM} Arm is a lightweight robot developed by the North American company Barrett Technology, Inc. A photograph of the WAM robot is shown in figure 4.1. This manipulator has some distinct features that sets it apart from classical industry robots. Some of the key features the WAM robot has are:

- seven revolution joints (7-DOF)
- anthropomorphic configuration,
- cable driven joints (except for the seventh joint),
- differential joints (two pairs),
- low motor–joint ratios.

The differential joints enable the placement of the motors closer to the robot base, allowing less self payload, thus lighter links. The combination of cable driven and lower "gear" ratios enable low friction, no backlash, and very high backdrivability. All of these characteristics contribute to a robot more suitable to force-controlled and compliant tasks. The joints are disposed similarly to a human arm: three shoulder joints, one elbow joint and three wrist joints. Figure 4.2 depicts the geometric model of the WAM, and the DH parameters are given in table 4.1.

4.2 WAM robot modeling

With respect to the dynamic model, the WAM manufacturer provides datasheet inertial parameters from CAD models. Such data includes not only link body inertias, but also cable pulleys and rotor inertia information. In this work, for identification purposes, the considered dynamic model includes: link inertias, joint viscous and Coulomb frictions, joint torque offset, and drive chain

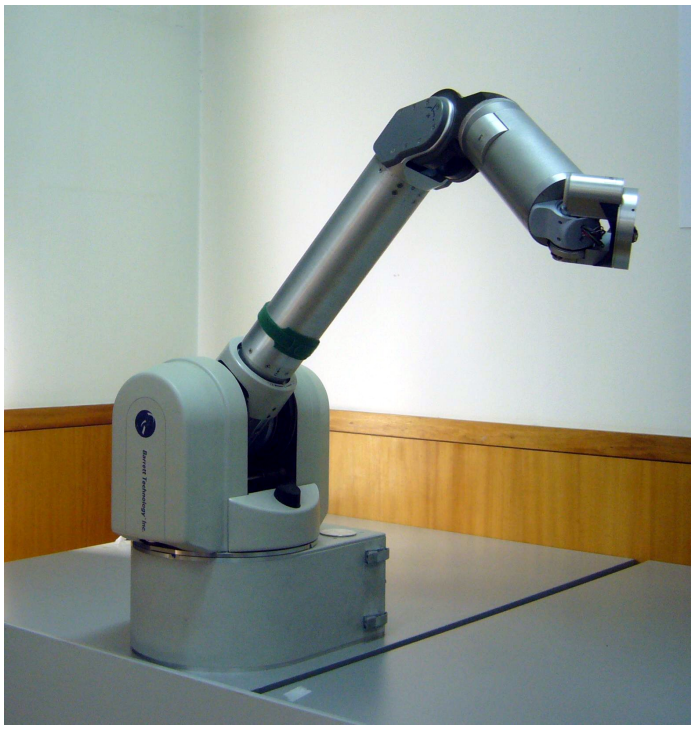

Figure 4.1: Barrett Technology, Inc. WAM^{TM} Arm.

Table 4.1: DH parameters of the WAM robot (standard notation).

Joint k	α_k	a_k	d_k	θ_k
1	$-\pi/2$	0	0	q_1
2	$\pi/2$	0	0	q_2
3	$-\pi/2$	0.045	0.55	q_3
4	$\pi/2$	-0.045	0	q_4
5	$-\pi/2$	0	0.3	q_5
6	$\pi/2$	0	0	q_6
7	0	0	0.06	q_7

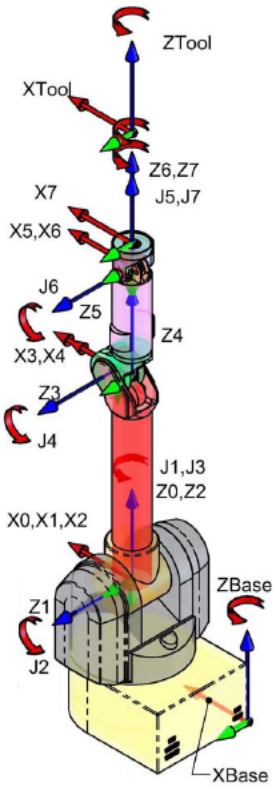

Figure 4.2: Geometry schematic of the WAM robot.

inertias. The WAM has two pairs of differential joints, where two motors have coupled actuation on two joints: joints 2 and 3 are deferentially actuated by motors 2 and 3, and joints 5 and 6 are deferentially actuated by motors 5 and 6. The differential effects are encoded in the motor–joint transmission rates given by

$$\boldsymbol{\tau} = \boldsymbol{T}\,\boldsymbol{\tau}_{\mathrm{m}} \tag{4.1}$$

and

$$\dot{\boldsymbol{q}}_{\mathrm{m}} = \boldsymbol{T}^{T}\,\dot{\boldsymbol{q}}\,, \tag{4.2}$$

where $\boldsymbol{\tau}_{\mathrm{m}}$ are the joint forces seen in the motor-side, $\dot{\boldsymbol{q}}_{\mathrm{m}}$ is the vector of motor velocities, and $\boldsymbol{T}$ is a block-diagonal matrix. $\boldsymbol{T}$ can be split into a diagonal matrix $\boldsymbol{T}_{\mathrm{r}}$ containing motor to joint ratios and a block-diagonal matrix $\boldsymbol{T}_{\mathrm{c}}$ containing coupling ratios,

$$\boldsymbol{T} = \boldsymbol{T}_{\mathrm{r}}\,\boldsymbol{T}_{\mathrm{c}}\,. \tag{4.3}$$

For the WAM, these matrices can be obtained for the datasheet:

$$\boldsymbol{T}_{\mathrm{r}}^{\mathrm{WAM}} = \begin{bmatrix} N_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & N_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & N_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & N_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & N_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & N_7 \end{bmatrix} , \tag{4.4}$$

with $N_1 = 42$, $N_2 = N_3 = 28.25$, $N_4 = 18$, $N_5 = N_6 = 10.27$ and $N_7 = 14.93$, and

$$\boldsymbol{T}_{\mathrm{c}}^{\mathrm{WAM}} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & \frac{-1}{n_3} & \frac{-1}{n_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{-1}{n_6} & \frac{1}{n_6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} , \tag{4.5}$$

with $n_3 = 1.68$ and $n_6 = 1$. To proper model the differential transmission, the friction is split into motor and joint sides, and the rotor inertia is modeled on the motor-side. Hence, the term of additional dynamics $\boldsymbol{h}$ of (2.9) is defined for the WAM, using (4.1), by

$$\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{f}_{\mathrm{j}}(\dot{\boldsymbol{q}}) + \boldsymbol{T}\, \boldsymbol{h}_{\mathrm{m}}(\dot{\boldsymbol{q}}_{\mathrm{m}}, \ddot{\boldsymbol{q}}_{\mathrm{m}}) , \tag{4.6}$$

where $\boldsymbol{f}_{\mathrm{j}}$ is a function of joint-side frictions and $\boldsymbol{h}_{\mathrm{m}}$ is the term of motor-side dynamics given by

$$\boldsymbol{h}_{\mathrm{m}}(\dot{\boldsymbol{q}}_{\mathrm{m}}, \ddot{\boldsymbol{q}}_{\mathrm{m}}) = \boldsymbol{f}_{\mathrm{m}}(\dot{\boldsymbol{q}}_{\mathrm{m}}) + \boldsymbol{I}_{\mathbf{a}\mathrm{m}} \ddot{\boldsymbol{q}}_{\mathrm{m}} , \tag{4.7}$$

where $\boldsymbol{f}_{\mathrm{m}}$ is a function of motor-side frictions and $\boldsymbol{I}_{\mathbf{a}\mathrm{m}}$ is a matrix of rotor inertias. Through (4.2), $\boldsymbol{h}_{\mathrm{m}}$ can be defined in terms of joint-side velocity and acceleration:

$$\boldsymbol{h}_{\mathrm{m}}(\dot{\boldsymbol{q}}_{\mathrm{m}}, \ddot{\boldsymbol{q}}_{\mathrm{m}}) = \boldsymbol{h}_{\mathrm{m}}(\boldsymbol{T}^T \dot{\boldsymbol{q}}, \boldsymbol{T}^T \ddot{\boldsymbol{q}}) = \boldsymbol{f}_{\mathrm{m}}(\boldsymbol{T}^T \dot{\boldsymbol{q}}) + \boldsymbol{I}_{\mathbf{a}\mathrm{m}} \boldsymbol{T}^T \ddot{\boldsymbol{q}} , \tag{4.8}$$

thus (4.6) expands to

$$\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{f}_{\mathrm{j}}(\dot{\boldsymbol{q}}) + \boldsymbol{T} \left(\boldsymbol{f}_{\mathrm{m}}(\boldsymbol{T}^T \dot{\boldsymbol{q}}) + \boldsymbol{I}_{\mathbf{a}\mathrm{m}} \boldsymbol{T}^T \ddot{\boldsymbol{q}} \right) . \tag{4.9}$$

The matrix $\boldsymbol{I}_{\mathbf{a}\mathrm{m}}$ is a diagonal matrix containing the rotor inertia $I_{\mathrm{a}\mathrm{m}k}$ of each motor k. The gyroscopic effects of the rotor inertias are not considered. The friction is modeled by the vector functions $\boldsymbol{f}_{\mathrm{j}}$ and $\boldsymbol{f}_{\mathrm{m}}$ defined as follows: the k-th element of function $\boldsymbol{f}_{\mathrm{j}}$ models the joint-side friction of joint k,

$$f_{\mathrm{j}k}(\dot{q}_k) = F_{\mathrm{v}\mathrm{j}k}\,\dot{q}_k + F_{\mathrm{c}\mathrm{j}k}\,\mathrm{sign}(\dot{q}_k) + F_{\mathrm{o}\mathrm{j}k}\;, \tag{4.10}$$

and the k-th element of $\boldsymbol{f}_{\mathrm{m}}$ models the motor-side friction of motor k,

$$f_{\mathrm{m}k}(\dot{q}_{\mathrm{m}k}) = F_{\mathrm{v}\mathrm{m}k}\,\dot{q}_{\mathrm{m}k} + F_{\mathrm{c}\mathrm{m}k}\,\mathrm{sign}(\dot{q}_{\mathrm{m}k}) + F_{\mathrm{o}\mathrm{m}k}\;. \tag{4.11}$$

The $F_{\mathrm{v}*}$ symbols denote the viscous friction parameters, the $F_{\mathrm{c}*}$ symbols denote the Coulomb friction parameters, and the $F_{\mathrm{o}*}$ symbols denote the Coulomb friction offsets which, for the motor-side, also include motor torque offsets (due to possible current offset). At the joints with no coupling, the friction parameters from joint and motor sides have proportional effects and can only be identified in combinations. Although such combinations can be obtained by hand, in this work they are computed by the method employed to find the general base dynamic parameters. Due to the motor–joint ratios, the regressor columns corresponding to motor-side friction and inertia have a too high magnitude which degrade the regressor condition number. To overcome this issue, the ratio matrix $\boldsymbol{T}_{\mathrm{r}}^{\mathrm{WAM}}$ of (4.3) is replaced by the identity matrix,

$$\boldsymbol{T} = \mathbf{1}\,\boldsymbol{T}_{\mathrm{c}}^{\mathrm{WAM}}\;, \tag{4.12}$$

thus the identified motor-side parameters are considered as being measured at the joint-side, not considering joint couplings though.

The final dynamic parameter vector $\boldsymbol{\delta}$ for the WAM is defined by

$$\boldsymbol{\delta} \equiv \begin{bmatrix} \boldsymbol{\delta}_{\mathrm{L}1}^{T} & \boldsymbol{\delta}_{\mathrm{A}1}^{T} & \dots & \boldsymbol{\delta}_{\mathrm{L}7}^{T} & \boldsymbol{\delta}_{\mathrm{A}7}^{T} \end{bmatrix}^{T}\;, \tag{4.13}$$

where for each link k, ranging from 1 to $N = 7$,

$$\boldsymbol{\delta}_{\mathrm{L}k} \equiv \begin{bmatrix} L_{xxk} & L_{xyk} & L_{xzk} & L_{yyk} & L_{yzk} & L_{zzk} & l_{xk} & l_{yk} & l_{zk} & m_k \end{bmatrix}^{T}\;, \tag{4.14}$$

and

$$\boldsymbol{\delta}_{\mathrm{A}k} = \begin{bmatrix} F_{\mathrm{v}\mathrm{j}k} & F_{\mathrm{c}\mathrm{j}k} & F_{\mathrm{o}\mathrm{j}k} & F_{\mathrm{v}\mathrm{m}k} & F_{\mathrm{c}\mathrm{m}k} & F_{\mathrm{o}\mathrm{m}k} & I_{\mathrm{a}\mathrm{m}k} \end{bmatrix}^{T}\;. \tag{4.15}$$

The equations for the dynamic model are devised by the *Python* package *SymPyBotics* (Sousa 2014b), an open source symbolic robotics toolbox based

on the *SymPy* computer algebra system, which has been developed by the authors for the present work. *Python* and *C* source codes for the whole inverse dynamic equations, as well as for each of its terms represented in (2.9), are generated by the *SymPyBotics* toolbox. A base dynamic parameter grouping is computed through to the numerical method developed by Gautier (1991), entailing a total of 76 base parameters, $\boldsymbol{\beta}$, presented in table 4.2.

Table 4.2: Base dynamic parameters of the WAM robot.

$\boldsymbol{\beta}$	corresponding linear combination
β_1	$L_{yy1} + I_{\mathrm{am1}} + L_{zz2}$
β_2	$F_{\mathrm{vj1}} + F_{\mathrm{vm1}}$
β_3	$F_{\mathrm{cj1}} + F_{\mathrm{cm1}}$
β_4	$F_{\mathrm{oj1}} - F_{\mathrm{om1}}$
β_5	$L_{xx2} - L_{zz2} + L_{zz3} - 1.1\, l_{y3} + 0.300475\, (m_3 + m_4 + m_5 + m_6 + m_7)$
β_6	L_{xy2}
β_7	L_{xz2}
β_8	$L_{yy2} + L_{zz3} - 1.1\, l_{y3} + 0.300475\, (m_3 + m_4 + m_5 + m_6 + m_7)$
β_9	L_{yz2}
β_{10}	l_{x2}
β_{11}	$l_{z2} - l_{y3} + 0.55\, (m_3 + m_4 + m_5 + m_6 + m_7)$
β_{12}	$F_{\mathrm{vj2}} + 2\, F_{\mathrm{vm3}}$
β_{13}	F_{cj2}
β_{14}	$F_{\mathrm{oj2}} + 1.68\, F_{\mathrm{oj3}} - 2\, F_{\mathrm{om3}}$
β_{15}	$F_{\mathrm{vm2}} - F_{\mathrm{vm3}}$
β_{16}	F_{cm2}
β_{17}	$F_{\mathrm{om2}} - 1.68\, F_{\mathrm{oj3}} + F_{\mathrm{om3}}$
β_{18}	I_{am2}
β_{19}	$L_{xx3} - L_{zz3} + 0.002025\, m_3 + L_{zz4}$
β_{20}	$L_{xy3} - 0.045\, l_{y3}$
β_{21}	L_{xz3}
β_{22}	$L_{yy3} - 0.002025\, m_3 + L_{zz4} - 0.00405\, (m_4 + m_5 + m_6 + m_7)$
β_{23}	L_{yz3}
β_{24}	$l_{x3} + 0.045\, (m_3 + m_4 + m_5 + m_6 + m_7)$
β_{25}	$l_{z3} + l_{y4}$
β_{26}	$F_{\mathrm{vj3}} + 0.70861678\, F_{\mathrm{vm3}}$
β_{27}	F_{cj3}
β_{28}	F_{cm3}
β_{29}	I_{am3}
β_{30}	$L_{xx4} - L_{zz4} + 0.002025\, m_4 + L_{zz5} - 0.6\, l_{y5} + 0.092025\, (m_5 + m_6 + m_7)$

continues

Table 4.2 continued

β	corresponding linear combination
β_{31}	$L_{xy4} + 0.045\, l_{y4}$
β_{32}	L_{xz4}
β_{33}	$L_{yy4} - 0.002025\, m_4 + L_{zz5} - 0.6\, l_{y5} + 0.087975\, (m_5 + m_6 + m_7)$
β_{34}	L_{yz4}
β_{35}	$l_{x4} - 0.045\, (m_4 + m_5 + m_6 + m_7)$
β_{36}	$l_{z4} - l_{y5} + 0.3\, (m_5 + m_6 + m_7)$
β_{37}	$F_{\mathrm{vj4}} + F_{\mathrm{vm4}}$
β_{38}	$F_{\mathrm{cj4}} + F_{\mathrm{cm4}}$
β_{39}	$F_{\mathrm{oj4}} + F_{\mathrm{om4}}$
β_{40}	I_{am4}
β_{41}	$L_{xx5} - L_{zz5} + L_{zz6}$
β_{42}	L_{xy5}
β_{43}	L_{xz5}
β_{44}	$L_{yy5} + L_{zz6}$
β_{45}	L_{yz5}
β_{46}	l_{x5}
β_{47}	$l_{z5} + l_{y6}$
β_{48}	$F_{\mathrm{vj5}} + 2\, F_{\mathrm{vm6}}$
β_{49}	F_{cj5}
β_{50}	$F_{\mathrm{oj5}} + F_{\mathrm{oj6}} + 2\, F_{\mathrm{om6}}$
β_{51}	$F_{\mathrm{vm5}} - F_{\mathrm{vm6}}$
β_{52}	F_{cm5}
β_{53}	$F_{\mathrm{om5}} - F_{\mathrm{oj6}} - F_{\mathrm{om6}}$
β_{54}	I_{am5}
β_{55}	$L_{xx6} - L_{zz6} + L_{yy7} + 0.12\, l_{z7} + 0.0036\, m_7$
β_{56}	L_{xy6}
β_{57}	L_{xz6}
β_{58}	$L_{yy6} + L_{yy7} + 0.12\, l_{z7} + 0.0036\, m_7$
β_{59}	L_{yz6}
β_{60}	l_{x6}
β_{61}	$l_{z6} + l_{z7} + 0.06\, m_7$
β_{62}	$F_{\mathrm{vj6}} + 2\, F_{\mathrm{vm6}}$
β_{63}	F_{cj6}
β_{64}	F_{cm6}
β_{65}	I_{am6}
β_{66}	$L_{xx7} - L_{yy7}$
β_{67}	L_{xy7}
β_{68}	L_{xz7}

continues

Table 4.2 continued

$\boldsymbol{\beta}$	corresponding linear combination
β_{69}	L_{yz7}
β_{70}	L_{zz7}
β_{71}	l_{x7}
β_{72}	l_{y7}
β_{73}	$F_{\mathrm{vj}7} + F_{\mathrm{vm}7}$
β_{74}	$F_{\mathrm{cj}7} + F_{\mathrm{cm}7}$
β_{75}	$F_{\mathrm{oj}7} - F_{\mathrm{om}7}$
β_{76}	$I_{\mathrm{am}7}$

4.3 Regression data set

4.3.1 WAM control for regression trajectory tracking

Unlike the majority of robots, the WAM robot is directly controlled in torque. Each motor receives torque set points which are converted into current set points through a constant predefined ratio. On the other hand, the WAM provides no torque or current measurements, thus good torque to current conversion and good current tracking control is assumed. The torque controlled joints enable computed torque control and nonlinear feedback linearization. This is used for the controller to track an excitation trajectory and obtain regression data.

The nonlinear feedback linearization is achieved by canceling the nonlinear terms of the dynamic equation (2.9). The commanded torque, $\boldsymbol{\tau}_{\mathrm{c}}$, is computed by

$$\boldsymbol{\tau}_{\mathrm{c}} = \hat{\boldsymbol{c}}(\boldsymbol{q}, \hat{\dot{\boldsymbol{q}}}) + \hat{\boldsymbol{g}}(\boldsymbol{q}) + \hat{\boldsymbol{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{\tau}_{\mathrm{pd}} \,, \tag{4.16}$$

where $\hat{\dot{\boldsymbol{q}}}$ is an estimation of joint velocities through numerical differences, $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{c}}$ and $\hat{\boldsymbol{g}}$ are estimations of $\boldsymbol{M}$, $\boldsymbol{c}$ and $\boldsymbol{g}$, respectively, $\ddot{\boldsymbol{q}}_{\mathrm{r}}$ is the reference trajectory acceleration, and $\boldsymbol{\tau}_{\mathrm{pd}}$ is a proportional–derivative (PD) controller command. The estimations of $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{c}}$ and $\hat{\boldsymbol{g}}$ are performed using the datasheet CAD-estimated dynamic parameters. The additional dynamics term $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ of (2.9) is not canceled since no prior information about friction constants is known. Neglecting modeling and estimation errors, by (2.9) and (4.16), the dynamic equation can be reduced to

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{M}\ddot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{\tau}_{\mathrm{pd}} \,. \tag{4.17}$$

To achieve good trajectory tracking, a PD position controller with command

$$\boldsymbol{\tau}_{\mathrm{pd}} = \boldsymbol{K}_{\mathrm{p}}(\boldsymbol{q}_{\mathrm{r}} - \boldsymbol{q}) + \boldsymbol{K}_{\mathrm{d}}(\dot{\boldsymbol{q}}_{\mathrm{r}} - \hat{\dot{\boldsymbol{q}}}) \tag{4.18}$$

is implemented, being $\boldsymbol{K}_{\mathrm{p}}$ and $\boldsymbol{K}_{\mathrm{d}}$ diagonal matrices with the proportional and derivative gains, respectively. The PD gains, which are given in table 4.3,

Table 4.3: PD joint position control gains for the WAM robot regression trajectory tracker.

Joint k	$K_{\mathrm{p}k}$	$K_{\mathrm{d}k}$
1	200.0	5.0
2	200.0	5.0
3	40.0	1.0
4	40.0	1.0
5	3.0	0.2
6	3.0	0.2
7	0.5	0.0

are tuned a joint at a time with a trial and error procedure seeking the best position tracking while maintaining smooth robot reaction. It would be possible to feed back the PD commands through the inertia matrix along with the feed-forward signal of $\ddot{\boldsymbol{q}}_{\mathrm{r}}$, hence effectively implementing a PD controller over a double integrator plant as presented in section §2.4.1. However, that is not implemented at this stage in order to avoid coupling of control commands between joints, and to allow the independent tune of the gains. The overall control scheme is depicted in figure 4.3. Its implementation runs at a sampling period of $Ts = 0.001\,\mathrm{s}$.

4.3.2 Excitation trajectories generation

To generate excitation trajectories the Fourier series formulation proposed by Swevers et al. (1997) is employed. This formulation has the advantages of being easy to generate and having analytic derivative. Each joint k trajectory is defined as a function of time t by

$$q_{\mathrm{r}k}(t) = \sum_{i=1}^{n_{\mathrm{H}}} \frac{a_{ik}}{\omega_{\mathrm{f}}\, i} \sin(\omega_{\mathrm{f}}\, i\, t) - \frac{b_{ik}}{\omega_{\mathrm{f}}\, i} \cos(\omega_{\mathrm{f}}\, i\, t) + q_{0k} \ , \tag{4.19}$$

where ω_{f} is the fundamental angular frequency of the Fourier series, n_{H} is the number of harmonics, a_{ik} and b_{ik} are the Fourier coefficients, and q_{0k} is

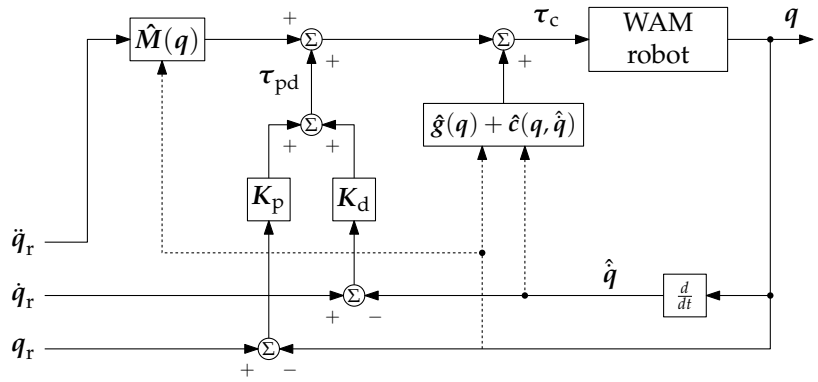

Figure 4.3: PD joint position control scheme of WAM robot with gravity and Coriolis forces compensation and acceleration feedforward for regression data collection. $\boldsymbol{q}_\mathrm{r}$ is the joint reference position.

the initial joint position. Joint velocities and accelerations are devised directly from (4.19):

$$\begin{aligned} \dot{q}_{\mathrm{r}k}(t) &= \sum_{i=1}^{n_\mathrm{H}} a_{ik} \cos(\omega_\mathrm{f}\, i\, t) - b_{ik} \sin(\omega_\mathrm{f}\, i\, t) \\ \ddot{q}_{\mathrm{r}k}(t) &= \sum_{i=1}^{n_\mathrm{H}} -a_{ik}\, \omega_\mathrm{f}\, i \sin(\omega_\mathrm{f}\, i\, t) + b_{ik}\, \omega_\mathrm{f}\, i \cos(\omega_\mathrm{f}\, i\, t)\ . \end{aligned} \tag{4.20}$$

To obtain the reference trajectory, the values of n_H and ω_f are fixed to

$$n_\mathrm{H} = 6 \tag{4.21}$$

and

$$\omega_\mathrm{f} = 2\pi 0.08 \quad (\mathrm{rad/s})\ , \tag{4.22}$$

entailing a fundamental frequency of 0.08 Hz and a trajectory period of $\frac{2\pi}{\omega_\mathrm{f}} = 12.5\,\mathrm{s}$. The system dynamic frequency is given by the maximum frequency of the trajectory which is

$$f_\mathrm{dyn} = n_\mathrm{H} \frac{\omega_\mathrm{f}}{2\pi} = 0.48\,\mathrm{Hz}\ . \tag{4.23}$$

The parameters q_{0k}, a_{ik} and b_{ik} are obtained through an optimization process whose objective is to get a maximal dynamic parameter excitation. The parameter excitation is as much higher as the base regression matrix condition number is lower (Gautier and Khalil 1992). Therefore, the optimization criterion is defined by the condition number of the regression matrix $\boldsymbol{H}_\mathrm{b}$ evaluated on all the S reference points computed by (4.19) and (4.20), as given by

Table 4.4: Parameters of identification trajectory A.

Joint k	$a_{k,1}$	$a_{k,2}$	$a_{k,3}$	$a_{k,4}$	$a_{k,5}$	$a_{k,6}$
1	0.53	−0.01	0.17	0.08	0.60	0.20
2	0.43	−0.19	0.34	0.24	0.11	0.05
3	0.00	0.55	−0.12	0.39	−0.13	−0.07
4	0.32	−0.20	−0.50	0.17	0.06	−0.15
5	0.06	−0.03	−0.41	−0.02	0.65	0.68
6	0.48	0.03	0.32	0.13	0.04	0.20
7	0.79	0.70	0.08	0.02	1.22	−0.28

k	$b_{k,1}$	$b_{k,2}$	$b_{k,3}$	$b_{k,4}$	$b_{k,5}$	$b_{k,6}$	$q_{k,0}$
1	−0.38	−0.17	0.21	0.11	−0.35	0.15	0.50
2	−0.04	0.19	−0.43	−0.19	−0.45	0.36	−0.11
3	0.38	1.21	0.27	0.55	0.06	−0.32	0.54
4	−0.44	−0.00	0.58	−0.10	0.46	0.01	1.76
5	−0.36	0.66	0.43	−0.18	−0.14	−0.14	−1.28
6	−0.07	0.55	0.38	0.19	0.45	0.15	−0.11
7	0.40	−0.13	0.31	0.79	−0.09	0.49	0.42

(2.57). Since joints have limited ranges of positions, velocities and accelerations, the optimization is constrained by a penalty function based on those limits. The objective function is defined over $k(2n_{\mathrm{H}} + 1)$ parameters and is highly non-linear, hence a nonlinear optimization by linear approximation (COBYLA) is employed. Also, due to being highly non-linear, the optimization is very sensible to the initial state, allowing the generation of different trajectories just by starting with different initial parameters.

Four trajectories, named A, B, C and D, are generated by the optimization method described above. Each optimization is started with random initial parameters and stopped when the objective, the condition number, is close to the value 110. For illustrative purposes, the parameters obtained for trajectories A and B are presented in table 4.4 and table 4.5, and their respective reference positions are plotted in figure 4.4 figure 4.5, respectively.

4.3.3 Regression data recording and processing

The robot performs all the excitation trajectories using the joint position control described in section §4.3.1. Each trajectory period is performed continuously three times, for a total of 37.5 second each. Since trajectories have non-zero joint velocities at start and end points, additional trajectory acceleration references are prepended and appended at those points in order to smooth

Table 4.5: Parameters of identification trajectory B.

Joint k	$a_{k,1}$	$a_{k,2}$	$a_{k,3}$	$a_{k,4}$	$a_{k,5}$	$a_{k,6}$
1	0.02	0.55	0.08	−0.30	0.46	0.14
2	−0.27	0.41	−0.15	−0.30	0.08	−0.15
3	0.01	−0.52	0.62	−0.14	−0.46	0.47
4	0.00	−0.16	0.06	0.39	0.55	−0.04
5	−0.79	−1.34	0.50	−0.62	−0.30	−0.55
6	0.07	0.22	0.39	0.12	−0.45	1.05
7	−0.12	−0.80	0.29	0.07	0.99	−0.28

k	$b_{k,1}$	$b_{k,2}$	$b_{k,3}$	$b_{k,4}$	$b_{k,5}$	$b_{k,6}$	$q_{k,0}$
1	−0.41	0.26	0.15	0.50	−0.60	−0.21	0.15
2	0.27	−0.47	−0.28	−0.51	0.11	−0.34	−0.24
3	−0.02	−0.20	−0.15	−0.41	0.73	0.04	0.39
4	0.60	−0.06	0.69	0.25	−0.66	0.12	1.18
5	−0.08	0.01	−0.59	0.42	−0.43	0.43	−1.41
6	−0.39	−0.15	0.51	0.14	−0.20	0.13	0.02
7	−0.64	−0.75	−0.55	0.81	0.36	−0.65	−0.07

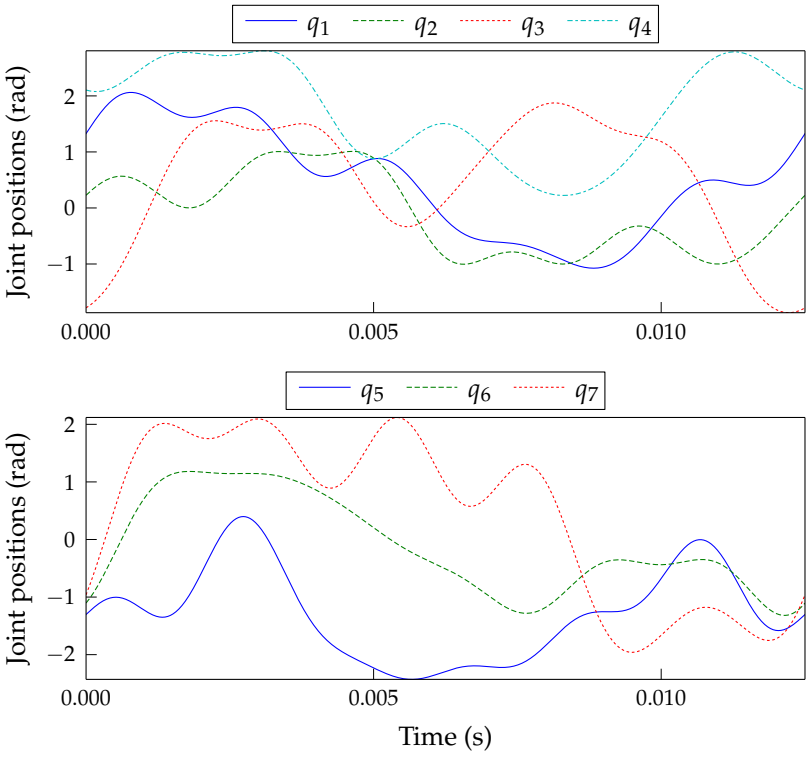

Figure 4.4: Joint reference for the identification trajectory A.

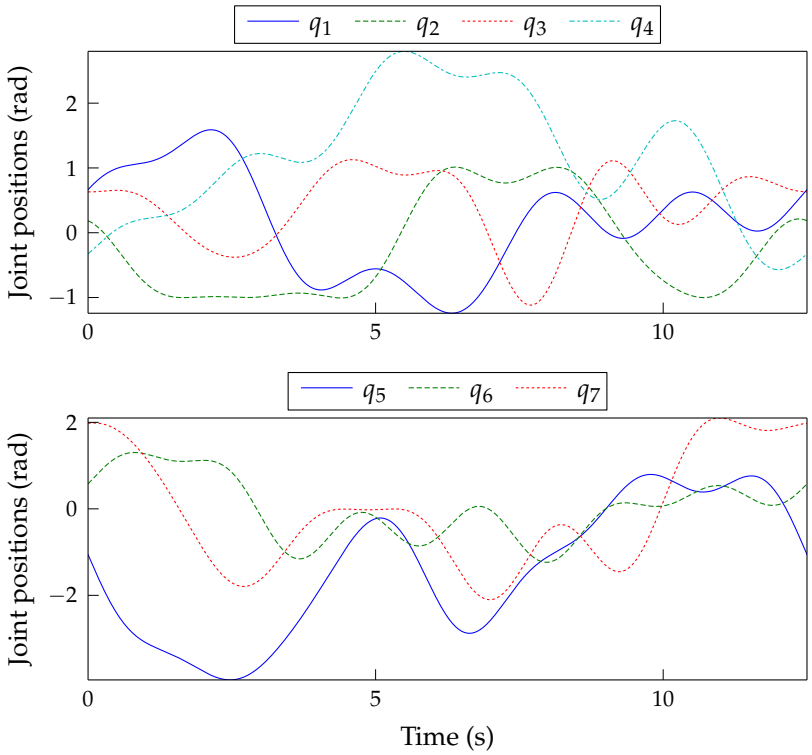

Figure 4.5: Joint reference for the identification trajectory B.

the reference velocities from and to zero. Joint position and torque data is collected while the robot performs the trajectories. Position measurements are obtained from motor encoder signals. Since no torque measurements are possible, torque data is obtained from the computed torque command, $\boldsymbol{\tau}_c$ from (4.16), which is assumed to be equal to the real torque.

Joint position signals are filtered with third order low-pass Butterworth filters with cutoff frequency of $f_{\text{cutoff}} = 10 f_{\text{dyn}} = 4.8\,\text{Hz}$, according to the general rules given by Gautier (1997). Each filter is applied twice, once in the positive time direction and another in the reverse direction, thus compensating phase distortion. First and second order derivatives of the position, respectively $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$, are computed through second order central differences. The data is then trimmed to exclude the starting and stopping acceleration zones, entailing, for each trajectory, $S = 37.5\,\text{s}/0.001\,\text{s} = 37500$ samples of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ and $\boldsymbol{\tau}_c$. The torque data of each trajectory is stacked into four regressand

vectors, $\boldsymbol{\omega}_A$, $\boldsymbol{\omega}_B$, $\boldsymbol{\omega}_C$ and $\boldsymbol{\omega}_D$, with a length of $NS = 262500$ each. Then, the base regressor function, $\boldsymbol{H}_\mathrm{b}$, is applied to the joint motion data, and the resulting matrices are stacked into four regression matrices, $\boldsymbol{W}_A$, $\boldsymbol{W}_B$, $\boldsymbol{W}_C$ and $\boldsymbol{W}_D$, each with a size of $NS \times n_\mathrm{b} = 262500 \times 76$. Therefore, a data set formed by a $(\boldsymbol{W}, \boldsymbol{\omega})$ pair is obtained for each trajectory. The computed condition number of each regression matrix is 105, 111, 111 and 124 for trajectories A, B, C and D, respectively. According to Gautier and Khalil (1992), all these values can be considered good conditions numbers.

4.4 Classical dynamic parameter identification

Trajectory A data is chosen to be used as the regression data set. Hence, $\boldsymbol{W}_A$ and $\boldsymbol{\omega}_A$ will be denoted by $\boldsymbol{W}$ and $\boldsymbol{\omega}$ henceforth. The other data sets are kept for later validation purposes. The classical OLS solution, $\hat{\boldsymbol{\beta}}_\mathrm{OLS}$, is computed by analytic means from the trajectory A data set:

$$\hat{\boldsymbol{\beta}}_\mathrm{OLS} = (\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{W}^T\boldsymbol{\omega} \,. \tag{4.24}$$

Using $\boldsymbol{R}_1$, $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ from the QR decomposition described by (3.45), (3.46) and (3.49), the solution $\hat{\boldsymbol{\beta}}_\mathrm{OLS}$ can be equivalently computed by

$$\hat{\boldsymbol{\beta}}_\mathrm{OLS} = \boldsymbol{R}_1^{-1}\boldsymbol{\rho}_1 \,, \tag{4.25}$$

and the regression error, given by (3.48), is

$$\begin{aligned} \|\boldsymbol{\epsilon}\|^2 &= \|\boldsymbol{\omega} - \boldsymbol{W}\hat{\boldsymbol{\beta}}_\mathrm{OLS}\|^2 \\ &= \|\boldsymbol{\rho}_2\|^2 \,, \end{aligned} \tag{4.26}$$

since $\|\boldsymbol{\rho}_1 - \boldsymbol{R}_1\hat{\boldsymbol{\beta}}_\mathrm{OLS}\|^2 = 0$. The solution $\hat{\boldsymbol{\beta}}_\mathrm{OLS}$ is presented in the third column of table 4.6. In the second column of the same table, the base parameters $\hat{\boldsymbol{\beta}}_\mathrm{CAD}$, mapped from the datasheet CAD-estimated parameters $\hat{\boldsymbol{\delta}}_\mathrm{CAD}$, are presented for comparison.

It is possible to compute an estimation of standard deviations of the identified parameters. A common approach is the formulation given by Khalil and Dombre (2002) which, while making strong assumptions on the determinism on the model and on the distribution of the error, can still be used to provide an idea about the parameter excitation. Through such formulation, the standard deviation of $\hat{\boldsymbol{\beta}}_\mathrm{OLS}$ is given by the vector $\hat{\boldsymbol{\sigma}}_{\hat{\boldsymbol{\beta}}_\mathrm{OLS}}$:

$$\hat{\boldsymbol{\sigma}}_{\hat{\boldsymbol{\beta}}_\mathrm{OLS}} = \sqrt{\mathbf{diag}\left(\hat{\sigma}^2_{\boldsymbol{\epsilon}_\mathrm{OLS}}(\boldsymbol{W}^T\boldsymbol{W})^{-1}\right)} \,, \tag{4.27}$$

where $\hat{\sigma}^2_{\epsilon_{\text{OLS}}}$ is the variance of the regression error $\boldsymbol{\epsilon}_{\text{OLS}}$ given by

$$\hat{\sigma}^2_{\epsilon_{\text{OLS}}} = \frac{\|\boldsymbol{\omega} - \boldsymbol{W}\hat{\boldsymbol{\beta}}_{\text{OLS}}\|^2}{SN - n_{\text{b}}} \,. \tag{4.28}$$

The percentage of relative standard deviation of the k-th parameter of $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ is given by

$$\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\text{OLS}k}} = \frac{\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\text{OLS}k}}}{|\hat{\boldsymbol{\beta}}_{\text{OLS}k}|} \cdot 100\% \,. \tag{4.29}$$

The values are presented in the fourth column of table 4.6 where it can be seen that some parameters have high values, specially the parameters related do the wrist links.

Table 4.6: Classical base parameter estimations for the WAM robot.

	$\hat{\boldsymbol{\beta}}_{\text{CAD}}$	$\hat{\boldsymbol{\beta}}_{\text{OLS}}$	$\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\text{OLS}}}$	$\hat{\boldsymbol{\beta}}_{\text{WLS}}$	$\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\text{WLS}}}$
$L_{yy1} + \ldots$	0.1328	0.3798	0.5	0.3712	0.6
$F_{\text{vj1}} + \ldots$	—	1.6591	0.2	1.6481	0.3
$F_{\text{cj1}} + \ldots$	—	1.8013	0.2	1.8033	0.2
$F_{\text{oj1}} + \ldots$	—	−0.0611	2.6	−0.0610	3.9
$L_{xx2} + \ldots$	1.1810	0.9722	0.3	1.0195	0.4
L_{xy2}	0.0000	−0.0161	9.1	−0.0150	11.3
L_{xz2}	0.0001	0.0056	21.6	0.0135	11.8
$L_{yy2} + \ldots$	1.1890	1.1509	0.3	1.1703	0.3
L_{yz2}	−0.0000	−0.0007	187.6	0.0011	123.2
l_{x2}	−0.0092	0.0399	3.2	0.0315	4.1
$l_{z2} + \ldots$	2.3328	2.4008	0.0	2.3913	0.0
$F_{\text{vj2}} + \ldots$	—	1.5648	0.5	1.5774	0.4
F_{cj2}	—	−0.0496	5.6	−0.0703	3.9
$F_{\text{oj2}} + \ldots$	—	0.7329	1.3	0.6776	1.5
$F_{\text{vm2}} + \ldots$	—	−0.0814	5.4	−0.0523	8.1
F_{cm2}	—	0.7759	0.3	0.7613	0.3
$F_{\text{om2}} + \ldots$	—	−0.1328	2.2	−0.1035	2.5
I_{am2}	—	0.0452	3.8	0.0500	3.5
$L_{xx3} + \ldots$	0.0039	0.0834	2.4	0.0527	3.9
$L_{xy3} + \ldots$	−0.0000	0.0116	6.9	0.0130	6.2
L_{xz3}	−0.0000	−0.0488	2.0	−0.0565	1.9
$L_{yy3} + \ldots$	−0.0070	−0.0179	6.1	−0.0239	4.9
L_{yz3}	0.0000	−0.0442	2.1	−0.0426	2.4
$l_{x3} + \ldots$	0.1476	0.1710	0.3	0.1693	0.2

continues

Table 4.6 continued

	$\hat{\boldsymbol{\beta}}_{\mathrm{CAD}}$	$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}$	$\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}}$	$\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$	$\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}}$
$l_{z3} + \ldots$	−0.0005	0.0327	1.3	0.0271	1.4
$F_{\mathrm{vj3}} + \ldots$	—	0.6395	0.5	0.6417	0.5
F_{cj3}	—	0.3473	0.8	0.3491	0.7
F_{cm3}	—	0.6677	0.4	0.6895	0.4
I_{am3}	—	0.1137	1.8	0.1255	1.7
$L_{xx4} + \ldots$	0.1149	0.0797	1.5	0.0830	1.3
$L_{xy4} + \ldots$	−0.0000	0.0379	1.8	0.0343	2.0
L_{xz4}	−0.0001	−0.0336	1.6	−0.0281	1.8
$L_{yy4} + \ldots$	0.1057	0.1195	1.2	0.0983	1.3
L_{yz4}	0.0001	0.0032	20.9	0.0070	8.9
$l_{x4} + \ldots$	−0.1235	−0.1129	0.3	−0.1133	0.2
$l_{z4} + \ldots$	0.5010	0.4882	0.1	0.4898	0.1
$F_{\mathrm{vj4}} + \ldots$	—	0.5961	0.6	0.5813	0.6
$F_{\mathrm{cj4}} + \ldots$	—	0.5338	0.5	0.5446	0.5
$F_{\mathrm{oj4}} + \ldots$	—	−0.1667	1.7	−0.1257	2.1
I_{am4}	—	−0.0450	2.9	−0.0458	2.9
$L_{xx5} + \ldots$	0.0006	−0.0185	6.0	−0.0019	32.3
L_{xy5}	−0.0000	−0.0115	4.3	−0.0047	4.5
L_{xz5}	0.0000	−0.0137	3.9	−0.0020	14.7
$L_{yy5} + \ldots$	0.0007	−0.0043	16.0	0.0028	8.9
L_{yz5}	0.0000	−0.0117	4.4	−0.0038	5.9
l_{x5}	0.0000	0.0188	1.8	0.0017	7.5
$l_{z5} + \ldots$	−0.0066	0.0031	8.2	−0.0038	2.1
$F_{\mathrm{vj5}} + \ldots$	—	0.0734	5.2	0.0509	1.9
F_{cj5}	—	−0.0041	64.8	−0.0020	31.4
$F_{\mathrm{oj5}} + \ldots$	—	0.0156	16.9	−0.0082	7.9
$F_{\mathrm{vm5}} + \ldots$	—	−0.0069	36.2	0.0148	4.3
F_{cm5}	—	0.0812	2.3	0.0574	0.8
$F_{\mathrm{om5}} + \ldots$	—	0.0039	45.5	0.0081	5.3
I_{am5}	—	0.0022	23.8	0.0041	3.4
$L_{xx6} + \ldots$	0.0006	0.0212	3.5	−0.0029	10.3
L_{xy6}	−0.0000	0.0146	3.0	−0.0021	6.7
L_{xz6}	0.0000	0.0109	3.2	0.0016	9.0
$L_{yy6} + \ldots$	0.0008	0.0104	6.9	−0.0031	7.3
L_{yz6}	0.0002	−0.0001	496.3	−0.0013	12.3
l_{x6}	−0.0001	−0.0093	3.3	0.0018	6.1
$l_{z6} + \ldots$	0.0142	0.0135	1.8	0.0112	0.6
$F_{\mathrm{vj6}} + \ldots$	—	0.0941	4.9	0.0551	2.1

continues

Table 4.6 continued

	$\hat{\boldsymbol{\beta}}_{\mathrm{CAD}}$	$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}$	$\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}}$	$\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$	$\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}}$
F_{cj6}	—	0.0211	13.2	0.0394	1.7
F_{cm6}	—	0.0504	4.7	0.0608	1.0
I_{am6}	—	0.0068	6.4	0.0008	14.7
$L_{xx7} + \ldots$	−0.0000	−0.0110	4.5	−0.0004	45.8
L_{xy7}	0.0000	0.0093	2.7	0.0020	4.1
L_{xz7}	−0.0000	0.0013	9.9	−0.0006	7.2
L_{yz7}	0.0000	−0.0004	33.3	0.0010	4.7
L_{zz7}	0.0001	0.0016	10.1	0.0003	19.4
l_{x7}	−0.0000	−0.0002	75.3	−0.0034	1.6
l_{y7}	0.0000	−0.0036	4.8	−0.0004	9.0
$F_{\mathrm{vj7}} + \ldots$	—	0.0364	4.3	0.0269	0.9
$F_{\mathrm{cj7}} + \ldots$	—	0.0150	13.9	0.0188	1.7
$F_{\mathrm{oj7}} + \ldots$	—	0.0050	36.7	−0.0118	3.0
I_{am7}	—	−0.0008	31.3	0.0007	8.3

In the WAM robot, joint torque amplitudes do not have the same order of magnitude across all joints, since the first four joints have to support much higher forces than the wrist joints. This can be appointed as a possible reason for the OLS solution to have higher relative estimation errors on the wrist joints.

4.4.1 Weighted least squares

The high relative standard deviations issue can be partially mitigated by employing weighted least squares (WLS) techniques to normalize the error. It is common to do a joint-by-joint weighting of the regression data using the inverse of the standard deviation of the OLS error of each joint (Gautier 1997; Gautier and Poignet 2001). The regressor matrix $\boldsymbol{W}$ and the regressand vector $\boldsymbol{\omega}$ are split by joints into N regression matrices $\boldsymbol{W}_{(k)}$ and N regressand vectors $\boldsymbol{\omega}_{(k)}$:

$$\boldsymbol{W}_{(k)} = \boldsymbol{W}_{[i+k,j]} \text{ for } i = 0,\, N,\, 2N,\, \ldots,\, (S-1)N \quad j = 1,\, \ldots,\, n_{\mathrm{b}}\,, \tag{4.30}$$

$$\boldsymbol{\omega}_{(k)} = \boldsymbol{\omega}_{[i+k]} \text{ for } i = 0,\, N,\, 2N,\, \ldots,\, (S-1)N\,, \tag{4.31}$$

which are obtained from the original data by picking the rows corresponding to the respective joint only. The variance of the regression error is then

computed independently for each joint k by

$$\hat{\sigma}^2_{\epsilon_{\text{OLS}}(k)} = \frac{\|\boldsymbol{\omega}_{(k)} - \boldsymbol{W}_{(k)}\hat{\boldsymbol{\beta}}_{\text{OLS}}\|^2}{S - n_{\text{b}}} \,. \tag{4.32}$$

The weights to be applied to each joint data are given by the inverse of the standard deviations,

$$\boldsymbol{w} = \begin{bmatrix} \frac{1}{\hat{\sigma}_{\epsilon_{\text{OLS}}(1)}} & \frac{1}{\hat{\sigma}_{\epsilon_{\text{OLS}}(2)}} & \cdots & \frac{1}{\hat{\sigma}_{\epsilon_{\text{OLS}}(N)}} \end{bmatrix} \,. \tag{4.33}$$

The final weight matrix $\boldsymbol{G}$ is given by repeating the weights $\boldsymbol{w}$ in its diagonal diagonal S times:

$$\boldsymbol{G} = \begin{bmatrix} \mathbf{Diag}(w) & & & \mathbf{0} \\ & \mathbf{Diag}(w) & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{Diag}(w) \end{bmatrix} \,. \tag{4.34}$$

The WLS estimation for the base parameters is then given by

$$\hat{\boldsymbol{\beta}}_{\text{WLS}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{G}\boldsymbol{\omega} - \boldsymbol{G}\boldsymbol{W}\boldsymbol{\beta}\|^2 \,. \tag{4.35}$$

Similarly to (2.55), the WLS analytic solution is computed as

$$\hat{\boldsymbol{\beta}}_{\text{WLS}} = (\boldsymbol{G}\boldsymbol{W}^T\boldsymbol{G}\boldsymbol{W})^{-1}\boldsymbol{G}\boldsymbol{W}^T\boldsymbol{G}\boldsymbol{\omega} \,. \tag{4.36}$$

The values of $\hat{\boldsymbol{\beta}}_{\text{WLS}}$ and their relative standard deviations, $\%\hat{\sigma}_{\hat{\boldsymbol{\beta}}_{\text{WLS}}}$, are given in the last two columns of table 4.6. There are still some high relative standard deviations, some of which may be explained by the fact that the respective real base parameters (thus estimated ones too) are very close to zero.

For regression quality assessment, it is common to measure the prediction relative error $\epsilon_r(\boldsymbol{W}, \boldsymbol{\omega}, \hat{\boldsymbol{\beta}})$ which, for a given data set $\boldsymbol{W}, \boldsymbol{\omega}$ and a given solution $\hat{\boldsymbol{\beta}}$, is defined as the ratio between the norm of the prediction error and the norm of the torque data itself:

$$\epsilon_r(\boldsymbol{W}, \boldsymbol{\omega}, \hat{\boldsymbol{\beta}}) = \frac{\|\boldsymbol{\epsilon}\|}{\|\boldsymbol{\omega}\|} = \frac{\|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}\|}{\|\boldsymbol{\omega}\|} = \frac{\|\boldsymbol{\omega} - \boldsymbol{W}\hat{\boldsymbol{\beta}}\|}{\|\boldsymbol{\omega}\|} \,. \tag{4.37}$$

The prediction relative error in percentage for both OLS and WLS techniques is presented in table 4.7 in a per-joint basis. While the relative errors are high on wrist joints, there is a significant decrease on their values when the WLS solution is used.

Table 4.7: Comparison of relative error percentage of predicted torque in each joint for OLS and WLS techniques. (Note: weight factors have no influence on WLS relative error when it is computed on a per-joint basis.)

Joint k	$\frac{\|\boldsymbol{\omega}_{(k)} - \boldsymbol{W}_{(k)}\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}\|}{\|\boldsymbol{\omega}_{(k)}\|}\%$	$\frac{\|\boldsymbol{\omega}_{(k)} - \boldsymbol{W}_{(k)}\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}\|}{\|\boldsymbol{\omega}_{(k)}\|}\%$
1	15.79	16.24
2	2.06	2.11
3	11.85	11.70
4	11.50	12.16
5	51.17	22.43
6	45.47	26.29
7	74.11	57.92

4.5 Dynamic parameter identification with LMI–SDP methods

The LMI–SDP methods devised in chapter 3 are implemented in *Python* and applied to the WAM regression data set. The LMIs are defined with symbolic matrices using the *SymPy* package, thus allowing easy mapping from the standard parameters space to the base–dependent parameters space. Referring to the *PyLMI-SDP* package developed within this work (Sousa 2014a), the coefficient matrices are extracted from the symbolic LMI and saved in SDPA file format. The SDPA format describe SDP problems and is accepted by a large number of SDP solvers which directly exploit the sparsity of LMI matrices. The SDP optimization is performed by the *DSDP5* solver (Benson and Ye 2008), which is based on the *interior-point* method, a common SDP solving technique. The *DSDP5* solver has a tolerance parameter to define the solution accuracy which is set to 10^{-6}. Similarly, the LMI strictness is enforced with a safe margin of $\varepsilon = 10^{-6}$ (see (3.29)), hence accounting for the highest possible solver error.

4.5.1 Feasibility test

Both the solutions $\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}$ and $\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$ obtained above are checked for physical feasibility referring to method devised in section §3.3. The "find" problem of (3.35) is implemented by setting an objective function equal to zero. In both cases, the solver issues a certificate of infeasibility, which proves that

$\hat{\boldsymbol{\beta}}_{\text{OLS}}$ and $\hat{\boldsymbol{\beta}}_{\text{WLS}}$ are not physically feasible. Furthermore, when evaluating the inertia matrix $\boldsymbol{M}(\boldsymbol{q})$ at several random $\boldsymbol{q}$ values using these solutions, there are some $\boldsymbol{q}$ points where the matrix is not positive definite, thus corroborating the physical infeasibility result given by the LMI–SDP method.

Henceforth, in the other LMI–SDP methods, the weighted data is used as regression data set and $\hat{\boldsymbol{\beta}}_{\text{WLS}}$ as the classical identification solution.

4.5.2 Infeasible parameter correction

The feasible base parameter vector closest to $\hat{\boldsymbol{\beta}}_{\text{WLS}}$ is found through the method of section §3.4. Such solution, denoted by $\boldsymbol{\beta}'_{\text{WLS}}$, is shown in table 4.6. The method finds a $(\imath', \boldsymbol{\beta}'_{\text{WLS}}, \delta'_{\text{d}})$ solution, whose $\imath'$ value is $0.0077 > 0$, once again showing $\hat{\boldsymbol{\beta}}_{\text{WLS}}$ to be not physically feasible. The numerical matrix $\bar{\boldsymbol{D}}_{\boldsymbol{\beta}}$ evaluated on $(\boldsymbol{\beta}'_{\text{WLS}}, \delta'_{\text{d}})$ is then computed and its positive definiteness is double checked by verifying that all its eigenvalues are positive.

4.5.3 Feasible LS parameter identification

The feasible identification method proposed in section §3.5 is applied to the weighted regression data used above. The procedure is the same with the exception that $\boldsymbol{GW}$ and $\boldsymbol{G\omega}$ are used in place of $\boldsymbol{W}$ and $\boldsymbol{\omega}$. For this data, the SDP solver finds the feasible solution $\boldsymbol{\beta}^{\star}_{\text{WLS}}$, presented in table 4.9. The matrix $\bar{\boldsymbol{D}}_{\boldsymbol{\beta}}(\boldsymbol{\beta}^{\star}_{\text{WLS}}, \delta^{\star}_{\text{d}})$ shows to be positive definite, thus corroborating that $\boldsymbol{\beta}^{\star}_{\text{WLS}}$ is physically feasible.

4.5.4 Considering additional constrains on dynamic parameters

The proposed methods can be used to find solutions that not only verify the basic feasibility conditions, but which also fit additional knowledge about a given robot. For the WAM robot, its datasheet supplies information about the link shapes and total mass that can be exploited to further constrain the space of the expected solution. For each WAM link there is information about its bounding box, i.e., the smallest cuboid aligned with the link frame which contains the whole link body. Hence, each center of mass is constrained by:

$$\begin{cases} \boldsymbol{l}_k - m_k \boldsymbol{r}_{k,l} \geq \boldsymbol{0} \\ -\boldsymbol{l}_k + mk \boldsymbol{r}_{k,u} \geq \boldsymbol{0} \end{cases} \text{for } k = 1, \ldots, N\,, \tag{4.38}$$

Table 4.8: Additional constraints: spatial limits of center of masses relative to link frames.

Link k	$\boldsymbol{r}_{k,l}$	$\boldsymbol{r}_{k,u}$
1	$(-0.14, -0.174, -0.084)$	$(0.14, 0.174, 0.346)$
2	$(-0.084, -0.174, -0.084)$	$(0.084, 0.174, 0.17)$
3	$(-0.09, -0.55, -0.045)$	$(0.04, 0.04, 0.045)$
4	$(-0.045, -0.045, -0.05)$	$(0.095, 0.045, 0.83)$
5	$(-0.045, -0.02, -0.045)$	$(0.045, 0.1, 0.045)$
6	$(-0.045, -0.06, -0.02)$	$(0.045, 0.045, 0.06)$
7	$(-0.045, -0.045, -0.018)$	$(0.045, 0.045, 0.001)$

where $\boldsymbol{r}_{k,l}$ and $\boldsymbol{r}_{k,u}$ are thee-dimensional lower and upper bounds, respectively. The bounding boxes limits, relatively to the respective link frames, are shown in table 4.8. Since the total robot mass is known to be around 27 Kg, the sum of link masses is also constrained this maximum,

$$27 - \sum_{k=1}^{N} m_k > 0 \,. \tag{4.39}$$

Introducing these constrains in the problem is as simple as appending them to the main LMI matrix $\boldsymbol{D}(\boldsymbol{\delta})$ formulation (see (3.25)), hence obtaining an extended LMI matrix $\bar{\boldsymbol{D}}_{\beta}(\boldsymbol{\beta}, \delta_{\mathrm{d}})$ to be used in the SDP methods. When taking into account these extra constraints in the feasibility test, the solution $\boldsymbol{\beta}^{\star}_{\mathrm{WLS}}$ does not pass, i.e., although that solution is physically feasibility, it does not meet the additional criterion. The feasible identification method including the extra constraints is then applied, leading to a new solution $\boldsymbol{\beta}^{\star \mathrm{e}}_{\mathrm{WLS}}$ shown in Table table 4.9.

Table 4.9: Classical and LMI–SDP estimated base parameters for the WAM robot.

	$\hat{\boldsymbol{\beta}}_{\mathrm{CAD}}$	$\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$	$\boldsymbol{\beta}'_{\mathrm{WLS}}$	$\boldsymbol{\beta}^{\star}_{\mathrm{WLS}}$	$\boldsymbol{\beta}^{\star \mathrm{e}}_{\mathrm{WLS}}$
$L_{yy1} + \ldots$	0.1328	0.3712	0.3712	0.3380	0.3404
$F_{\mathrm{vj1}} + \ldots$	—	1.6481	1.6481	1.6312	1.6209
$F_{\mathrm{cj1}} + \ldots$	—	1.8033	1.8033	1.8076	1.8116
$F_{\mathrm{oj1}} + \ldots$	—	-0.0610	-0.0610	-0.0603	-0.0598
$L_{xx2} + \ldots$	1.1810	1.0195	1.0195	1.0705	1.0679
L_{xy2}	0.0000	-0.0150	-0.0149	-0.0034	0.0018
L_{xz2}	0.0001	0.0135	0.0135	0.0028	-0.0003
$L_{yy2} + \ldots$	1.1890	1.1703	1.1712	1.2719	1.2805

continues

Table 4.9 continued

	$\hat{\boldsymbol{\beta}}_{\mathrm{CAD}}$	$\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$	$\boldsymbol{\beta}'_{\mathrm{WLS}}$	$\boldsymbol{\beta}^{\star}_{\mathrm{WLS}}$	$\boldsymbol{\beta}^{\star\mathrm{e}}_{\mathrm{WLS}}$
L_{yz2}	−0.0000	0.0011	0.0011	0.0067	0.0074
l_{x2}	−0.0092	0.0315	0.0315	0.0258	0.0258
$l_{z2} + \ldots$	2.3328	2.3913	2.3913	2.3910	2.3914
$F_{\mathrm{vj2}} + \ldots$	—	1.5774	1.5774	1.4507	1.4463
F_{cj2}	—	−0.0703	0.0000	0.0000	0.0000
$F_{\mathrm{oj2}} + \ldots$	—	0.6776	0.6776	0.6379	0.6487
$F_{\mathrm{vm2}} + \ldots$	—	−0.0523	−0.0523	−0.0140	−0.0130
F_{cm2}	—	0.7613	0.7613	0.7552	0.7536
$F_{\mathrm{om2}} + \ldots$	—	−0.1035	−0.1035	−0.1042	−0.1036
I_{am2}	—	0.0500	0.0500	0.0065	0.0000
$L_{xx3} + \ldots$	0.0039	0.0527	0.0531	0.0467	0.0446
$L_{xy3} + \ldots$	−0.0000	0.0130	0.0096	0.0070	0.0063
L_{xz3}	−0.0000	−0.0565	−0.0559	−0.0499	−0.0518
$L_{yy3} + \ldots$	−0.0070	−0.0239	−0.0098	0.0061	0.0093
L_{yz3}	0.0000	−0.0426	−0.0433	−0.0279	−0.0122
$l_{x3} + \ldots$	0.1476	0.1693	0.1687	0.1698	0.1711
$l_{z3} + \ldots$	−0.0005	0.0271	0.0267	0.0244	0.0220
$F_{\mathrm{vj3}} + \ldots$	—	0.6417	0.6417	0.6245	0.6256
F_{cj3}	—	0.3491	0.3491	0.3601	0.3655
F_{cm3}	—	0.6895	0.6895	0.6997	0.6971
I_{am3}	—	0.1255	0.1255	0.0638	0.0619
$L_{xx4} + \ldots$	0.1149	0.0830	0.0899	0.0874	0.0855
$L_{xy4} + \ldots$	−0.0000	0.0343	0.0271	0.0204	0.0162
L_{xz4}	−0.0001	−0.0281	−0.0136	−0.0241	−0.0237
$L_{yy4} + \ldots$	0.1057	0.0983	0.1002	0.0870	0.0862
L_{yz4}	0.0001	0.0070	−0.0006	−0.0003	−0.0030
$l_{x4} + \ldots$	−0.1235	−0.1133	−0.1116	−0.1130	−0.1127
$l_{z4} + \ldots$	0.5010	0.4898	0.4878	0.4918	0.4919
$F_{\mathrm{vj4}} + \ldots$	—	0.5813	0.5813	0.5764	0.5774
$F_{\mathrm{cj4}} + \ldots$	—	0.5446	0.5446	0.5548	0.5568
$F_{\mathrm{oj4}} + \ldots$	—	−0.1257	−0.1257	−0.1192	−0.1174
I_{am4}	—	−0.0458	0.0000	0.0000	0.0000
$L_{xx5} + \ldots$	0.0006	−0.0019	0.0011	−0.0007	−0.0004
L_{xy5}	−0.0000	−0.0047	−0.0017	−0.0016	−0.0013
L_{xz5}	0.0000	−0.0020	0.0003	−0.0022	−0.0023
$L_{yy5} + \ldots$	0.0007	0.0028	0.0038	0.0049	0.0049
L_{yz5}	0.0000	−0.0038	−0.0002	−0.0013	−0.0011
l_{x5}	0.0000	0.0017	0.0022	0.0026	0.0024

continues

Table 4.9 continued

	$\hat{\beta}_{\mathrm{CAD}}$	$\hat{\beta}_{\mathrm{WLS}}$	β'_{WLS}	$\beta^{\star}_{\mathrm{WLS}}$	$\beta^{\star\mathrm{e}}_{\mathrm{WLS}}$
$l_{z5} + \ldots$	−0.0066	−0.0038	−0.0031	−0.0039	−0.0039
$F_{\mathrm{vj5}} + \ldots$	—	0.0509	0.0509	0.0446	0.0444
F_{cj5}	—	−0.0020	0.0000	0.0000	0.0000
$F_{\mathrm{oj5}} + \ldots$	—	−0.0082	−0.0082	−0.0079	−0.0085
$F_{\mathrm{vm5}} + \ldots$	—	0.0148	0.0148	0.0153	0.0153
F_{cm5}	—	0.0574	0.0574	0.0567	0.0566
$F_{\mathrm{om5}} + \ldots$	—	0.0081	0.0081	0.0065	0.0067
I_{am5}	—	0.0041	0.0041	0.0025	0.0026
$L_{xx6} + \ldots$	0.0006	−0.0029	0.0009	−0.0004	−0.0006
L_{xy6}	−0.0000	−0.0021	−0.0000	−0.0003	−0.0003
L_{xz6}	0.0000	0.0016	0.0001	0.0012	0.0013
$L_{yy6} + \ldots$	0.0008	−0.0031	0.0017	0.0025	0.0024
L_{yz6}	0.0002	−0.0013	−0.0000	−0.0005	−0.0006
l_{x6}	−0.0001	0.0018	0.0008	0.0003	0.0004
$l_{z6} + \ldots$	0.0142	0.0112	0.0081	0.0112	0.0112
$F_{\mathrm{vj6}} + \ldots$	—	0.0551	0.0551	0.0583	0.0576
F_{cj6}	—	0.0394	0.0394	0.0370	0.0374
F_{cm6}	—	0.0608	0.0608	0.0606	0.0605
I_{am6}	—	0.0008	0.0008	0.0001	0.0000
$L_{xx7} + \ldots$	−0.0000	−0.0004	0.0001	0.0001	0.0002
L_{xy7}	0.0000	0.0020	−0.0000	0.0013	0.0013
L_{xz7}	−0.0000	−0.0006	−0.0000	−0.0004	−0.0004
L_{yz7}	0.0000	0.0010	0.0000	0.0001	0.0001
L_{zz7}	0.0001	0.0003	0.0005	0.0006	0.0006
l_{x7}	−0.0000	−0.0034	−0.0022	−0.0032	−0.0032
l_{y7}	0.0000	−0.0004	−0.0004	−0.0006	−0.0006
$F_{\mathrm{vj7}} + \ldots$	—	0.0269	0.0269	0.0274	0.0274
$F_{\mathrm{cj7}} + \ldots$	—	0.0188	0.0188	0.0183	0.0183
$F_{\mathrm{oj7}} + \ldots$	—	−0.0118	−0.0118	−0.0115	−0.0115
I_{am7}	—	0.0007	0.0007	0.0004	0.0004

4.6 Analysis and validation of dynamic parameter estimations

In order to assess the validity of the base parameter solutions presented above, trajectories B, C and D are used as validation trajectories. Hence, comparisons between torque prediction and measured torque are made for these

Table 4.10: Percentage of relative error ($100\% \cdot \|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}\|/\|\boldsymbol{\omega}\|$) of predicted torque for identification and validation trajectories.

	$\hat{\boldsymbol{\beta}}_{\text{WLS}}$	$\boldsymbol{\beta}'_{\text{WLS}}$	$\boldsymbol{\beta}^{\star}_{\text{WLS}}$	$\boldsymbol{\beta}^{\star\text{e}}_{\text{WLS}}$
(weighted data) regression traj. A	4.57	4.73	4.62	4.63
regression traj. A	4.53	4.61	4.57	4.58
validation traj. B	7.90	7.81	7.80	7.79
validation traj. C	6.33	6.25	6.20	6.19
validation traj. D	7.05	6.99	6.85	6.87

trajectories. The analysis is also made on the trajectory used for regression, trajectory A. Predicted torque, $\hat{\boldsymbol{\omega}}_{T,\boldsymbol{\beta}}$, is computed for all data sets T and solutions $\boldsymbol{\beta}$:

$$\begin{aligned} \hat{\boldsymbol{\omega}}_{T,\boldsymbol{\beta}} = \boldsymbol{W}_T\boldsymbol{\beta} \quad & \text{for } T = A, B, C, D \\ & \text{for } \boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{\text{WLS}}, \boldsymbol{\beta}'_{\text{WLS}}, \boldsymbol{\beta}^{\star}_{\text{WLS}} \, . \end{aligned} \tag{4.40}$$

The respective relative error of each prediction, as given by (4.37), is also computed:

$$\epsilon_r(\boldsymbol{W}_T, \boldsymbol{\omega}_T, \boldsymbol{\beta}) = \frac{\|\boldsymbol{\omega}_T - \hat{\boldsymbol{\omega}}_{T,\boldsymbol{\beta}}\|}{\|\boldsymbol{\omega}_T\|} \, . \tag{4.41}$$

The values of the relative error in percentage are shown in table 4.10.

We starting by looking only to the regression trajectory results, i.e., the two first rows of table 4.10. It can be seen that the relative error of the weighted data ($\boldsymbol{G}\boldsymbol{W}_A$ and $\boldsymbol{G}\boldsymbol{\omega}_A$) is below 5% for all solutions indicating good parameter estimation. As expected, physically feasible constrained estimations show higher regression errors than the classical unconstrained one $\hat{\boldsymbol{\beta}}_{\text{WLS}}$. Also, as expected, $\boldsymbol{\beta}^{\star}_{\text{WLS}}$ has less error than $\boldsymbol{\beta}'_{\text{WLS}}$ and less error than $\boldsymbol{\beta}^{\star\text{e}}_{\text{WLS}}$. The errors computed on the non-weighted regression data follow the same pattern. The differences between the errors across the several estimations are very low, as well as the differences between the estimations themselves as seen in table 4.9. This indicates that the classical solution $\hat{\boldsymbol{\beta}}_{\text{WLS}}$, while not being physically feasible, is very close to feasibility region. Figure 4.6 shows plots of measured torque, predicted torque by $\boldsymbol{\beta}^{\star\text{e}}_{\text{WLS}}$ and respective error for regression trajectory. Figure 4.7 compares the prediction error of that solution with the one from the classical solution $\hat{\boldsymbol{\beta}}_{\text{WLS}}$. It can be seen in the plots that the differences between solution predictions is small.

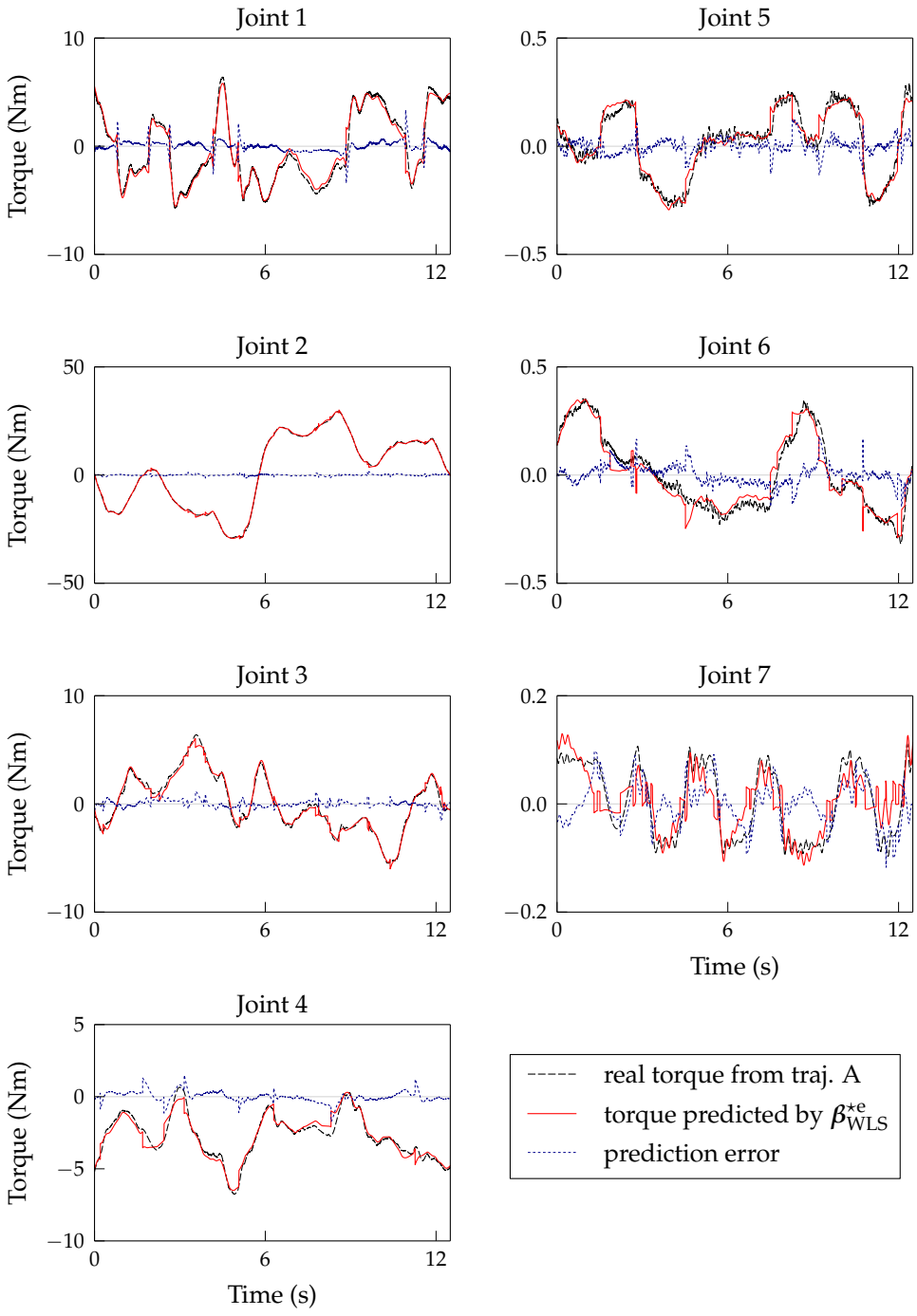

Figure 4.6: Measured torque and torque predicted by $\beta^{\star e}_{WLS}$ for the regression trajectory (traj. A).

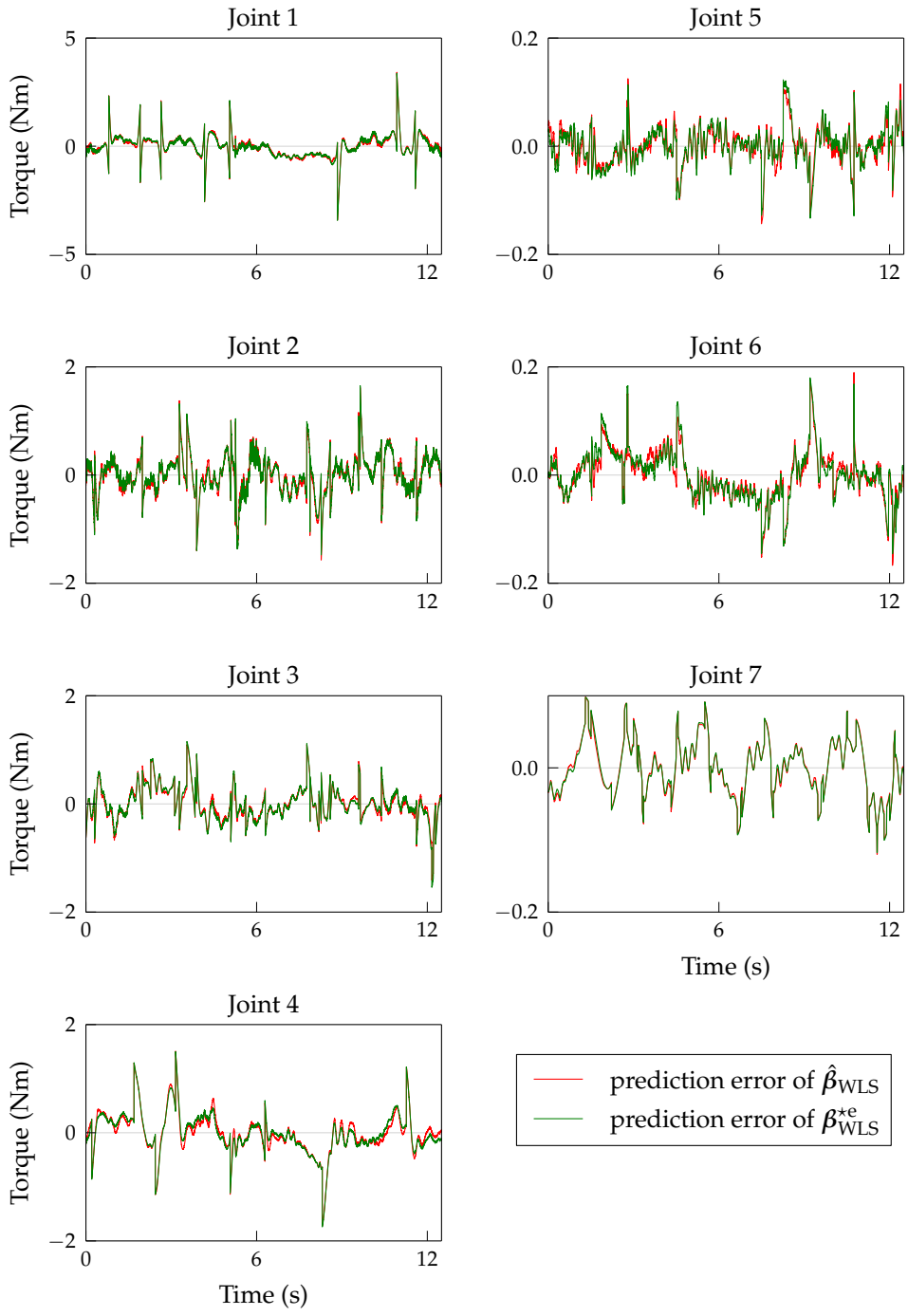

Figure 4.7: Classical solution versus feasible solution prediction error for the regression trajectory.

Table 4.11: Percentage of relative error ($100\% \cdot \|\boldsymbol{\omega}_{(k)} - \hat{\boldsymbol{\omega}}_{(k)}\| / \|\boldsymbol{\omega}_{(k)}\|$) of predicted torque per joint. Arrows indicate an increase (↗) or a decrease (↘) of the error from the classical solution to the feasible solution.

	regression traj.			validation traj.							
	A			B			C			D	
k	$\hat{\boldsymbol{\beta}}_{\text{WLS}}$		$\boldsymbol{\beta}^{\star\text{e}}{}_{\text{WLS}}$	$\hat{\boldsymbol{\beta}}_{\text{WLS}}$		$\boldsymbol{\beta}^{\star\text{e}}{}_{\text{WLS}}$	$\hat{\boldsymbol{\beta}}_{\text{WLS}}$		$\boldsymbol{\beta}^{\star\text{e}}{}_{\text{WLS}}$	$\hat{\boldsymbol{\beta}}_{\text{WLS}}$		$\boldsymbol{\beta}^{\star\text{e}}{}_{\text{WLS}}$
1	16.2	↗	16.3	30.3	→	30.3	24.1	↘	23.9	24.8	↘	24.5
2	2.1	→	2.1	3.4	↘	3.2	3.4	↘	3.3	3.9	↘	3.7
3	11.7	↗	12.2	16.3	↘	15.6	10.9	↘	10.4	16.5	↘	15.8
4	12.2	→	12.2	11.2	↘	10.5	9.5	↘	9.3	10.0	↘	9.3
5	22.4	↗	22.8	33.5	↘	29.0	39.1	↘	34.4	46.3	↘	41.7
6	26.3	↗	27.3	34.4	↘	31.8	31.1	↘	28.6	35.1	↘	31.6
7	57.9	↗	58.2	67.4	↘	65.9	73.3	↘	72.6	78.6	↘	77.8

4.6.1 Validation trajectories

The relative prediction errors of validation trajectories (bottom three rows of table 4.10) are below 8%, indicating also good torque prediction for this trajectories. Moreover, there is a tendency in feasible solutions to show less error than the classical solution, showing better performance outside the identification set.

Additionally to the total relative error, the relative error of predictions split by joints is also computed. Table 4.11 shows these errors for the classical estimation $\hat{\boldsymbol{\beta}}_{\text{WLS}}$ and the feasible estimation $\boldsymbol{\beta}^{\star\text{e}}{}_{\text{WLS}}$. The proposed feasible solution shows lower errors on all joints outside the regression set (green arrows), which seems to indicate that this solution gives a generally better model of the robot. Plots of measured torque, torque predicted by $\boldsymbol{\beta}^{\star\text{e}}{}_{\text{WLS}}$, and respective error for the validation trajectory B are shown in figure 4.8. In figure 4.9, prediction errors for this trajectory, from both the classical and the feasible solutions, are compared. Again, the differences between solution predictions is small.

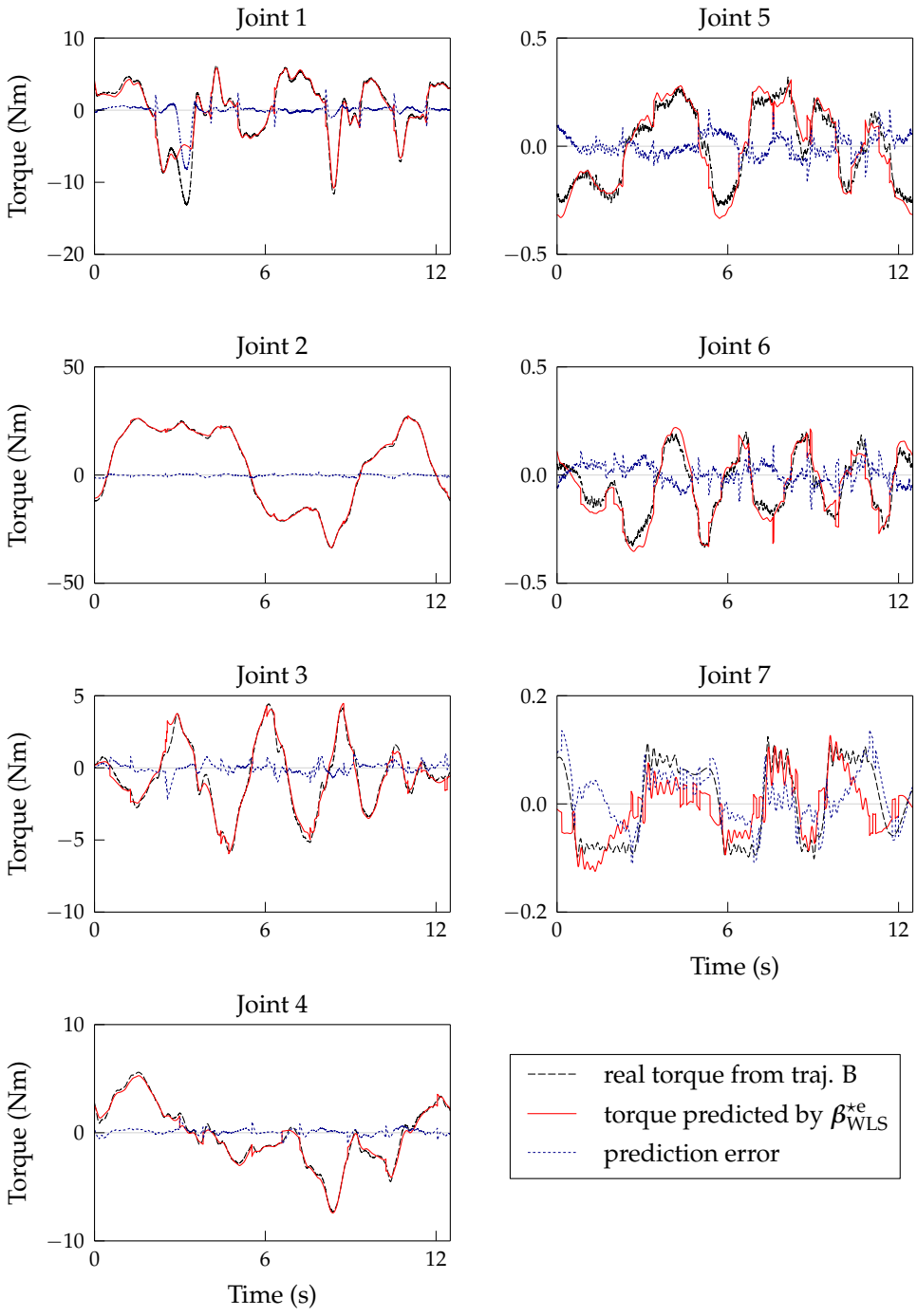

Figure 4.8: Measured torque and torque predicted by $\beta^{\star e}_{WLS}$ for the validation trajectory B.

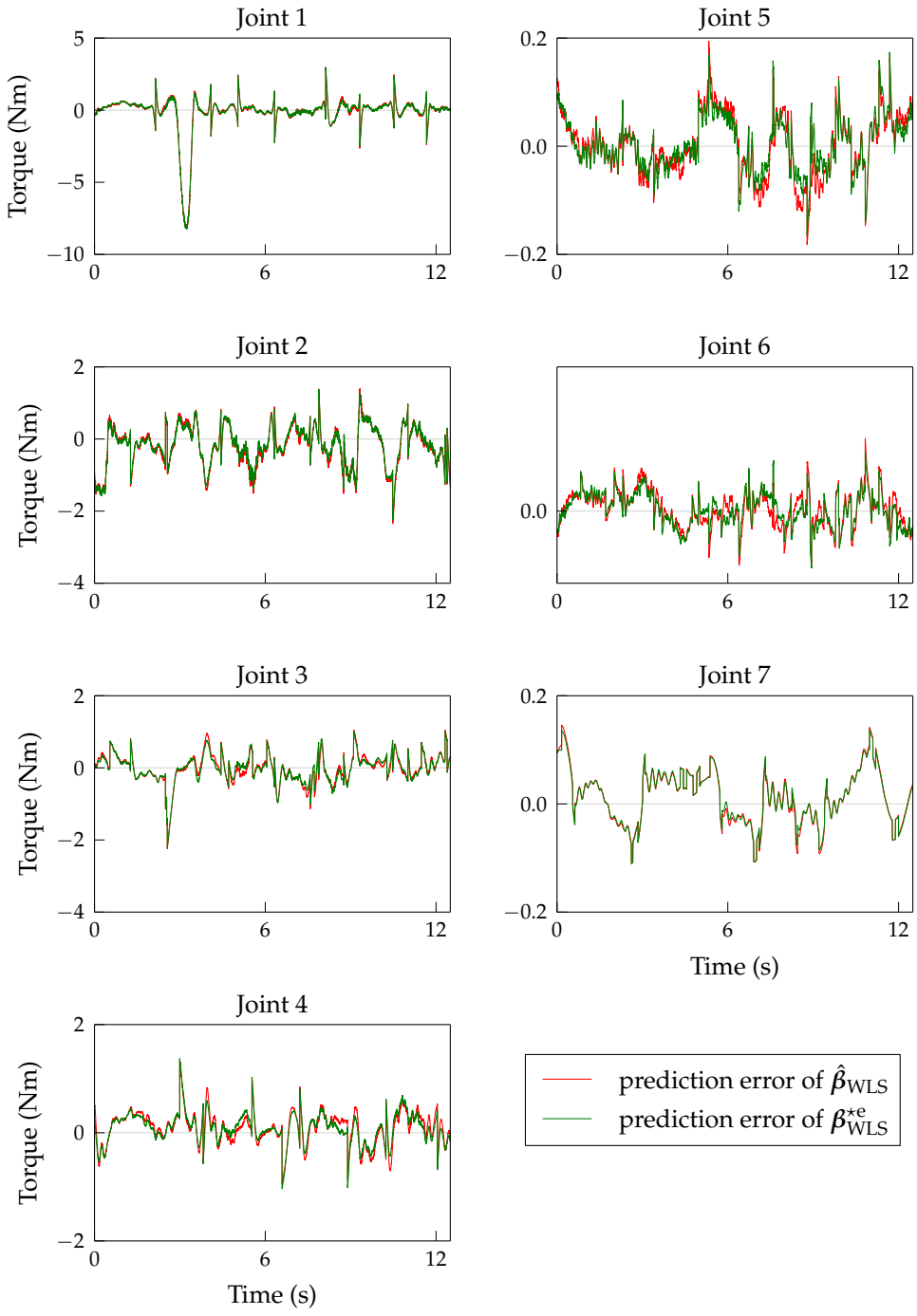

Figure 4.9: Classical solution versus feasible solution prediction error for the validation trajectory B.

5 EXPERIMENTAL APPLICATIONS

5.1 Robot dynamic simulation

Robot behavior simulation is one of the major applications for the dynamic model. Dynamics simulation is very useful even for testing non model-based controllers. While it is expected that physically infeasible dynamic parameters entail unstable simulations (Yoshida and Khalil 2000), there is no example in the literature showing to what extent that happens. This section presents results for the simulation of the WAM robot using the dynamic model and the identified parameters of chapter 4.

5.1.1 Direct dynamic model

Equation (2.9) is in the form of the so-called inverse dynamic model, as it encodes the forces at the joints as function of the joint motion. To simulate the robot, the direct dynamic equation, which presents motion as function of force, is needed:

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}(\boldsymbol{q})^{-1} \left(\boldsymbol{\tau}_{\mathrm{c}}(t) + \boldsymbol{\tau}_{\mathrm{e}}(t) - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \right) . \tag{5.1}$$

This equation is a second order ordinary differential equation (ODE) in time. With an extension of the states, it can be written as a first order equation as

$$\dot{\boldsymbol{y}} = \boldsymbol{\phi}(t, \boldsymbol{y}) , \tag{5.2}$$

where

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} , \tag{5.3}$$

and

$$\boldsymbol{\phi}(t, \boldsymbol{y}) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}(\boldsymbol{q})^{-1} \left(\boldsymbol{\tau}_{\mathrm{c}}(t) + \boldsymbol{\tau}_{\mathrm{e}}(t) - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \right) \end{bmatrix} . \tag{5.4}$$

Having initial values for $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, and having $\boldsymbol{\tau}_{\mathrm{c}}$ and $\boldsymbol{\tau}_{\mathrm{e}}$ as functions of time and motion, a numerical integration method can be used to compute the evolution of the system over time, which is in fact the simulation of the robot dynamic.

5.1.2 Comparison of simulation using infeasible and feasible parameters

In order to test the dynamic parameters, the same model used to perform the identification is used for the simulation, including joint frictions and rotor inertias as described in section §4.2. The only difference is that the sign function of (4.10) and (4.11) is approximated by a tight sigmoid function, for the sake of numerical stability in the numerical integration method. Since the additional dynamic term considered in (4.9) includes the rotor inertia, which is function of $\ddot{\boldsymbol{q}}$, such term is split in the direct dynamic equation:

$$\ddot{\boldsymbol{q}} = \left(\boldsymbol{M}(\boldsymbol{q}) + \boldsymbol{I}_{\mathbf{a}\mathrm{j}}\right)^{-1} \left(\boldsymbol{\tau}_{\mathrm{c}}(t) + \boldsymbol{\tau}_{\mathrm{e}}(t) - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{h}_f(\boldsymbol{q}, \dot{\boldsymbol{q}})\right) , \tag{5.5}$$

for

$$\boldsymbol{I}_{\mathbf{a}\mathrm{j}} = \boldsymbol{T}\boldsymbol{I}_{\mathbf{a}\mathrm{m}}\boldsymbol{T}^T , \tag{5.6}$$

and

$$\boldsymbol{h}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{f}_{\mathrm{j}}(\dot{\boldsymbol{q}}) + \boldsymbol{T}\, \boldsymbol{f}_{\mathrm{m}}(\boldsymbol{T}^T \dot{\boldsymbol{q}}) . \tag{5.7}$$

Having implemented the direct dynamic equation, four simulation tests are performed, one for the WLS classical solution $\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$, and the others for the feasible solutions $\boldsymbol{\beta}'_{\mathrm{WLS}}$, $\boldsymbol{\beta}^{\star}_{\mathrm{WLS}}$ and $\boldsymbol{\beta}^{\star\mathrm{e}}_{\mathrm{WLS}}$ (see table 4.9). Each simulation starts with the joints at the zero position, $\boldsymbol{q}_{\mathrm{init}} = \mathbf{0}$, as depicted in figure 4.2, and null velocities, $\dot{\boldsymbol{q}}_{\mathrm{init}} = \mathbf{0}$. The motors are simulated to be always off, $\boldsymbol{\tau}_{\mathrm{c}}(t) = \mathbf{0}$, thus entailing a passive system. The contact forces are also set to zero, $\boldsymbol{\tau}_{\mathrm{c}}(t) = \mathbf{0}$, entailing not only no environment contact but also no simulation of joint limits or robot inter-link collisions — the robot is free to "enter into itself". Hence, the test consist on the simulation of the the free fall of the robot, where the joint torques are only due to gravity force, frictions, and Coriolis and centripetal effects. While this may seem to have little utility for real simulation applications, it is enough for the sake of dynamic parameter validation.

The simulation is performed referring to the *Dormand–Prince 4(5)* numerical integration method from the well-known family of Runge–Kutta ODE solvers. For the WLS classical solution $\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$, the integration method is unable to proceed past 0.041 s of simulated time due to the lack of numerical accuracy and stability. The trajectory performed by the joints within the simulation time is shown in figure 5.1, where an exponential divergent behavior is clearly seen. The total energy of the system ($E_k + E_p$, see section §2.2) over

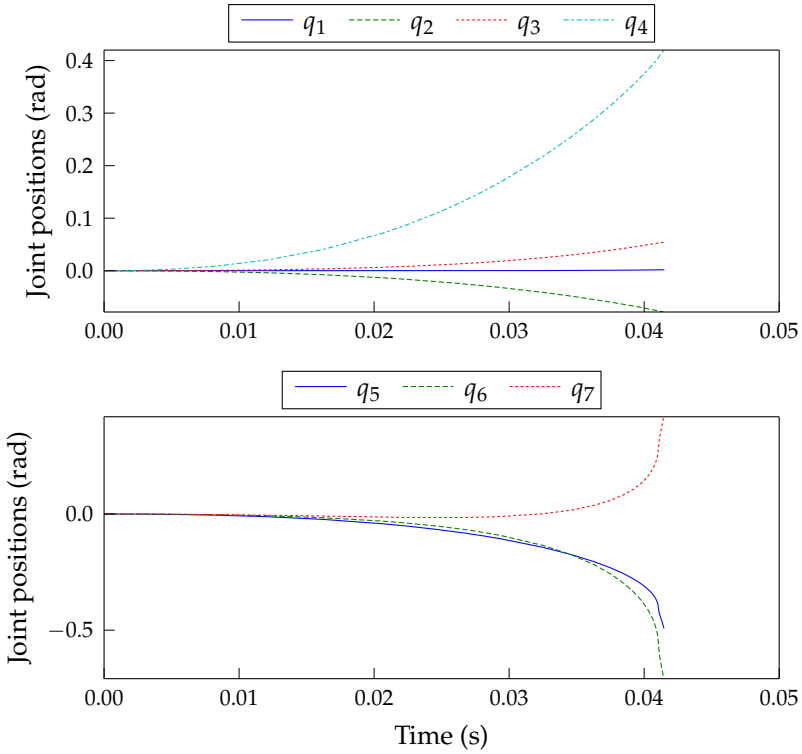

Figure 5.1: Joint trajectory of free fall robot simulation using classical regression parameters $\hat{\boldsymbol{\beta}}_{\text{WLS}}$.

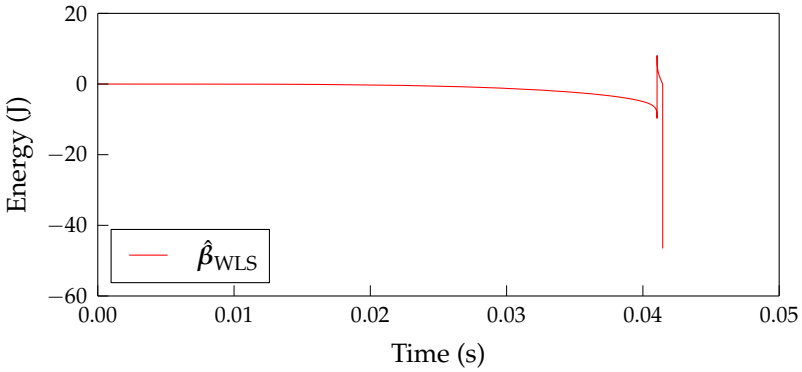

Figure 5.2: Total energy of free fall robot simulation using classical regression parameters $\hat{\boldsymbol{\beta}}_{\text{WLS}}$.

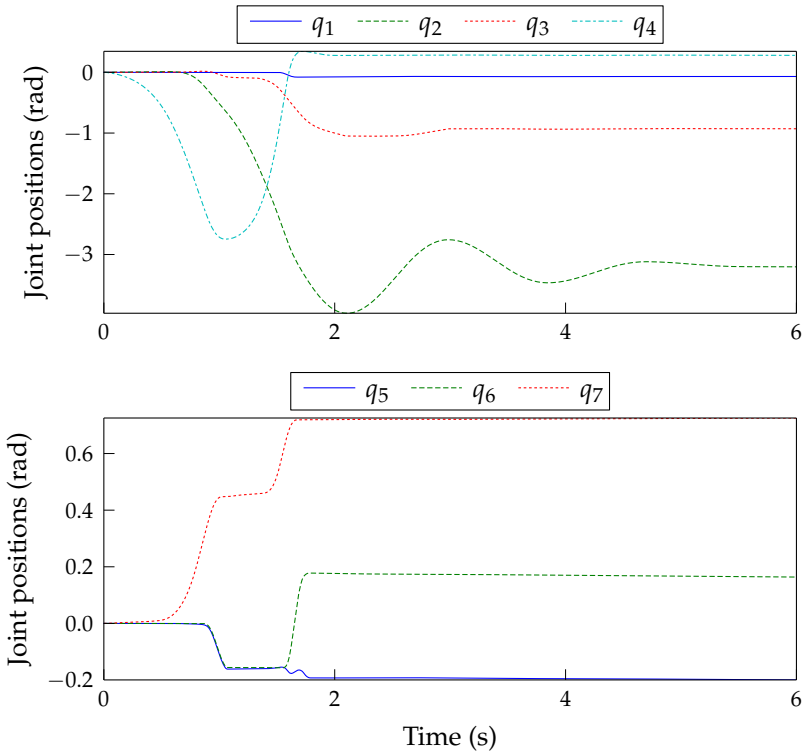

Figure 5.3: Joint trajectory of free fall robot simulation using feasible parameters $\boldsymbol{\beta}^{\star e}_{\text{WLS}}$.

the time is then computed from the simulated data entailing the values plotted in figure 5.2. Since the simulated system is passive, it is expected that the energy over time is non-increasing. Moreover, due to the dissipative nature of the friction it is expected that the energy decreases over time. The energy of the simulation using the classical WLS not only is not non-increasing but it also shows a high instability at the last milliseconds, as it can be see in figure 5.2. Therefor, as expected, the physically infeasible solution $\hat{\boldsymbol{\beta}}_{\text{WLS}}$ cannot be used for simulation purposes.

On the other hand, when testing any of the feasible solutions, the integration method is perfectly capable of performing the simulation as long as desired. The robot trajectory for the first 6 s of simulation using the parameters $\boldsymbol{\beta}^{\star e}_{\text{WLS}}$ is presented in figure 5.3 (for the other feasible solutions, the trajectory is similar). The trajectory is stable and shows a very realistic behavior

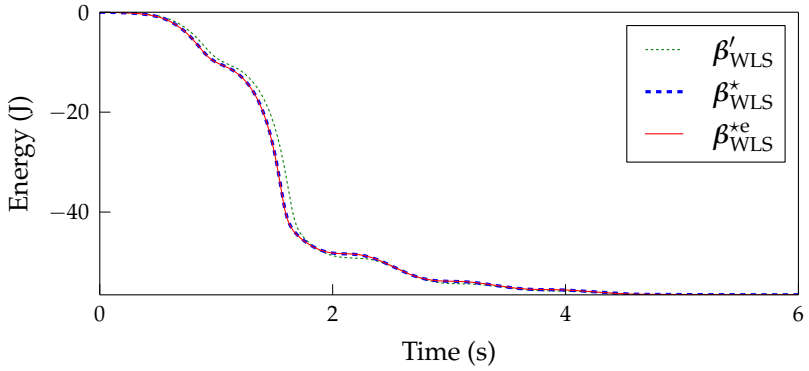

Figure 5.4: Total energy of free fall robot simulation using the three feasible parameter solutions.

when played by a 3D model of the WAM robot. Plots of the total energy for the three feasible solutions are presented in figure 5.4, showing a decreasing evolution over time until the minimal energy steady-state is reached.

5.2 Robot dynamic model-based controller

Both the classical and the LMI–SDP parameter estimations have been tested in the WAM robot in a model-based control application. A PD controller is designed to make the robot track the validation trajectory B, given in section §4.3.2. Although there are some similarities with the regression excitation trajectory tracking, some major differences distinguish this controller:

- the input reference for the controller is just the actual desired joint position, and no velocity or acceleration reference is provided;
- the linearization by non-linear feedback technique, presented in section §2.4.1, is used.

Using instantaneous position reference alone helps to better mimic the use cases where the reference cannot be known beforehand, like cases where the robot is being directly commanded by a human operator. The computed torque is given by

$$\tau_{\text{c}} = \hat{c}(q, \hat{\dot{q}}) + \hat{g}(q) + \hat{M}(q)\, a_{\text{c}} \,, \tag{5.8}$$

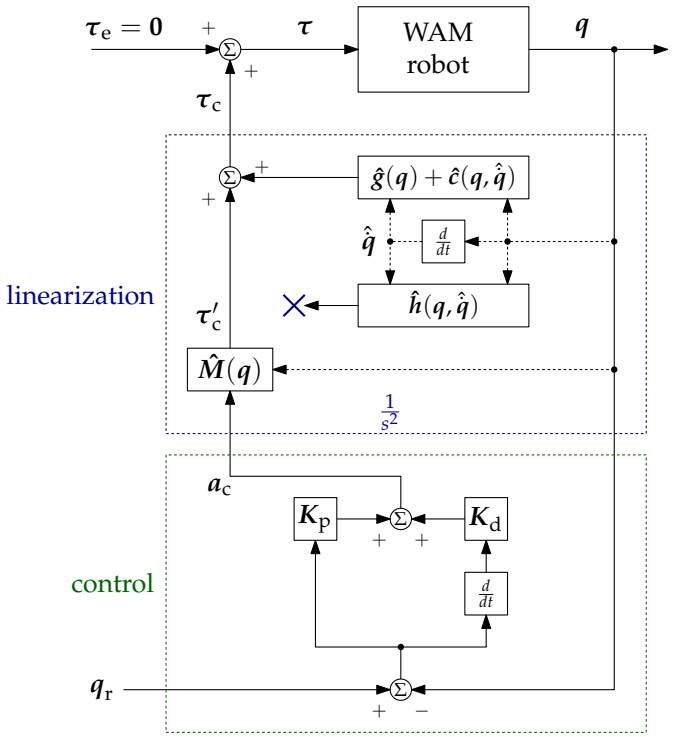

Figure 5.5: Control scheme for the WAM robot dynamic identification assessment.

which differs from (2.23) in the fact that no environment forces ($\hat{\tau}_e$) and no drive chain forces ($\hat{h}$) are compensated. The reference trajectory maintains the robot in free space, thus no robot–environment contact forces arise. Although the drive chain friction and rotor inertia parameters are estimated, the respective terms are not used in this controller due to their numerical instability near the zero velocity, being accounted as system disturbances instead. Nevertheless, their inclusion in the identification model is still crucial to reduce the error in the other parameters. Neglecting the disturbances, the PD controller sees a double integrator decoupled plant, as shown in figure 5.5. The proportional and derivative gains, K_{p_k} and K_{dk} given in table 5.1, respectively, are tuned for critically damped response but slow response time. Although the slow response times increase the tracking error, they entail more fair comparisons between different dynamic parameter estimations, as more importance is put on the model compensation than on the controller itself. In the experiment, the the control is turned on in three stages. First, at time

Table 5.1: PD control gains for the WAM robot dynamic identification assessment.

Joint k	$K_{\mathrm{p}k}$	$K_{\mathrm{d}k}$
1	156.3	25.0
2	156.3	25.0
3	156.3	25.0
4	156.3	25.0
5	123.5	22.2
6	123.5	22.2
7	123.5	22.2

$t = 0\,\mathrm{s}$, the robot is let at the first trajectory point with only gravity and Coriolis force compensation ($\hat{\boldsymbol{c}}(\boldsymbol{q}, \hat{\dot{\boldsymbol{q}}}) + \hat{\boldsymbol{g}}(\boldsymbol{q})$) turned on. Then, at $t = 2\,\mathrm{s}$ the PD control ($\hat{\boldsymbol{M}}(\boldsymbol{q})\,\boldsymbol{a}_{\mathrm{c}}$) is turned on while the reference position $\boldsymbol{q}_{\mathrm{r}}$ is kept at the initial point. Finally, at $t = 4\,\mathrm{s}$, trajectory B positions start to be feed to control reference. Both the classical infeasible parameters $\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$, and the feasible parameters $\boldsymbol{\beta}^{\star}_{\mathrm{WLS}}$ are tested in the linearization stack.

The recorded joint positions for the experiment with infeasible solution ($\hat{\boldsymbol{\beta}}_{\mathrm{WLS}}$) are plotted in figure 5.6. This estimation shows good gravity estimation (showing good gravity compensation), and the robot remains stopped when the controller is turned on with the static reference. However, the robot enters a divergent trajectory as soon as the controller starts to play the trajectory, leading to a high velocity motion which is stopped by the robot safety module. It is experimentally verified that such behavior occurs even if the control gains are decreased to near-zero values. On the other hand, the feasible parameters $\boldsymbol{\beta}^{\star}_{\mathrm{WLS}}$ show good model estimation through good initial gravity compensation and stable trajectory tracking, see figure 5.7. As an additional experiment, the optimal feasible parameters are tested with the higher control gains presented in table 5.2. With such gains, the system shows better trajectory tracking, as seen in figure 5.8. The feasible parameters show to model the robot very well, even without taking the drive chain dynamics into account in the controller. The experiment has also been conducted with the other feasible solutions ($\boldsymbol{\beta}'_{\mathrm{WLS}}$ and $\boldsymbol{\beta}^{\star\mathrm{e}}_{\mathrm{WLS}}$), showing little difference to the $\boldsymbol{\beta}^{\star}_{\mathrm{WLS}}$ solution.

No quantitative comparisons are made against the infeasible parameters, since those parameters are not stable enough to allow the record of useful

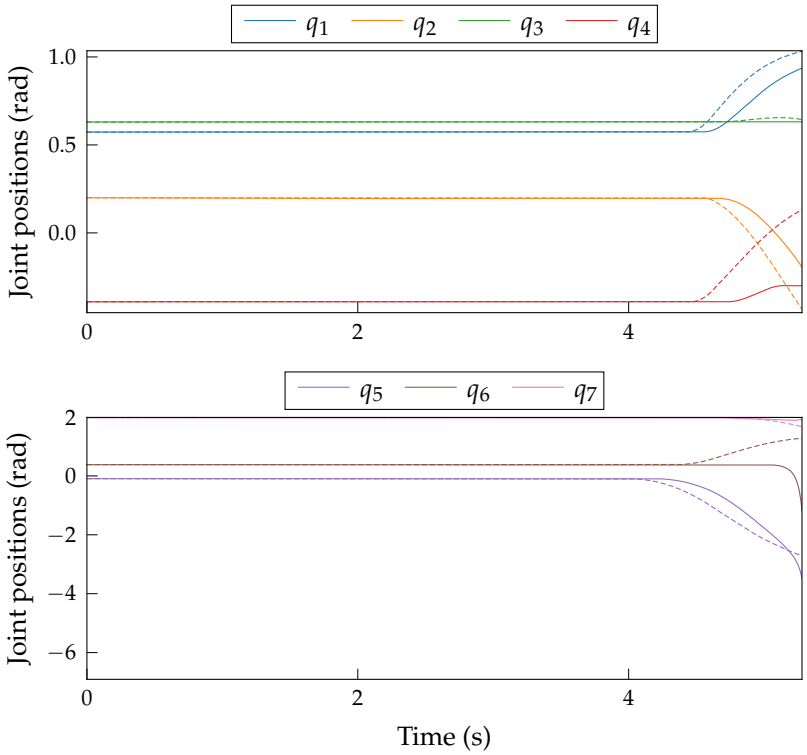

Figure 5.6: WAM robot joint trajectory for the assessment of classical infeasible parameters. Dashed lines are joint position reference; plain ones are recorded joint positions.

Table 5.2: Tighter PD control gains for assessment of the WAM robot optimal feasible parameters.

Joint k	K_{p_k}	$K_{\mathrm{d}k}$
1	2500.0	100.0
2	2500.0	100.0
3	2500.0	100.0
4	2500.0	100.0
5	625.0	50.0
6	625.0	50.0
7	625.0	50.0

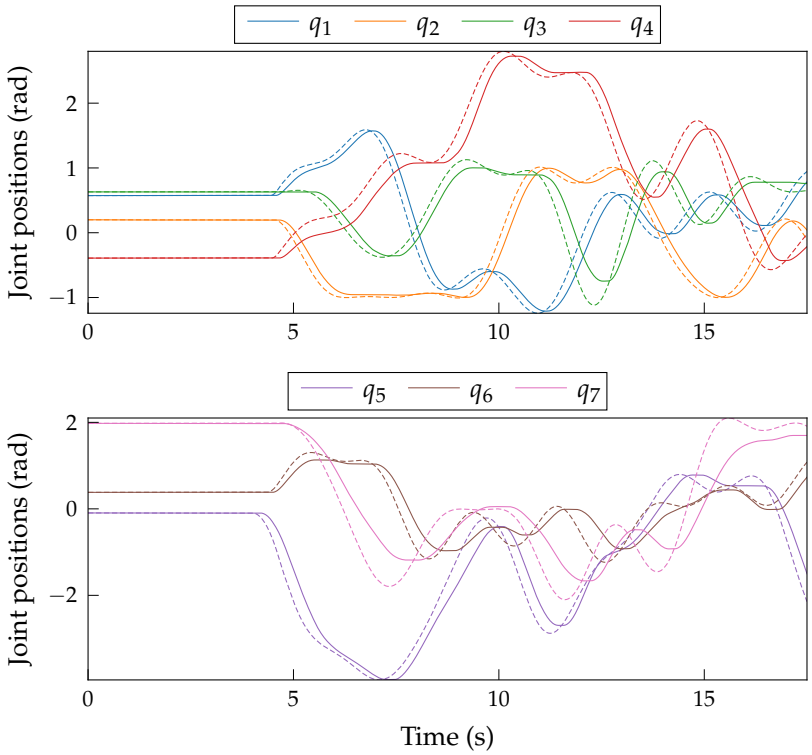

Figure 5.7: WAM robot joint trajectory for the assessment of optimal feasible parameters. Dashed lines are reference joint positions (one period of trajectory B); plain ones are recorded joint positions.

data. Once again, this shows that even being numerically close, these two estimations present very different outcomes. On a qualitative level, the difference between the classical infeasible solution and the feasible one is just abysmal, since, just as it happens to the simulation application, the classical parameter estimation is simply useless.

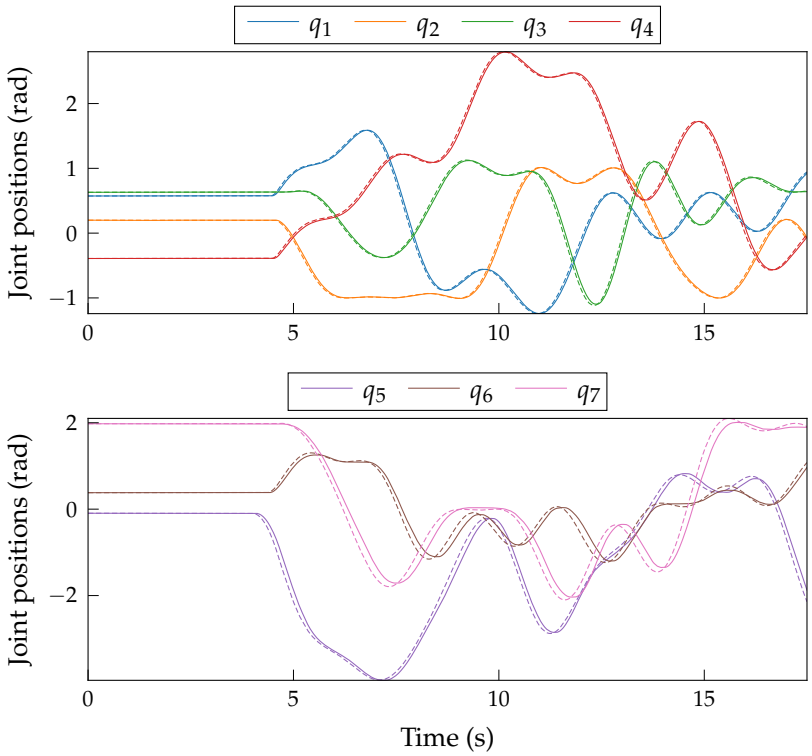

Figure 5.8: WAM robot joint trajectory using optimal feasible parameters and tight PD control gains. Dashed lines are reference joint positions; plain ones are recorded joint positions.

6 DISCUSSION AND COMPARISON

6.1 Physically feasible vs. infeasible solutions

Although it is a fact that the regression error of a feasible solution is higher than the error of the classical infeasible solution, it does not necessarily mean that feasible estimations entail worse models.

The question of whether estimations constrained to the feasible set are worse or better than physically infeasible ones can be related to the kind of desired model. If one needs a *prediction* model (Hollerbach et al. 2008), which seeks for the lowest prediction error, not mattering whether the identified parameters are feasible or not, then an unconstrained least squares (LS) solution can be used. However, if a *structural* model is needed (Hollerbach et al. 2008), where a meaningful physical system description is mandatory, then a strictly physically feasible identification has to be used. As seen in previous sections, simulation and model-based control shall use structural models. The higher regression error of regressions constrained to the feasible set is a minor effect, perfectly acceptable for the sake of having a structural model. Moreover, although it cannot be taken for guaranteed, it seems that feasible solution entail better models when tested outside the regression data set (see section §4.6.1).

Another important aspect is the fact that the difference between an infeasible estimation and a feasible one, as well as the difference between their regression errors, can be really low while the outcome of their application is completely different. This is observed in chapter 5, where a marginally infeasible parameter estimation has a completely unstable behavior whereas the marginally feasible solution works nicely. We can say that the feasible solution is marginally feasible since it is pulled against the feasible region boundary by the regression minimization. That is true for any constrained solution whose respective unconstrained solution is infeasible. We can also say the classical infeasible solution used in chapter 5 is marginally infeasible since the distance to the feasible set is very low (see section §4.5.2).

The more a infeasible parameter estimation is close to the feasible set, the more probable is that its instability effects appear in less robot postures. Hence, a marginally infeasible solution can be used without showing unstable behavior except for a little set of robot postures, thus being potentially

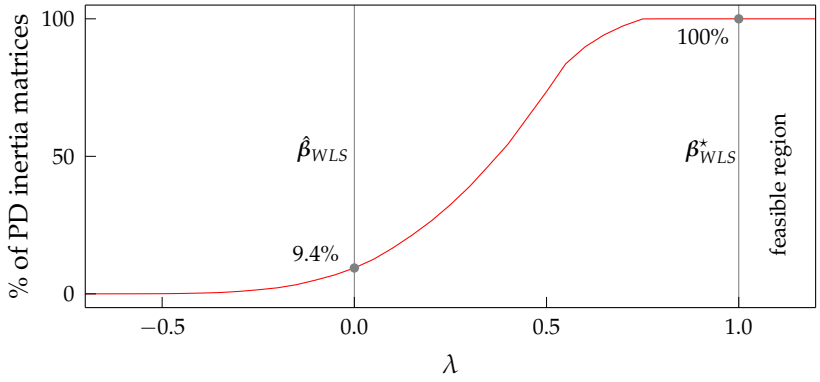

Figure 6.1: Ratio of positive definite (PD) inertia matrices, computed at random joint positions, as function of the "degree of infeasibility" of a parameter solution β_t given by $\beta_t(\lambda) = (1-\lambda)\hat{\beta}_{WLS} + \lambda\beta^{\star}_{WLS}$.

dangerous. Prior to starting this work, we have observed such behavior in a four-link version of the WAM, when a controller based on an infeasible parameter estimation showed instability just near a certain joint configuration. To illustrate this issue, a simple experience is performed in the sequel. Given some base dynamic parameter estimation, it is possible to compute the inertia matrix at large number of random joint positions and count the percentage of positions where the matrix is positive definite. If the parameters are feasible, then the inertia matrix will be always positive definite. If the matrix is not positive definite at least for a single joint posture, then the parameter estimation is physically infeasible. Hence, the ratio can be taken as a rough measure of infeasibility. Such measurement was applied to the dynamic model, regression data and solutions of chapter 4. Figure 6.1 plots the ratio of positive definite inertia matrices, computed on a set of 10000 random joint position, as function of a testing base parameter solution β_t given by

$$\beta_t(\lambda) = (1-\lambda)\hat{\beta}_{WLS} + \lambda\beta^{\star}_{WLS}\ , \tag{6.1}$$

which equates a line in the base parameter space passing through both $\hat{\beta}_{WLS}$ and $\beta^{\star}_{WLS}$. For $\lambda = 0$, β_t equals $\hat{\beta}_{WLS}$, and for $\lambda = 1$ the line crosses the feasibility region boundary at $\beta_t = \beta^{\star}_{WLS}$. As it can be seen, for this example, there is a zone close to the feasibility boundary where the inertia matrix is positive definite for a high number of joint positions while the parameters are still infeasible. Estimations in this zone can be dangerously misused as valid

Table 6.1: Base parameter estimations of the 3-link robot example provided by Yoshida and Khalil (2000)

β	$\hat{\beta}_{t1}$	$\hat{\beta}_{t2}$	β'_{t2}
$L_{1yy} + L_{2yy} + L_{3yy} + m_3$	**6.4**	**6.2**	6.200951
$L_{2xx} - L_{2yy} - m_3$	-5.48	-5.48	-5.479049
L_{2xy}	0.072	0.072	0.071966
$L_{2xz} - l_{3z}$	-0.087	-0.087	-0.086967
L_{2yz}	0.051	0.051	0.050999
$L_{2zz} + m_3$	5.6	5.6	5.600000
$l_{2x} + m_3$	6.5	6.5	6.500000
l_{2y}	-0.00075	-0.00075	-0.000750
$L_{3xx} - L_{3yy}$	-0.72	-0.72	-0.719049
L_{3xy}	-0.0098	-0.0098	-0.009819
L_{3xz}	-0.0098	-0.0098	-0.009817
L_{3yz}	-0.00045	-0.00045	-0.000450
L_{3zz}	0.72	0.72	0.720000
l_{3x}	0.95	0.95	0.949999
l_{3y}	0.015	0.015	0.014966

solution for model-based control or for simulation. Furthermore, it must be taken into account that non-positive definite inertia matrices are not the only effect of infeasible estimation. For instance, negative masses can lead to unstable gravity compensation, however, the effects of infeasible estimations in the model have not yet been fully studied.

The importance of not disregarding the physically feasibility aspect is clear. The propose LMI–SDP methods make a major contribution to deal with these issues. In the sequel, the advantages of the LMI–SDP approach over previous ones is presented.

6.2 Comparison to Yoshida methods

Yoshida and Khalil (2000) introduced a base parameter combination for a 3-link robot and presented two examples of base parameter vector values, $\hat{\beta}_{t1}$ and $\hat{\beta}_{t2}$ shown in table 6.1, which differ only in the value of the first base parameter. Using their method they have shown that $\hat{\beta}_{t1}$ is feasible whereas $\hat{\beta}_{t2}$ is not. To compare, our LMI–SDP method for feasibility checking (see section §3.3) is applied to this estimations. For $\hat{\beta}_{t1}$, a δ_d solution is found

implying physical feasibility. In the case of $\hat{\beta}_{t2}$ a certificate of infeasibility is issued by the SDP solver. Hence, our methods corroborate the results of Yoshida and Khalil. Furthermore, we applied the correction method of section §3.4 to $\hat{\beta}_{t2}$ for which the closest feasible parameter vector β'_{t2} presented in table 6.1 is found. The distance from β'_{t2} to $\hat{\beta}_{t2}$, measured in β space, is given by $\sqrt{u'} = 1.65 \times 10^{-3}$. As expected, β'_{t2} passes our feasibility test. For this example no regression data is available, thereby the feasible identification method could not be tested.

Compared to the methods for feasibility check presented in Yoshida and Khalil (2000), which include trial and error and visual graph steps, the LMI–SDP approach is more feasible in terms of practical implementation. For the LMI–SDP methods, there is a well established set of mathematical tools in several programming languages which enable implementations providing almost fully automatic execution. The application of a SDP solver to the LMI formulation of physical feasibility is, in a certain sense, a way to automate the rules described in Yoshida and Khalil (2000). Thereby, the complexity of the implementation and execution of our methods is not dependent on the size (DOF number) of the problem. That is probably the major issue with Yoshida et al. method, which is to some extent confirmed by the fact that no other work has resorted to those methods since then. Notwithstanding, the work of Yoshida et al. is the cornerstone in the physical feasibility problem definition.

6.3 Comparison to constrained regression methods

The idea of identifying the dynamic parameters through a regression constrained to feasible solutions, as presented in section §3.5, has its roots in the idea proposed by Mata et al. (2005), which was also explored by Farhat et al. (2008) and Ting et al. (2011). Comparing to the previous approaches, the proposed LMI–SDP solution takes a step forward in the mathematical formulation of the constrained regression problem. First, with little manipulation, the feasibility problem naturally fits the LMI representation. Second, our methods benefit from SDP advantages, noticeably the effective and efficient convergence to the global optimum, since SDP is the optimization technique which better exploits positive semidefinite constraints.

A series of comparison points show the advantages of the LMI–SDP ap-

proach:

Convexity Neither Mata et al. (2005), Ting et al. (2011) or related works, make the discussion, let alone the proof, on whether the constrained regression problem is convex or non-convex. In the LMI–SDP approach the proof of convexity naturally arises from the fact that all problems which are writable as LMIs are convex. It is important to know that a given optimum is for sure the global and unique optimum.

Constraints definition In all previous works, there is some mapping from the inertia tensors to another representation in order to define the positive definiteness constraints. Those mappings are based either in eigenvalues decomposition (Mata et al. 2005), Sylvester's criterion (Farhat et al. 2008) or Cholesky decomposition (Ting et al. 2011). In the LMI–SDP approach, there is a simple transformation using the Schur complement conditions to allow LMI representation, however, the final constraint is still a positive definite constraint. Such constraint is directly input to the solver, thus requiring no formulation of custom constraint equations or constraint penalty functions like in the other methods.

Effectiveness and efficiency The use of general non-linear programming to tackle positive definite constraints, although perfectly possible, is sub-optimal in the sense that the particular properties of the problem are not properly explored. That leads to both reduced effectiveness and reduced efficiency, i.e., worst approximations to the optimal solution and higher consumption of computational resources, respectively. These are the same reasons why one wouldn't usually solve a linear programming problem with a quadratic programming solver.

Method application scope The previous works present methods to perform regression constrained to the feasible region. On the other hand, the LMI–SDP framework we propose provides not only a constrained regression method, but also formulates a feasibility verification method, and correction method for infeasible prior estimations. It takes special advantage of SDP capabilities for checking the feasibility of given regions.

Extensibility The methods we propose can be easily extended to include additional parameters and respective constraints, as seen in section §3.2.1,

as well as to include additional constraints in the inertial parameters, as seen in section §4.5.4. Indeed, a large range of simple and complex conditions can be easily cast into the LMI formulation (Boyd et al. 1994). This can be an issue with the formulation of Ting et al. (2011). On the other hand, the formulation of Mata et al. (2005) is the most extensible one since it requires no special structure in the constraints, allowing, for example, the nonlinear friction model of Farhat et al. (2008).

Prior solution Mata et al. (2005) and Ting et al. (2011) methods use a prior solution, be it the unconstrained regression one, be it a CAD estimation. Our approach needs no prior solution.

Numerical stability Regression in the standard dynamic parameters entails non-unique solutions which can be a problem for the optimization solver. The use of a prior solution for which the regression solution distance is minimized can turn out to be fundamental to add numerical stability to the methods (Ayusawa and Nakamura 2010). Our methods have shown good numerical stability in all the experiments.

Parameter space Although the proposed LMI–SDP framework is mainly formulated in the base–dependent $(\boldsymbol{\beta}, \boldsymbol{\delta}_\mathrm{d})$ parameter space, it is straightforwardly formulated on the standard parameter space $\boldsymbol{\delta}$ as seen in section §3.6. This allows a greater freedom for choosing the space which better suits each application.

6.3.1 Point-mass method

Ayusawa and Nakamura (2010) presented a clever approach to simplify the physical feasibility problem at the expense of obtaining solutions which are only limited approximations to the optimal ones. They identified the feasible region as being a convex cone, and the simplifications approximate its interior with by convex polyhedron. The regression can then be performed very effectively and efficiently with quadratic programming (QP) which can be seen as a particular case of SDP. Compared to the LMI–SDP methods, this approach has some drawbacks:

Solution approximation The more obvious one is the fact that parameter estimations will be only approximations to the real optimum. This factor

can, however, be controlled by choosing different number and position of mass points.

Application as a feasibility test Although it is possible to map any given dynamic parameter solution $\hat{\delta}$ into the point mass parameterization $\hat{p}$ (assuming that $\boldsymbol{R}$ of (2.74) is full rank), it is not straightforward to assess the feasibility of $\hat{\delta}$ from the mass point parameters positiveness, since the map is not unique. Even if it was possible to guarantee that no positive $\hat{p}$ corresponds to the given $\hat{\delta}$, the infeasibility could not be inferred since the $\hat{\delta}$ could lie in a point within the feasibility region $\mathcal{D}$ which was not covered by the approximation polyhedron.

Method tuning The mass point method present many variables which have to be chosen and tuned. They include the number of mass points, their position, and the weighting factors, which can lead to very different results and requires additional tuning efforts. On the other hand, our LMI–SDP approaches require no method parameters to be tuned.

Other Drawbacks As well as the other constrained regression methods mentioned above, the mass point method also present drawbacks relatively to numerical stability, parameters space and application scope.

For experimental comparison, the point-mass method was implemented and applied to the identification data of the 7-link WAM robot presented in chapter 4. For the sake of easier implementation, only the inertial parameters (see (2.18)) are modeled. The data used for regression is weighted using the same technique of section §4.4.1. The implementation takes link spatial limits (see table 4.8) and the LS solution into account, as presented in the original article. The final solution, $\hat{\delta}_{\text{Ayusawa}}$, is obtained by a QP solver, being its physical feasibility double-checked by our method. The prediction relative error (see (4.37)) of the solution is presented in table 6.2 along with the infeasible WLS and the feasible LMI–SDP solutions for the same model and regression data. The increase from the classical solution to the point-mass error is low, however it is noticeably and relatively higher when compared to the small increase of the error of the best feasible solution (given by LMI–SDP), 0.61 ($= 20.25 - 19.64$) against 0.04 ($= 19.68 - 19.64$), respectively.

Table 6.2: Comparison of relative error percentage of predicted torque given by WLS, LMI–SDP, and point-mass method (Ayusawa and Nakamura 2010).

	% relative error
$\hat{\beta}$	19.64
$\beta^{\star}$	19.68
$\hat{\delta}_{\text{Ayusawa}}$	20.25

6.3.2 Effectiveness, efficiency and scalability comparison

To compare effectiveness, efficiency and scalability of our SDP–LMI approach, the method of Mata et al. (2005) was also implemented in addition to the implementation of the Ayusawa and Nakamura (2010) method. The method of Mata et al. (2005) is implemented using a SQP solver and uses the gradient functions from both the objective least squares equation and the nonlinear inequality constraints given by the Sylvester's criterion (Farhat et al. 2008). The method of Ting et al. (2011) is not implemented, however, since it uses generic nonlinear optimization it is expected to be comparable to the method of Mata et al. (2005) in terms of effectiveness and efficiency.

In order to test the methods at different scales, simulated data is generated for $N = 1$ up to $N = 40$ number of links. The regressor matrix data is generated randomly considering 10 parameters per link and 1000 measurement points, thus leading to a matrix of size $1000 \cdot 10 \times 10N$. The regressand vector data is generated from the regressor matrix using a randomly generated parameter vector, and adding a normally distributed noise. With exception for the $N = 1$ case, the classical OLS solutions show to always be physically infeasible when tested with our methods. In figure 6.2, the regression error of the different constrained regression methods are compared relatively to the OLS solution error. The time took by the solver of each method is plotted in figure 6.3. These times and errors must be taken only as a rough indication of the relation between the methods. As it can be seen, the LMI–SDP approach always gives the lowest possible error as it always finds the optimal solution (up to a 10^{-6} error), and its performance follows a predicable curve as the number of links grow. The error curve of ours method is indeed the lower bound for any possible feasibility method. The point-mass method of Ayusawa and Nakamura (2010) is as fast as the classical OLS method while providing always feasible solutions. However, its approximate nature entails

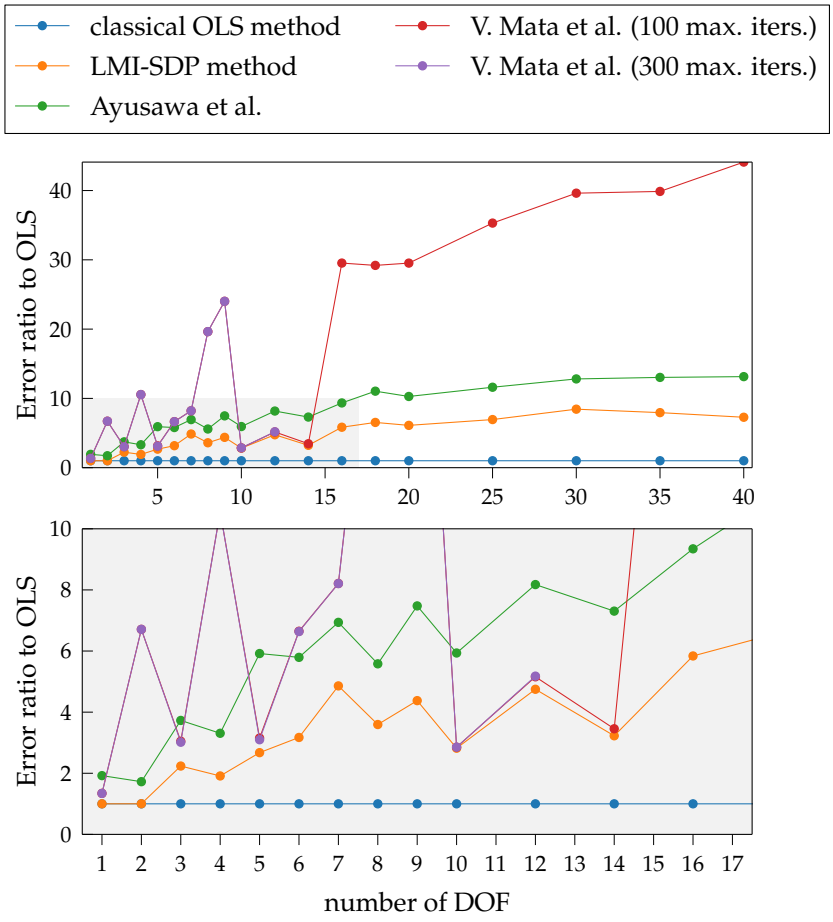

Figure 6.2: Comparison of method regression errors relatively to the OLS infeasible solution error. (Top plot: overview; bottom plot: zoom into first 17 experiments.)

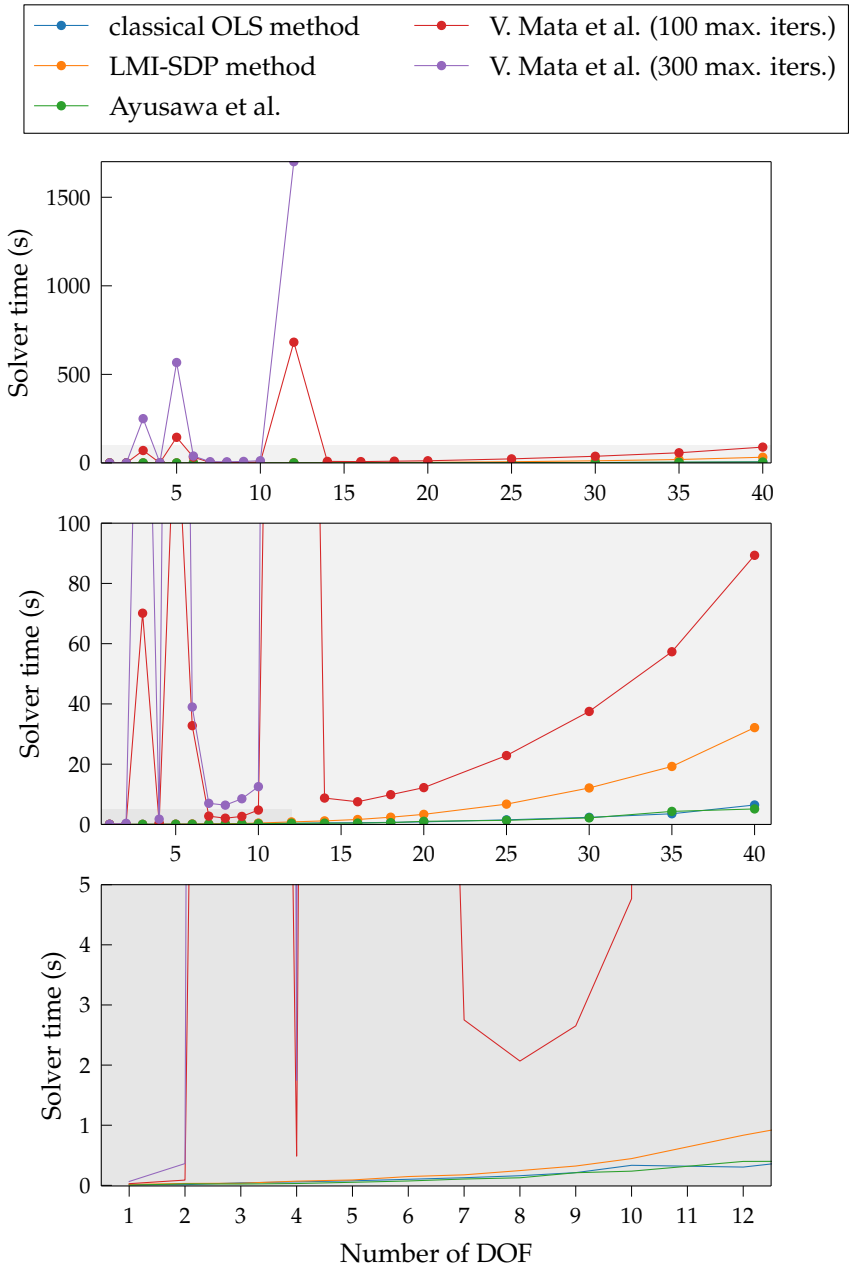

Figure 6.3: Comparison of solver times of different regression methods. (Top plot: overview; bottom plots: detail zoom in.)

non-optimal solutions in what concerns to regression error, leading to a grow in the error from the classical infeasible solution around two times the one from our method. The general non-linear optimization of Mata et al. (2005) shows a non-predicable behavior, entailing solutions which sometimes are close to the optimal and other times are really far from it. Moreover, the times are always higher the the other methods. Such times are dependent on a parameter to limit the number of iterations which shows to give no visible advantage for higher number as the solution errors are the same. The experiments for the number of iteration limit equaling 300 are stopped at 12 links as the times were too high after that.

Our LMI–SDP method shows to be adequate for high number of link robots, like humanoids, being capable to find the best solution in very short time. It may even be possible that with proper modifications it can be used in real-time, similarly to Ayusawa et al. (2011).

6.4 Comparison to model reduction methods

The works of Díaz-Rodríguez et al. (2010) and Gautier et al. (2013a) follow the idea that infeasibility of a given regression is due to a set of parameters not well identifiable. Hence, instead of reducing the regression solution space to the physical feasibility region, they reduce the number of parameters by discarding the less identifiable ones. It is then expected that the classical regression solution for the reduced model is feasible.

This approach has two major issues when compared to the LMI–SDP methods. The first one is that it does not solve the problem of testing the feasibility in the base parameter space. To overcome that, Gautier et al. (2013a) do the optimization in the standard parameter space, minimizing the distance to a prior estimation in this space, while Díaz-Rodríguez et al. (2010) uses a brute-force search in the base space. The second major drawback is that the model reduction process does not guarantee that a feasible solution will ever be found, or, at least, that such solution is found in a model not overly reduced and without too high regression errors.

The iterative reduction method of Díaz-Rodríguez et al. (2010) has been implemented by us and is tested with the WAM weighted regression data set (see section §4.4.1). The feasibility check step, however, is performed by our LMI–SDP approach instead of the original solution proposed by Díaz-

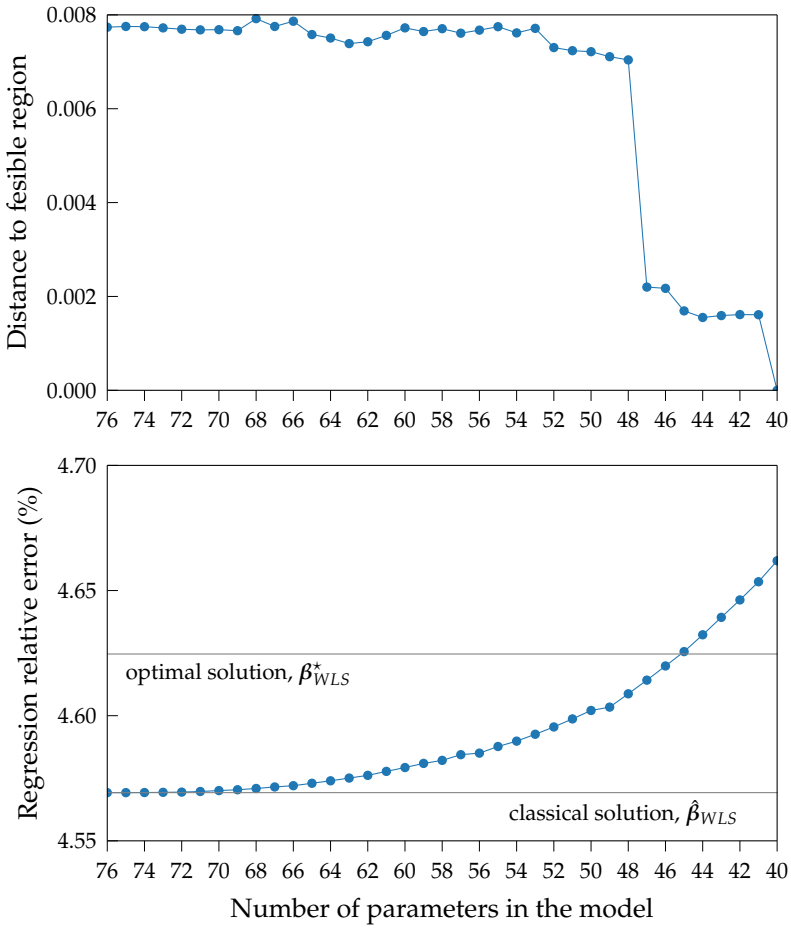

Figure 6.4: Application of the model reduction method of Díaz-Rodríguez et al. (2010) to the WAM data set. (Distance measured in the base parameter space.)

Rodríguez et al. Figure 6.4 plots the regression error and the solution distance to feasible region as function of the number of parameters excluded from the model, from a total of 76 base parameters (see section §4.2). The distance to feasible region is computed by the method of section §3.4, which finds the feasible solution closer to the given one, hence entailing the minimal distance to the feasible region. When the classical solution is feasible, that distance is zero. As in can be seen in the plots, it is possible, with a reduction of 36 base parameters, to the classical LS solution to become feasible. While the number of discarded parameters is almost a half of the original number, the regres-

sion error does not grows too much. The final regression relative error of the reduced model is 4.66%, just slightly higher when compared to the non-reduced model LS solution, 4.57%, and, as expected, higher than the optimal value given by our LMI–SDP method, 4.62% (see table 4.10). Therefore, reduction model techniques are effective in this data set, however, it must not be forgotten that such techniques are by nature not guaranteed to always work, unlike the LMI–SDP approach. Moreover, as we have shown, LMI–SDP methods can also be used to improve these model reduction methods and clearly present advantages over the techniques they propose for the feasibility checking of each solution.

7

PRACTICAL IMPLEMENTATION

In order to numerically evaluate the dynamic model, it is common to use the recursive Newton–Euler (RNE) approach, which is simple and computationally more efficient than closed-form equations. Starting in the eighties, there had been the development of software packages which perform symbolic optimizations on the equations, customizing them for given robots. Examples are *ARM* (Murray and Neuman 1988), *EMDEG* (Burdick 1986), and *SYM* (Timcenko et al. 1991) software. Those programs generate closed-form expressions with intermediate variables, seeking for a lower number of arithmetic operations. With the raise of general computer algebra systems (CAS), many have been used as a base to symbolic robot modeling packages. Examples are Spong's *Robotica* (Nethery and Spong 1994), and IRCCyN's *SYMORO+* (Khalil and Creusot 1997), both based on *Mathematica*. Furthermore, there are also more general robotics software which also include symbolic expression generation, e.g., *SYMOFROS* from SoftSim Technologies (Piedboef et al. 1999) based on *Maple*, and Corke's *MATLAB Symbolic Toolbox* (Corke 2011). With exception to *SYMORO+*, all of those packages are free and open source. Still, all of them work on top of proprietary/payed software. With respect to open source based software, there are several robotic packages which, however, only provide generic/numerical, thus slower, model implementations. Nowadays computer power is relatively high, and generic methods can numerically compute robot models very quickly. Even so, symbolically generated custom models always represent a great decrease in computation times, which is desirable or even a requirement in high frequency controllers and real-time simulations. Additionally, the symbolic expressions are useful for research and education.

This chapter presents the software developed to support the theoretical and experimental work of this thesis. Notwithstanding, the software has been developed not only for this work but also, and mainly, as an open tool available to future projects and to other researchers and developers.

7.1 SymPyBotics: robot symbolic modeling in Python

For the robot modeling used in the presented work, the *SymPyBotics* software (Sousa 2014b), a toolbox which generates symbolic custom equations of robot models for identification, control and simulation, has been developed. *SymPyBotics* is the evolution of the author's *SageRobotics* framework (Sousa and Cortesão 2012) which worked on top of *Sage Mathematics* software which by its turn is based on *Python*. Besides the change in the name, *SymPyBotics* is pure *Python*, using *SymPy* and *NumPy* libraries to do symbolical and numerical computations, respectively. Not only this framework is free and open source, it is also the first one to rely only upon free and open source software. The cost to use *SymPyBotics* in any kind of research, industry or education is minimal. Moreover, *Python* is a full-fledged programming language, cross-platform, very popular in engineering fields and with a really wide set of available libraries.

SymPyBotics key features are: custom symbolic model generation, code generation, and dynamic parameter identification. As input, *SymPyBotics* need the standard or modified DH parameters of a robot. Then *SymPyBotics* generates symbolic equations for geometric and kinematic direct models, and inverse dynamic model. It also generates dynamic terms, i.e., the inertia matrix, the Coriolis and centripetal forces term and the gravity forces term. The software includes dynamic parameter identification tools too. It generates the regressor of the linear to parameters model form, and computes the base parameter combination. Models can be obtained as single expressions or as expressions with intermediate variables. *SymPyBotics* is able to generate *C*, *Julia* and *Python* code from the expressions. With intermediate variables, common sub-expressions appear only once, thus reducing the number of operations in the generated code. *SymPyBotics* code is openly available in GitHub at `https://github.com/cdsousa/sympybotics`.

7.1.1 SymPyBotics internal structure

SymPyBotics toolbox can be used both as a library and as a end-user program. At its core there are the symbolical algorithms and the physic quantities symbolic representation. On a higher level there are routines which receive DH robot parameters as input and output generated code, ready to be used. Figure figure 7.1 presents a rough overview of *SymPyBotics* main packages

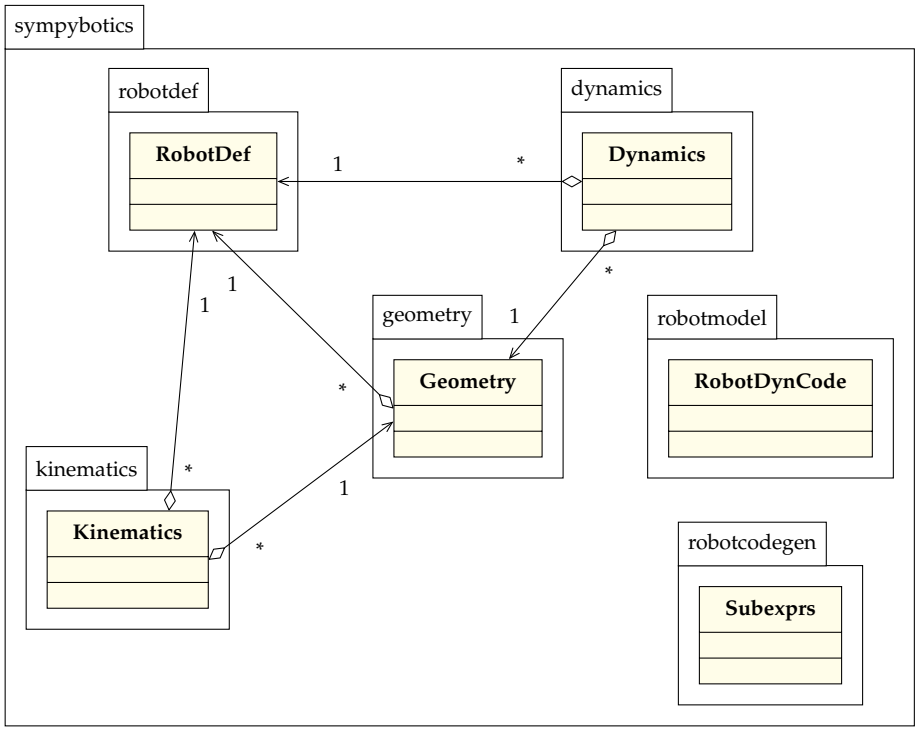

Figure 7.1: SymPyBotics UML package/class overview diagram.

and classes. The base of the framework is the `RobotDef` class, contained on the `robotdef` module. A `RobotDef` object is used to hold a given robot geometry definition, through DH parameters, and the related symbolic variables. The `geometry` module defines the `Geometry` class which generates and holds the direct geometric model for a given `RobotDef` object. The `Kinematics` class, contained on the `kinematics` module, receives one `RobotDef` object and one `Geometry` object and generates the respective direct kinematic model, i.e., the Jacobian matrices. Dynamics related code is in the `dynamics` module which, like the others, includes a `Dynamics` class to hold that model. This module also includes several standalone RNE functions which are used by the `Dynamics` class. On a higher level, there are the `robotmodel` and the `robotcodegen` modules. The `robotmodel` module includes the `RobotDynCode` class which generates the models and code for a given robot. It is dependent on the modules presented before and on the `robotcodegen` module for code generation. The `robotcodegen` module has tools to generate computer code from robot model equations.

7.1.2 SymPyBotics example

For demonstration purposes, an example of a possible use of *SymPyBotics* is shown next. The creation of a `RobotDef` instance just needs the α, a, d and θ DH parameters per link, and optionally a list of desired friction terms which can include "Coulomb", "viscous" and "offset". For a 2-DOF example robot, one can initialize an instance of `RobotDef` as follows:

```
>>> from sympybotics import q, RobotDef, RobotDynCode
>>> from sympy import pi
>>> rbtdef = RobotDef("Example Robot",  # name
...                   [(-pi/2, 0, 0, q+pi/2),  # DH parameters
...                    ( pi/2, 0, 0, q-pi/2)],
...                   dh_convention="standard")
>>> rbtdef.frictionmodel = {"Coulomb", "viscous"}
```

After this, the `rbtdef` object will contain symbols representing several quantities, as for example the joint positions, accessible through `rbtdef.q`, or the standard dynamic parameters, accessible through `rbtdef.dynparms()`:

```
>>> rbtdef.dynparms()
[L_1xx, L_1xy, L_1xz, L_1yy, L_1yz, L_1zz, l_1x, l_1y, l_1z,
 m_1, fv_1, fc_1, L_2xx, L_2xy, L_2xz, L_2yy, L_2yz, L_2zz,
 l_2x, l_2y, l_2z, m_2, fv_2, fc_2]
```

To generate the symbolic equations of geometric, kinematic and dynamic models, one can then use the high-level interface `RobotDynCode`:

```
>>> rbt = RobotDynCode(rbtdef, verbose=True)
generating geometric model
generating kinematic model
generating inverse dynamics code
generating gravity term code
generating coriolis term code
generating coriolis matrix code
generating inertia matrix code
generating regressor matrix code
generating friction term code
done
```

The returned `RobotDynCode` object contains sub-objects holding symbolic geometry model, for instance the end-effector to base transformation,

```
>>> rbt.geo.T[-1]
Matrix([
```

```
[-sin(q1)*sin(q2), -cos(q1),  sin(q1)*cos(q2), 0],
[ sin(q2)*cos(q1), -sin(q1), -cos(q1)*cos(q2), 0],
[         cos(q2),        0,          sin(q2), 0],
[               0,        0,                0, 1]])
```

the kinematic model Jacobians, for example

```
>>> rbt.kin.J[-1]
Matrix([
[0,         0],
[0,         0],
[0,         0],
[0, -cos(q1)],
[0, -sin(q1)],
[1,         0]])
```

and dynamic model terms in the `rbt.dyn` sub-object. The base dynamic parameters, computed through a numerical process, can be obtained by doing

```
>>> rbt.calc_base_parms()
>>> rbt.dyn.baseparms
Matrix([
[L_1yy + L_2zz],
[          fv_1],
[          fc_1],
[L_2xx - L_2zz],
[         L_2xy],
[         L_2xz],
[         L_2yy],
[         L_2yz],
[          l_2x],
[          l_2z],
[          fv_2],
[          fc_2]])
```

The matrices $\boldsymbol{P}_\mathrm{b}$, $\boldsymbol{P}_\mathrm{d}$ and $\boldsymbol{K}_\mathrm{d}$, introduced in section §2.5.1, become accessible through the attributes `dyn.Pb`, `dyn.Pd` and `dyn.Kd`, respectively.

7.1.3 Code generation

Although *SymPy* provides some functions to generate (and run) code from symbolic expressions, *SymPyBotics* supplies a set of tools to generate *Python*, *Julia* and *C/C++* source code specific for dynamic related expressions. For common robots, with several DOF, dynamic equations are usually complex

with a great number of operations and repeated sub-expressions. The complexity can be reduced by breaking down such expressions into smaller ones assigned to intermediate variables, in a sequential computer code form. This can be performed by a *common sub-expression elimination* algorithm like the `cse` function provided by *SymPy*. However, *SymPyBotics* implements a special common sub-expression elimination technique (implemented by the class `Subexprs`) which iteratively collects sub-expressions and substitute them by intermediate variables, thus taking advantage of the recursive nature of the RNE algorithm. Each symbolic term can be turned into code through the function `robot_code_to_func` from the `robotcodegen` module. For example, to generate *C/C++* code for the inverse dynamic function (2.21) of the example robot given above, one can do

```
>>> from sympybotics import robotcodegen
>>> robotcodegen.robot_code_to_func("C", rbt.invdyn_code,
                                    "tau_out", "invdyn",
                                    rbtdef)
```

which will output the following *C* code:

```
void invdyn( double* tau_out, const double* parms,
          const double* q, const double* dq, const double* ddq )
{
  double x0 = cos(q[1]);
  double x1 = sin(q[1]);
  double x2 = 9.81*x1;
  double x3 = dq[0];
  double x4 = x1*x3;
  double x5 = x0*x3;
  double x6 = dq[1]*parms[15] + parms[13]*x5 + parms[16]*x4;
  double x7 = -x4;
  double x8 = -ddq[0];
  double x9 = -x8;
  double x10 = dq[1]*x5 + x1*x9;
  double x11 = dq[1]*x7 + x0*x9;
  double x12 = dq[1]*parms[16] + parms[14]*x5 + parms[17]*x4;
  double x13 = dq[1]*parms[13] + parms[12]*x5 + parms[14]*x4;
  double x14 = 9.81*x0;
//
  tau_out[0] = dq[0]*parms[10] + parms[11]*sign(dq[0]) -
               parms[3]*x8 + x0*(ddq[1]*parms[13] + dq[1]*x12 +
               parms[12]*x11 + parms[14]*x10 + parms[19]*x2 +
               x6*x7) - x1*(-ddq[1]*parms[16] + dq[1]*x13 -
```

```
                parms[14]*x11 - parms[17]*x10 + parms[19]*x14 -
                x5*x6);
  tau_out[1] = ddq[1]*parms[15] + dq[1]*parms[22] +
                parms[13]*x11 + parms[16]*x10 - parms[18]*x2 +
                parms[20]*x14 + parms[23]*sign(dq[1]) - x12*x5 +
                x13*x4;
//
  return;
}
```

Here, `tau_out` is the τ_c output argument, `parms` is an input argument which must contain the numerical standard dynamic parameters ordered according to `rbtdef.dynparms()`, and `q`, `dq` and `ddq` are input arguments which must contain $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ respectively.

7.1.4 Performance measurements

To assess performance of generated code, experiments with code generation have been carried out for three robot models: the 6-DOF Stanford manipulator, the 6-DOF Puma 560, and the 7-DOF WAM robot. Table 7.1 resumes the measurements done for the inverse dynamics generated code. The table shows the time to generate the function code, the number of arithmetic operations (additions and multiplications) of the generated code, and the execution time of such code both for *C* and *Python* languages. The computer used in the experiments has a processor Intel i3-380M at 2.53 GHz. The number of additions and multiplications are in the same order of magnitude of the code obtained by *ARM* and *SYMORO* software (Murray and Neuman 1988; Khalil and Kleinfinger 1987). Moreover, the obtained performance is very high when compared to generic numerical routines which can be up to one hundred times slower. The generated code enabled the dynamic model-based controllers presented in previous sections to run with sampling times as low as 1 ms. It also enabled the WAM robot simulator to run in real-time.

7.2 Implementation of LMI–SDP methods

The practical implementation of the LMI–SDP methods has been done in a very high-level way from within a *Python* environment. The *SymPy* library, the *PyLMI-SDP* package (Sousa 2014a), and the *DSDP5* solver (Benson and Ye 2008) have been the base tools used to deploy those methods. The *PyLMI-*

Table 7.1: Performance measurements of inverse dynamics generated code.

	Stanford	Puma 560	WAM 7-DOF
code generation time (s)	10	14	15
adds. / muls.	393 / 477	454 / 576	493 / 626
Python function execution time (µs)	350	480	570
C function execution time (µs)	0.6	0.7	0.9

SDP package has been developed within this work, is open source and is openly available in GitHub at `https://github.com/cdsousa/PyLMI-SDP`. An overview of the LMI–SDP method implementation is given in the sequel. All presented methods share a standard set of steps that can be summarized as follows:

1st Each block of the LMI matrix diagonal is written using *SymPy* symbolic objects to represent each standard dynamic parameter (and the minimization slack variable if appropriate).

- Blocks for the physical feasibility of all links are similar, hence they can be initialized in a loop, following a template which is applied to the dynamic parameters corresponding to each link.
- Matrices from least squares minimizations (matrices $\boldsymbol{U}$) are created by mixing the numerical matrices and vectors with the vector of symbolic parameters. This can be done straightforwardly with *SymPy*.

2nd If the method is to be implemented in the base–dependent space, then, using *SymPy* symbol substitution routines, a substitution of symbol variables corresponding to the map $\mathfrak{m}_t$ of (3.10) and (3.11) is performed.

- In a practical way, that is implemented by the substitution of the symbols of independent standard parameters (δ_b) by linear combinations of symbols of both base and dependent parameters ($\boldsymbol{\beta}$ and

δ_d), as given by

$$\delta_b \quad \xrightarrow[\text{subs. by}]{} \quad \boldsymbol{\beta} - \boldsymbol{K}_d \delta_d \ .$$

Note that here, $\boldsymbol{\beta}$ refers to a vector of symbols rather than to a vector of linear expressions.

3rd Referring to the *PyLMI-SDP* package, the symbolic LMI diagonal blocks are converted to a set of numerical coefficient matrices, one for each block and for each variable. In pseudo-code that can be described by the following steps:

- For each symbolic variable a, create a numerical matrix for it;
- Create a numerical matrix for the constant terms;
- For each element (expression) in the input symbolic matrix:
  - Extract the constant term from the expression;
  - Write the constant term to the respective place in the matrix of constant terms;
  - For each symbolic variable a:
    - Extract the numerical coefficient of the variable a from the expression;
    - Write the coefficient to the respective place in the matrix of variable a;

4th An SDPA file is constructed from the numerical matrices and from the objective function coefficients, referring to a proper *PyLMI-SDP* routine.

5th The *DSDP5* solver is called with the created SDPA file as input, and its output is parsed back to the *Python* environment.

The use of SDPA files has the advantage of allowing a large number of SDP solvers to be used. Besides the *DSDP5* solver, the *CSDP* (Borchers 1999) and the *SDPA* (Yamashita et al. 2003) solvers have also been successful used by the LMI–SDP feasibility methods.

8 CONCLUSIONS

The scientific relevance of dynamic parameter estimation has been shown, being strongly correlated to model-based control and robot simulation. Good parameter identification is a key to allow advanced applications such as the robot-assisted medical tasks which were the primary motivation for this work.

It has been shown that although many regression techniques are available in the literature, there was an open problem about the physical feasibility of identified parameters. The problem poses particular challenges due to the non-identifiability of some dynamic parameters. In this context, a completely new approach to the problem has been proposed and analyzed. Firstly, the problem structure has been identified and has been formulated in the LMI framework. Then, new approaches using advanced SDP techniques have been proposed to solve the physical feasibility issue. Three LMI–SDP-based methods have been proposed to address three main applications:

- a method to test if a given parameter estimation is physically feasible or not;
- a method to compute the closest physically feasible solution for a given (infeasible) solution;
- and a method to perform optimal identification of base parameters (for a given data set) through least squares regression constrained to feasible solutions.

The methods have been implemented with open source software and have originated the packages *SymPyBotics* and *PyLMI-SDP*, which are free and open source. The proposed approach has been applied to the real case of the WAM robot identification, which presents physical feasibility issues when performed by classical techniques. The LMI–SDP methods have shown effective and efficient results. The optimal WAM parameter identification found through our methods has shown good behavior when applied to simulation and to experimental model-base control. The higher quality of this identification is not even measurable against the classical solution, since the latter has shown to be unusable in both simulation and control. The major importance of taking physical feasibility into account has been discussed and clarified

with further data. Finally, our methods have been compared with previous works, showing clear advantages in almost all aspects.

The LMI–SDP formulation of the physical feasibility issue opens new possibilities in the dynamic parameter identification research field. In the sequel, some possible future research directions are presented.

8.1 Future work

Study and model the impact of infeasible parameters in the dynamic terms While non-positive definite inertia matrices have been pointed as the problem of physically infeasible parameters, the effects are not limited to the inertia matrix (see section §6.1). It may be important to study all the impact of infeasible parameters in the dynamic model and in model-based applications.

Study and relate each base parameter and standard parameter to the infeasibility of the whole estimation The infeasibility issue has been addressed as a whole, but a formulation to understand the contribution of each standard or base parameter is yet not known. By the works of Díaz-Rodríguez et al. (2010) and Gautier et al. (2013a), and the results of section §6.4 it seems that there is a relation between poorly identified parameters and physical infeasibility, however, this relation has not yet been formulated nor even proved. Such study may be important to devise excitation techniques and excitation measures which entail regression data less prone to give infeasible solutions. On the other hand, the individual degree of infeasibility of each parameter can be used as a measurement for the low confidence of each parameter, along with the established standard deviation measure.

Integrate LMI–SDP techniques into other advanced regression methods An important aspect of the LMI–SDP formulation it that it is orthogonal to the majority of other identification techniques, as it was shown, for instance, with the reformulation from OLS to WLS in section §4.5.3. Another given example is the use of LMI–SDP techniques as a complement to the model reduction methods in section §6.4. Hence, the LMI–SDP formulation enables further research on its own but also has the possibility to be merged into established techniques. Among others, techniques making a more statistically

meaningful treatment of the regression data may possibly benefit from the physically meaningful approach of LMI–SDP formulation. Such techniques include the maximum-likelihood approach of Olsen and Petersen (2001) or even the Bayesian approach of Ting et al. (2011).

Study a real-time implementation of feasible identification SDP solvers use iterative techniques, hence, it may be possible to implement a real-time and iterative regression method constrained to feasible solutions as the one proposed by Ayusawa et al. (2011). Such method, although computationally more expensive, would provide optimal feasible solutions in real-time.

Extend the formulation to different kind of robots or systems Although the presented methods have been applied to the serial manipulator robot concept, no fundamental problem keeps them from being applied to other robotic devices. That includes tree robots, closed loop robots, parallel robots, and humanoids. As physical feasibility issues extend to the identification of other kind of systems of rigid bodies, the same happens to the LMI–SDP solution. Moreover, while we do not considered flexible structures, the developed methods can be extended to these cases too.

Research and develop an *on-the-fly* identification method Complete identification can only be done in base parameter space since only these parameters affect robot dynamics. Taking this idea a little further, we can say that for a given limited set of motions, the only identifiable parameters are the ones that in fact affect the dynamics in that region. Having these considerations, it may be possible to devise a controller that identifies the parameters on-the-fly. As the joints move, motion and torque data would be collected, and the parameters implied in the dynamics of such motion could be identified. The identification confidence would increase as more motion space is covered, and the controller gains would be adapted taking individual parameter confidence into account. The confidence could then be feed back to a trajectory reference generator which would try to excite the parameters with lower confidence. Classical regression approaches require all base parameters to be minimally well excited or else the solution cannot be found or may be physically infeasible. LMI-SDP regression techniques would guarantee feasible and stable solutions, hence allowing the initial lack of parameter knowledge.

REFERENCES

Armstrong, B. 1989. "On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics". *Int. J. Robotics Research* 8, no. 6 (): 28–48. doi:10.1177/027836498900800603. (Cit. on p. 21).

Atkeson, C. G., C. H. An, and J. M. Hollerbach. 1986. "Estimation of Inertial Parameters of Manipulator Loads and Links". *Int. J. Robotics Research* 5, no. 3 (): 101–119. doi:10.1177/027836498600500306. (Cit. on pp. 13, 18).

Ayusawa, Ko, and Yoshihiko Nakamura. 2010. "Identification of standard inertial parameters for large-DOF robots considering physical consistency". In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* 6194–6201. IEEE. doi:10.1109/IROS.2010.5649628. (Cit. on pp. 23, 30, 31, 94, 96).

Ayusawa, Ko, Gentiane Venture, and Yoshihiko Nakamura. 2011. "Real-time implementation of physically consistent identification of human body segments". In *Proc. IEEE Int. Conf. Robot. Autom.* 6282–6287. IEEE. doi:10.1109/ICRA.2011.5979903. (Cit. on pp. 31, 99, 115).

Benimeli, F., Vicente Mata, N. Farhat, and A. Valera. 2003. "Experimental set-up and some results in parameter identification in robots". In *RAAD'03, 12th International Workshop on Robotics in Alpe-Adria-Danube Region*. Cassino. (Cit. on p. 23).

Benson, Steven J., and Yinyu Ye. 2008. "DSDP5-Software for Semidefinite Programming". *ACM Transactions on Mathematical Software* 34, no. 3 (): 1–20. doi:10.1145/1356052.1356057. (Cit. on pp. 67, 109).

Bierstone, Edward, and Pierre D. Milman. 1988. "Semianalytic and subanalytic sets". *Publications Mathématiques de l'IHÉS* 67:5–42. (Cit. on p. 36).

Borchers, Brian. 1999. "CSDP, A C library for semidefinite programming". *Optimization Methods and Software* 11, no. 4 (): 613–623. doi:10.1080/10556789908805765. (Cit. on p. 111).

Box, George E. P., and Norman Richard Draper. 1987. *Empirical model-building and response surfaces*. New York: John Wiley & Sons. (Cit. on pp. 6, 7).

Boyd, Stephen, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. 1994. *Linear Matrix Inequalities in System and Control Theory*. Vol. 15. Studies in Applied Mathematics. Philadelphia: SIAM. (Cit. on pp. 37, 39, 94).

Burdick, J. 1986. "An algorithm for generation of efficient manipulator dynamic equations". In *Proc. IEEE Int. Conf. Robot. Autom.* 212–218. 33. doi:10.1109/ROBOT.1986.1087659. (Cit. on p. 103).

Calafiore, G, M Indri, and B Bona. 2001. "Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation". *Journal of Robotic Systems* 18, no. 2 (): 55–68. doi:10.1002/1097-4563(200102)18:2<55::AID-ROB1005>3.0.CO;2-O. (Cit. on pp. 21–23).

Collins, G E. 1975. "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition". In *Proceedings Second GI Conference on Automata Theory and Formal Languages*, ed. by H Brakhage, 33:134–183. Springer-Verlag. (Cit. on p. 36).

Corke, Peter I. 2011. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer. (Cit. on p. 103).

Cortesão, Rui, O. Khatib, and Jaeheung Park. 2006. "Real-time adaptive control for haptic telemanipulation with Kalman active observers". *IEEE Trans. Robot.* 22, no. 5 (): 987–999. doi:10.1109/TRO.2006.878787. (Cit. on p. 16).

Cortesão, Rui, Cristóvão Sousa, and Pedro Queirós. 2010. "Active impedance control design for human-robot comanipulation". In *American Control Conference*, 2805–2810. doi:10.1109/ACC.2010.5531398. (Cit. on p. 2).

Craig, John J. 2004. *Introduction to Robotics: Mechanics and Control*. Prentice Hall. (Cit. on pp. 4, 10).

Díaz-Rodríguez, Miguel, Vicente Mata, Ángel Valera, and Álvaro Page. 2010. "A methodology for dynamic parameters identification of 3-DOF parallel robots in terms of relevant parameters". *Mechanism and Machine Theory* 45, no. 9 (): 1337–1356. doi:10.1016/j.mechmachtheory.2010.04.007. (Cit. on pp. 31, 32, 99, 100, 114).

Farhat, Nidal, M.-A. Díaz, and Vicente Mata. 2007. "Dynamic parameter identification of parallel robots considering physical feasibility and nonlinear friction models". *12th IFToMM World Congress*. (Cit. on pp. 23, 28).

Farhat, Nidal, Vicente Mata, Álvaro Page, and Francisco Valero. 2008. "Identification of dynamic parameters of a 3-DOF RPS parallel manipulator". *Mechanism and Machine Theory* 43, no. 1 (): 1–17. doi:10.1016/j.mechmachtheory.2006.12.011. (Cit. on pp. 22, 28, 92–94, 96).

Featherstone, Roy. 2010. "A beginner's guide to 6-D vectors (Part 1)". *IEEE Robot. Automat. Mag.* 17, no. 3 (): 83–94. doi:10.1109/MRA.2010.937853. (Cit. on p. 38).

Gautier, M. 1996. "A comparison of filtered models for dynamic identification of robots". In *Proceedings of IEEE Conference on Decision and Control*, 1:875–880. IEEE. doi:10.1109/CDC.1996.574537. (Cit. on p. 22).

Gautier, Maxime. 1997. "Dynamic identification of robots with power model". In *Proc. IEEE Int. Conf. Robot. Autom.* 3:1922–1927. IEEE. doi:10.1109/ROBOT.1997.619069. (Cit. on pp. 22, 61, 65).

— . 1986. "Identification of robot dynamics". In *Proceedings of the IFAC symposium: Theory of robots*, 351–356. Vienna. (Cit. on p. 18).

— . 1990. "Numerical calculation of the base inertial parameters of robots". In *Proc. IEEE Int. Conf. Robot. Autom.* 1020–1025. Cincinnati. doi:10.1109/ROBOT.1990.126126. (Cit. on p. 20).

— . 1991. "Numerical calculation of the base inertial parameters of robots". *Journal of Robotic Systems* 8, no. 4 (): 485–506. doi:10.1002/rob.4620080405. (Cit. on pp. 20, 54).

Gautier, Maxime, Sebastien Briot, and Gentiane Venture. 2013a. "Identification of consistent standard dynamic parameters of industrial robots". *IEEE/ASME Int. Conf. Adv. Intell. Mechatronics* (Wollongong) (): 1429–1435. doi:10.1109/AIM.2013.6584295. (Cit. on pp. 32, 99, 114).

Gautier, Maxime, Alexandre Janot, and Pierre-Olivier Vandanjon. 2013b. "A New Closed-Loop Output Error Method for Parameter Identification of Robot Dynamics". *IEEE Trans. Control Syst. Technol.* 21, no. 2 (): 428–444. doi:10.1109/TCST.2012.2185697. (Cit. on p. 22).

Gautier, Maxime, and W. Khalil. 1990. "Direct calculation of minimum set of inertial parameters of serial robots". *IEEE Trans. Robot. Autom.* 6, no. 3 (): 368–373. doi:10.1109/70.56655. (Cit. on pp. 19, 20, 27).

Gautier, Maxime, and Wisama Khalil. 1992. "Exciting Trajectories for the Identification of Base Inertial Parameters of Robots". *Int. J. Robotics Research* 11, no. 4 (): 362–375. doi:10.1177/027836499201100408. (Cit. on pp. 21, 58, 62).

Gautier, Maxime, and P. Poignet. 2001. "Extended Kalman filtering and weighted least squares dynamic identification of robot". *Control Engineering Practice* 9 (12): 1361–1372. (Cit. on pp. 22, 65).

Gautier, Maxime, and Gentiane Venture. 2013. "Identification of standard dynamic parameters of robots with positive definite inertia matrix". *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* (): 5815–5820. doi:10.1109/IROS.2013.6697198. (Cit. on p. 32).

Gilbert, George T. 1991. "Positive Definite Matrices and Sylvester's Criterion". *The American Mathematical Monthly* 98, no. 1 (): 44. doi:10.2307/2324036. (Cit. on p. 25).

Goldman, A. J., and Motakuri Ramana. 1995. "Some geometric results in semidefinite programming". *Journal of Global Optimization* 7, no. 1 (): 33–50. doi:10.1007/BF01100204. (Cit. on pp. 37, 40).

Hartenberg, Richard S, and Jacques Denavit. 1955. "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices". *Transaction ASME J Appl Mech* 22:215–221. (Cit. on p. 9).

Hogan, Neville. 1985. "Impedance control: An approach to manipulation: Part I - theory". *Journal of Dynamic Systems, Measurement, and Control* 107 (1): 1. doi:10.1115/1.3140702. (Cit. on p. 16).

Hollerbach, John M., Wisama Khalil, and Maxime Gautier. 2008. "Model Identification". Chap. 14 in *Handbook of Robotics*, ed. by Bruno Siciliano and O. Khatib, 321–344. Springer. (Cit. on p. 89).

Janot, A., P.O Vandanjon, and Maxime Gautier. 2009. "Using robust regressions and residual analysis to verify the reliability of LS estimation: Application in robotics". In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* 1962–1967. IEEE. doi:10.1109/IROS.2009.5354469. (Cit. on p. 22).

Janot, Alexandre, Pierre-Olivier Vandanjon, and Maxime Gautier. 2014. "A Generic Instrumental Variable Approach for Industrial Robot Identification". *IEEE Trans. Control Syst. Technol.* 22, no. 1 (): 132–145. doi:10.1109/TCST.2013.2246163. (Cit. on p. 22).

Khalil, W., and F. Bennis. 1995. "Symbolic Calculation of the Base Inertial Parameters of Closed-Loop Robots". *Int. J. Robotics Research* 14, no. 2 (): 112–128. doi:10.1177/027836499501400202. (Cit. on p. 20).

Khalil, W., and J.-F. Kleinfinger. 1987. "Minimum operations and minimum parameters of the dynamic models of tree structure robots". *IEEE J. Robot. Autom.* 3, no. 6 (): 517–526. doi:10.1109/JRA.1987.1087145. (Cit. on pp. 19, 109).

Khalil, W, Senior Member, and F Bennis. 1994. "Comments on "Direct Calculation of Minimum Set of Inertial Parameters of Serial Robots"". *IEEE Trans. Robot. Autom.* 10 (I): 78–79. (Cit. on p. 20).

Khalil, Wisama, and Denis Creusot. 1997. "SYMORO+: A system for the symbolic modelling of robots". *Robotica* 15, no. 2 (): 153–161. doi:10.1017/S0263574797000180. (Cit. on p. 103).

Khalil, Wisama, and Etienne Dombre. 2002. *Modeling, identification & control of robots*. London: Hermes Penton. (Cit. on pp. 4, 18, 32, 62).

Khatib, O. 1987. "A unified approach for motion and force control of robot manipulators: The operational space formulation". *IEEE J. Robot. Autom.* 3, no. 1 (): 43–53. doi:10.1109/JRA.1987.1087068. (Cit. on p. 15).

Lamy, X., F. Colledani, F. Geffard, Y. Measson, and G. Morel. 2009. "Achieving efficient and stable comanipulation through adaptation to changes in human arm impedance". In *Proc. IEEE Int. Conf. Robot. Autom.* 265–271. IEEE. doi:10.1109/ROBOT.2009.5152294. (Cit. on p. 2).

Maes, P., J.-C. Samin, and P.-Y. Willems. 1989. "Linearity of Multibody Systems with Respect to Barycentric Parameters: Dynamics and Identification Models Obtained by Symbolic Generation". *Mechanics of Structures and Machines* 17, no. 2 (): 219–237. doi:10.1080/15397738909412817. (Cit. on pp. 12, 13).

Marescaux, J., J. Leroy, M. Gagner, F. Rubino, D. Mutter, et al. 2001. "Transatlantic robot-assisted telesurgery". *Nature* 413, no. 6854 (): 379–80. doi:10.1038/35096636. (Cit. on p. 2).

Mata, V., F. Benimeli, N. Farhat, and A. Valera. 2005. "Dynamic parameter identification in industrial robots considering physical feasibility". *Journal of Advanced Robotics* 19 (1): 101–120. (Cit. on pp. 23, 27, 28, 30, 92–94, 96, 99).

Mayeda, H., Koji Yoshida, and K. Osuka. 1990. "Base parameters of manipulator dynamic models". *IEEE Trans. Robot. Autom.* 6, no. 3 (): 312–321. doi:10.1109/70.56663. (Cit. on pp. 19, 20, 27).

Murray, J.J., and C.P. Neuman. 1988. "Organizing customized robot dynamics algorithms for efficient numerical evaluation". *IEEE Trans. Syst. Man Cybern.* 18 (1): 115–125. doi:10.1109/21.87059. (Cit. on pp. 103, 109).

Murray, Richard M., Zexiang Li, and S. Shankar Sastry. 1994. *A mathematical introduction to robotic manipulation*. CRC Press. (Cit. on p. 4).

Nakanishi, J., R. Cory, M. Mistry, J. Peters, and S. Schaal. 2008. "Operational Space Control: A Theoretical and Empirical Comparison". *Int. J. Robotics Research* 27, no. 6 (): 737–757. doi:10.1177/0278364908091463. (Cit. on p. 29).

Nesterov, Yurii, and Arkadii Nemirovskii. 1987. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM. (Cit. on p. 41).

Nethery, J.F., and M.W. Spong. 1994. "Robotica: A Mathematica package for robot analysis". *IEEE Robot. Automat. Mag.* 1, no. 1 (): 13–20. doi:10.1109/100.296449. (Cit. on p. 103).

Olsen, M. M., J. Swevers, and W. Verdonck. 2002. "Maximum Likelihood Identification of a Dynamic Robot Model: Implementation Issues". *Int. J. Robotics Research* 21, no. 2 (): 89–96. doi:10.1177/027836402760475379. (Cit. on p. 22).

Olsen, M.M., and H.G. Petersen. 2001. "A new method for estimating parameters of a dynamic robot model". *IEEE Trans. Robot. Autom.* 17 (1): 95–100. doi:10.1109/70.917088. (Cit. on pp. 22, 115).

Palep, Jaydeep H. 2009. "Robotic assisted minimally invasive surgery". *Journal of Minimal Access Surgery* 5, no. 1 (): 1–7. doi:10.4103/0972-9941.51313. (Cit. on p. 2).

Park, F. C., J. E. Bobrow, and S. R. Ploen. 1995. "A Lie Group Formulation of Robot Dynamics". *Int. J. Robotics Research* 14, no. 6 (): 609–618. doi:10.1177/027836499501400606. (Cit. on p. 38).

Park, Kyung-Jo. 2006. "Fourier-based optimal excitation trajectories for the dynamic identification of robots". *Robotica* 24, no. 05 (): 625–633. (Cit. on p. 21).

Peters, Jan. 2010. "Using model knowledge for learning inverse dynamics". In *Proc. IEEE Int. Conf. Robot. Autom.* 2677–2682. IEEE. doi:10.1109/ROBOT.2010.5509858. (Cit. on p. 4).

Pham, C.M., and M. Gautier. "Essential parameters of robots". In *Proceedings of IEEE Conference on Decision and Control*, 2769–2774. IEEE. doi:10.1109/CDC.1991.261862. (Cit. on p. 32).

Piedboef, C. J., M. Doyon, P. Langlois, and R. L'Archevêque. 1999. "SYMOFROS: A flexible dynamics modelling software". In *Proc. 5th Int. Symp. Artificial Intelligence, Robotics and Automation in Space*, 440:709. Noordwijk. (Cit. on p. 103).

Prufer, M., C. Schmidt, and F. Wahl. 1994. "Identification of robot dynamics with differential and integral models: a comparison". In *Proc. IEEE Int. Conf. Robot. Autom.* 340–345. IEEE Comput. Soc. Press. doi:10.1109/ROBOT.1994.351272. (Cit. on p. 22).

Radkhah, Katayon, Dana Kulic, and Elizabeth Croft. 2007. "Dynamic parameter identification for the CRS A460 robot". In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* 3842–3847. Ml. IEEE. doi:10.1109/IROS.2007.4399314. (Cit. on p. 23).

Santos, Luís, and Rui Cortesão. 2014. "A Dynamically Consistent Hierarchical Control Architecture for Robotic-Assisted Tele-Echography". In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* IEEE. (Cit. on p. 2).

Siciliano, B., and O. Khatib. 2008. *Springer Handbook of Robotics*, 129:2865. Springer. (Cit. on pp. 4, 18).

Siciliano, Bruno, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. 2009. *Robotics: Modelling, planning and control*. 632. Springer. (Cit. on pp. 4, 18).

Siciliano, Bruno, and Luigi Villani. 1999. *Robot Force Control*. Kluwer Academic. (Cit. on p. 4).

Sousa, Cristóvão D. 2014a. "PyLMI-SDP v1.0". ZENODO. doi:10.5281/zenodo.11795. (Cit. on pp. 67, 109).

— . 2014b. "SymPyBotics v1.0". ZENODO. doi:10.5281/zenodo.11365. (Cit. on pp. 53, 104).

Sousa, Cristóvão D., and Rui Cortesão. 2012. "SageRobotics: open source framework for symbolic computation of robot models". In *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*, 262–267. ACM Press. doi:10.1145/2245276.2245329. (Cit. on p. 104).

Sousa, Cristóvão, Rui Cortesão, and Luís Santos. 2010. "Computed torque posture control for robotic-assisted tele-echography". In *18th Mediterranean Conf. on Control and Automation*, 1561–1566. IEEE. doi:10.1109/MED.2010.5547825. (Cit. on p. 2).

Spong, Mark W., Seth Hutchinson, and M. Vidyasagar. 2005. *Robot modeling and control*. Wiley. (Cit. on p. 4).

Swevers, J., C. Ganseman, D.B. Tukel, J. de Schutter, and H. Van Brussel. 1997. "Optimal robot excitation and identification". *IEEE Trans. Robot. Autom.* 13 (5): 730–740. doi:10.1109/70.631234. (Cit. on pp. 21, 22, 57).

Swevers, Jan, Walter Verdonck, and Joris De Schutter. 2007. "Dynamic Model Identification for Industrial Robots". *IEEE Control Systems Magazine* 27, no. 5 (): 58–71. doi:10.1109/MCS.2007.904659. (Cit. on p. 18).

Taylor, Russell H. 2006. "A perspective on medical robotics". *Proceedings of the IEEE* 94, no. 9 (): 1652–1664. doi:10.1109/JPROC.2006.880669. (Cit. on p. 1).

Timcenko, A., N Kircanski, D. Urosevic, and M. Vukobratovic. 1991. "SYM-program environment for manipulator modeling, control and simulation". In *Proc. IEEE Int. Conf. Robot. Autom.* 1122–1127. doi:10.1109/ROBOT.1991.131744. (Cit. on p. 103).

Ting, Jo-Anne, Aaron D'Souza, and Stefan Schaal. 2011. "Bayesian robot system identification with input and output noise." *Neural Networks* 24, no. 1 (): 99–108. doi:10.1016/j.neunet.2010.08.011. (Cit. on pp. 22, 28, 29, 92–94, 96, 115).

Ting, Jo-anne, Michael Mistry, Jan Peters, Stefan Schaal, and Jun Nakanishi. 2006. "A bayesian approach to nonlinear parameter identification for rigid body dynamics". In *Proc. of Robotics: Science and Systems*. (Cit. on pp. 22, 23, 28).

Van der Meijden, O. A. J., and M. P. Schijven. 2009. "The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: A current review". *Surgical Endoscopy* 23, no. 6 (): 1180–90. doi:10.1007/s00464-008-0298-x. (Cit. on p. 2).

Vandanjon, P.O., Maxime Gautier, and P. Desbats. 1995. "Identification of robots inertial parameters by means of spectrum analysis". In *Proc. IEEE Int. Conf. Robot. Autom.* 3033–3038. IEEE. doi:10.1109/ROBOT.1995.525715. (Cit. on p. 21).

Vandenberghe, Lieven, and Stephen Boyd. 1996. "Semidefinite programming". *SIAM review* 38 (1): 49–95. (Cit. on p. 40).

Venture, Gentiane, Ko Ayusawa, and Yoshihiko Nakamura. 2009. "Realtime identification software for human whole-body segment parameters using motion capture and its visualization interface". In *IEEE International Conference on Rehabilitation Robotics*, 109–114. IEEE. doi:10.1109/ICORR.2009.5209622. (Cit. on p. 23).

Venture, Gentiane, and Maxime Gautier. 2012. "Dynamics calibration using inverse dynamics and LS technique — An application to the human body". In *IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, 508–513. 1. IEEE. doi:10.1109/AIM.2012.6265993. (Cit. on p. 32).

Wu, Jun, Jinsong Wang, and Zheng You. 2010. "An overview of dynamic parameter identification of robots". *Robotics and Computer-Integrated Manufacturing* 26, no. 5 (): 414–419. doi:10.1016/j.rcim.2010.03.013. (Cit. on pp. 18, 22).

Yamashita, Makoto, Katsuki Fujisawa, and Masakazu Kojima. 2003. "Implementation and evaluation of SDPA 6.0 (Semidefinite Programming Algorithm 6.0)". *Optimization Methods and Software* 18, no. 4 (): 491–505. doi:10.1080/1055678031000118482. (Cit. on p. 111).

Yoshida, Koji. 1995. "Studies on Dynamics of Robot Manipulators". PhD thesis. (Cit. on pp. 22, 27).

Yoshida, Koji, and Wisama Khalil. 2000. "Verification of the Positive Definiteness of the Inertial Matrix of Manipulators Using Base Inertial Parameters". *Int. J. Robotics Research* 19, no. 5 (): 498–510. doi:10.1177/02783640022066996. (Cit. on pp. 5, 21–24, 26, 27, 79, 91, 92).

Yoshida, Koji, K. Osuka, H. Mayeda, and T. Ono. 1994. "When is the set of base parameter values physically impossible?" In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* 1:335–342. Ieee. doi:10.1109/IROS.1994.407372. (Cit. on pp. 22–24, 27).

Yoshikawa, Tsuneo. 1990. *Foundations of Robotics: Analysis and Control*. MIT Press. (Cit. on p. 4).