Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Steuerungs- und Regelungstechnik

Learning and Feedback in Robotics with Stabilizing Controller Parameterizations

Stefan Roland Friedrich

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr.-Ing. Dr. h. c. Ralph Kennel

Prüfer der Dissertation:

1. Prof. Dr.-Ing./Univ. Tokio habil. Martin Buss

2. Prof. Dr.-Ing. Jürgen Adamy
   Technische Universität Darmstadt

Die Dissertation wurde am 08.06.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07.02.2021 angenommen.

Preface

This dissertation summarizes my research conducted during my journey as a research associate with the Chair of Automatic Control Engineering at the Technical University of Munich. This thesis would not have been possible without numerous people whom I would like to thank here.

I would like to thank my doctoral advisor, Prof. Martin Buss, for offering me the opportunity to work in academia in an exciting and challenging field. The stimulating recommendations along the scientific freedom granted to me and his trust and confidence in my abilities allowed me to become independent in doing research and to develop personally. The support for exchange with the scientific community at international flagship conferences as well as for open access publishing is highly acknowledged. I also appreciate his encouragement to pursue an academic career. I will always remember the truly unique atmosphere, inspiring environment and excellent equipment at this lab.

Moreover, I would like to thank the examiner Prof. Jürgen Adamy for his interest in my work and his constructive feedback on this thesis, as well as Prof. Ralph Kennel who headed the committee.

Furthermore, I am grateful to Prof. Dieter Fox for some inspiring discussions at the very beginning of my doctoral studies. I would also like to thank Prof. Sandra Hirche for inviting excellent scholars to ITR who were a source of inspiration to me. I am also much obliged to Dr. Marion Leibold for some in-depth proofreading, her constructive feedback, and always having an open ear. I also appreciated teaching the class on optimization together with her.

Many sincere thanks to my roommates Tim Brüdigam, Sheraz Khan and Yongxu He for the good memories we share of our office companionship and the fun moments. Likewise, I thank my colleagues and especially the SHRINE team for an open and supportive attitude, the regular discussions and the mutual exchange: Philine Meister, Markus Schill, Stefan Kersting, Sotiris Apostolopoulos, Alexander Pekarovskiy, Michaela Semmler, Fangzhou Liu, and Rameez Hayat. Special thanks to Rameez, Sotiris and Markus for providing robot models. I owe much to Philine and Andreas Lawitzky for encouraging me towards a paper when it was about time. Also, from the bottom of my heart, let me say thank you for the music to Christian Landsiedel. Some very special thanks is dedicated to those hard-working admins who have been managing over the years to provide a carefree system to others at the lab, with Volker Gabler particularly standing out. Some more extra credit goes to Volker for hints on running the LWR, and to Gerold Huber who supported me not only with insightful explanations concerning peculiarities in manipulator control. I am also thankful for kind and professional support in organizational and technical matters by Larissa Schmid, Karin Rosenits, Miruna Werkmeister, Wolfgang Jaschik, Tobias Stoeber, Brigitta Renner, and all other non-scientific staff.

During my time at LSR, I had the chance to work with some bright and talented students. I would especially like to thank Michael Schreibauer and Manuel Rösler for their commitment and excellent contributions.

Finally, my friends and family deserve my deepest gratefulness: I would like to thank my parents Elke and Werner for their continuous support and invaluable advice in hard times, additionally thanks also to Roland, Sonja, and Kerstin. Also I am fortunate to have friends who have been so forgiving. Most importantly, my beloved wife Anja backed my endeavors, although countless evenings and weekends were sacrificed throughout the years. Thank you.

Bayreuth, May 2021 Stefan Friedrich

Acknowledgment

The research leading to these results has received funding partly from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. [267877] and the Technical University of Munich – Institute for Advanced Study (www.tum-ias.de), funded by the German Excellence Initiative.

Abstract

The demand for increasingly autonomous robots is growing in both industrial and domestic applications. Intelligent control technologies are required that enable robots to acquire new skills, safely learn how to execute a task, or improve in order to deliver the desired performance. Correspondingly, the field draws attention from researchers in the machine learning, feedback control theory, and robotics communities. This thesis combines these perspectives and is motivated by the question how machine learning may be employed for performance optimization of a feedback control system, such that stability is always preserved despite learning in the closed loop. Aiming to find solutions to this challenging question, the thesis builds a bridge from the parameterization of stabilizing controllers to reinforcement learning methods, with a particular focus on robotics. Novel learning control architectures, algorithms, and insights are presented resulting from the combination of these distinct approaches.

First, the parameterization of stabilizing controllers is investigated with attention to adaptive schemes and the robotic manipulator control domain. Opposed to data-driven machine learning approaches that typically work without modeling of the physical plant, the parameterization is model based. Prior dynamical models of process and control are not ignored but serve to structure the solution space to account for the effects of feedback. Motivated by the fact that certain robotic applications demand for switching or interpolation between state feedback, a simple architecture is derived for controller interpolation with a special focus on applicability to robotic hardware. Subsequently, a novel robot manipulator control framework is developed to assure stability to the loop in spite of online modifications of the feedback controller. The new parameterization for robust control of rigid manipulators leverages a dual parameterization as the key tool to quantify uncertainty in the control loop. Hence, the admissible search space can be systematically tightened to contain only robustly stabilizing controllers for robot manipulators. The proportional-derivative feedback controller commonly used among practitioners can serve as a starting point to implement the framework, allowing for a wide range of approximate inverse dynamics control configurations, and a two-degree-of-freedom controller design.

Next, intelligent control methods are presented aiming towards the important goal of powerful online learning robot control. Two novel online least-squares policy iteration algorithms are developed with both robotics as well as the deployment to the parameterization approach in mind. In particular, an online algorithm with a polynomial basis for continuous action representation is endowed with a kernel-inspired automatic feature selection method of low computational complexity. In the subsequent analysis, the general interplay between the model-free reinforcement learning methods and the model-based controller parameterization is investigated. From the combination, five control architectures are systematically revealed and the construction of the learning control framework is summarized in an overall workflow. It is shown that from the reinforcement learning perspective, the model-based parameterization can be seen as a structured way to create specific policies based on prior dynamic model domain information.

Finally, the thesis presents two laboratory case studies to illustrate the proposed methods by means of practical examples on robotic hardware. In the first experiment, active variable impedance control based on the parameterization of arbitrarily interpolated state feedback is deployed to an industrial robotic reference platform. The experiment verifies that the nominal gains constitute a degree of freedom in the design to decrease uncertainty and construct a suitable controller parameterization although the known dynamic model is imprecise. This way, instabilities due to hidden coupling can be effectively avoided to reduce the constraints that otherwise need to be imposed when learning stiffness schedules. The experimental results of the second study confirm that the aim of learning performance enhancement directly on hardware is feasible in the presented stability-by-design framework.

The thesis concludes with a summary of the proposed control strategy to exploit domain knowledge for learning and feedback in robotics based on stabilizing controller parameterizations. Potential directions for future research based on the parameterization approach are pointed out in the interdisciplinary fields of stable machine learning control as well as robotic learning control.

Kurzfassung

Der Bedarf an weiter autonom werdenden Robotern wächst sowohl für industrielle als auch private Anwendungen. Technologien intelligenter Regelungsverfahren sind erforderlich, die es Robotern ermöglichen, neue Fertigkeiten zu erwerben, sicher zu lernen wie eine Aufgabe auszuführen ist oder sich selbst zu verbessern, um das gewünschte Verhalten zu erreichen. Dieses Forschungsgebiet zieht daher das Interesse von Forschern aus dem Bereich des maschinellen Lernens, der Regelungstechnik und der Robotik auf sich. Die vorliegende Arbeit kombiniert diese Sichtweisen und ist durch die Frage motiviert, wie maschinelles Lernen zur Leistungsverbesserung eines Regelungssystems eingesetzt werden kann, so dass die Stabilität im geschlossenen Kreis trotz des Lernens in der Rückkopplung erhalten bleibt. Um Lösungen für diese herausfordernde Fragestellung zu finden, schlägt diese Arbeit eine Brücke zwischen der Parametrierung aller stabilisierender Regler und der Methode des Bestärkenden Lernens, mit einem besonderem Augenmerk auf Robotik. Neuartige Architekturen lernender Regelungen, Algorithmen und Erkenntnisse werden vorgestellt, die sich aus der Kombination dieser unterschiedlichen Herangehensweisen ergeben.

Zunächst wird die Parametrierung stabilisierender Regler hinsichtlich adaptiver Schemen und dem Hintergrund der Manipulatorregelung untersucht. Im Gegensatz zu Ansätzen des maschinellen Lernens, die typischerweise rein datengetrieben und ohne die Modellierung des physikalischen Prozesses verwendet werden, ist die Parametrierung modellbasiert. Bereits existierende dynamische Modelle der Strecke und eines Reglers werden daher nicht verworfen, sondern dienen der Strukturierung des Lösungsraums, so dass die Auswirkungen der Rückkopplung berücksichtigt werden. Motiviert durch die Tatsache, dass bestimmte Robotikanwendungen das Umschalten oder Interpolieren zwischen verschiedenen Zustandsrückführungen erforderlich machen, wird mit Hinblick auf die Anwendbarkeit im realen System eine relativ einfache Architektur für die Reglerinterpolation entworfen. Anschließend wird ein neuartiges Konzept zur Regelung von Robotern entwickelt, das die Stabilität der Regelschleife trotz der Online-Modifikation des Reglers gewährleistet. Diese neuartige Parametrierung zur robusten Regelung von als Starrkörpern modellierten Manipulatoren nutzt eine duale Parametrierung als zentrales Element zur Quantifizierung der Unsicherheit im Regelkreis. Der zulässige Suchraum kann somit systematisch auf robust stabilisierende Regler für Robotermanipulatoren eingeschränkt werden. Als Ausgangspunkt zur Implementierung des Frameworks kann die in der robotischen Praxis sehr weit verbreitete PD-Rückführung verwendet werden. Im Design ist ferner eine generelle approximationsbasierte Inversdynamik- und Zwei-Freiheitsgrade-Regelungsstruktur berücksichtigt.

Als nächstes werden intelligente Methoden vorgestellt, die zum wichtigen Ziel performanter online lernender Regelungen für Roboter beitragen. Es werden zwei neuartige Online Least-Squares Policy Iteration Algorithmen entwickelt, sowohl im Hinblick auf den Einsatz in der Robotik als auch dem Ansatz der obigen Reglerparametrierung. Insbesondere wird einer der Algorithmen mit einer Polynomialbasis zur Approximation des kontinuierlichen Eingangsraums ausgestattet, die aufgrund der geringen Rechenkomplexität onlinefähig ist und eine Kernel-inspirierte automatische Wahl der Merkmale ermöglicht. In der nachfolgenden Analyse wird das allgemeine Zusammenspiel zwischen dem modellfreien Bestärkenden Lernen und der modellbasierten Reglerparametrierung untersucht. Die Kombination zeigt fünf systematische Steuerungsarchitekturen auf und führt zu einem Workflow, der die Benutzung der gesamten lernenden Regelungsstrategie zusammenfasst. Es wird gezeigt, dass die modellbasierte Parametrierung aus Sicht des bestärkenden Lernens als ein strukturierter Weg verstanden werden kann, den Suchraum über Regelungsstrategien spezifisch derart zu gestalten, dass vorab vorhandenes Domänenwissen eingebracht wird.

Schließlich werden zwei Fallstudien vorgestellt, um die vorgeschlagenen Methoden anhand von Laborexperimenten mit praktischen Beispielen auf Roboterhardware zu veranschaulichen. Im ersten Experiment wird die aktive variable Impedanzregelung mittels der Parameterierung beliebig interpolierter Zustandsrückführungen auf einer industriellen Roboterreferenzplattform umgesetzt. Das Experiment bestätigt, dass trotz des nur ungenau angenommenen Dynamikmodells eine geeignete Parametrierung konstruiert werden kann, indem die nominellen Verstärkungsfaktoren als Entwurfsfreiheitsgrad zur Reduktion der Unsicherheit verwendet werden. Auf diese Weise kann Instabilität effektiv vermieden werden, die ansonsten aufgrund der verborgenen Kopplung enstehen kann und somit Einschränkungen beim Lernen der benötigten Steifigkeitsprofile notwendig macht. Die experimentellen Ergebnisse der zweiten Fallstudie belegen, dass es im vorgestellten, insgesamt primär auf Stabilität ausgelegten Regelungsansatz möglich ist, Lernalgorithmen zur Verbesserung der Performanz direkt auf der Hardware einzusetzen.

Die Arbeit schließt mit einer Zusammenfassung der vorgeschlagenen Strategie, Domänenwissen mittels der Parametrierung stabilisierender Regler für das maschinelle Lernen und die Regelung in der Robotik zu verwenden. Abschließend werden mögliche künftige Forschungsrichtungen basierend auf der Parametrierung für die interdisziplinären Forschungsfelder der stabilitätserhaltenden Regelung mit maschinellem Lernen und der lernenden Roboterregelung aufgezeigt.

Contents

Notation

Acronyms and Abbreviations

ADP	Approximate (Adaptive) Dynamic Programming
AI	Artificial Intelligence
AID	Approximate Inverse Dynamics
ALD	Approximate Linear Dependency
AOLSPI	Automated Online Least-Squares Policy Iteration
BBO	Black-Box Optimization
BF	Basis Function
BIBO	Bounded-Input Bounded-Output
CS	Coherence Sparsification
CQLF	Common Quadratic Lyapunov Function
DARE	Discrete-Time Algebraic Riccati Equation
DMP	Dynamic Movement Primitive
DoF	Degree Of Freedom
DP	Dynamic Programming
FMP	Feedback Motion Planning
FIR	Finite Impulse Response
FRI	Fast Research Interface
GP	Gaussian Process
GMM	Gaussian Mixture Model
IQC	Integral Quadratic Constraints
KDHP	Kernel-Based Dual Heuristic Programming
KDPP	Kernel Dynamic Policy Programming
KLSPE	Kernel Least-Squares Policy Evaluation
KLSPI	Kernel-Based Least-Squares Policy Iteration
KLSTD	Kernel-Based Least-Squares Temporal Difference
LFT	Linear Fractional Transformation
LMI	Linear Matrix Inequality
LPV	Linear Parameter-Varying
LQN	Local Q-Network
LQR	Linear Quadratic Regulator
LSPI	Least-Squares Policy Iteration

LTI	Linear Time-Invariant
LTV	Linear Time-Varying
LWR	Lightweight Robot
MDP	Markov Decision Process
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MSDP	Multi-Stage Decision Process
NN	(Artificial) Neural Network
OKLSPI	Online Kernel Least-Squares Policy Iteration
OLSPI	Online Least-Squares Policy Iteration
PbD	Programming By Demonstration
PD	Proportional-Derivative
PI	Policy Iteration
RBF	Radial Basis Function
RMS	Root Mean Square
RKHS	Reproducing Kernel Hilbert Space
RL	Reinforcement Learning
TD	Temporal Difference
VFA	Value Function Approximation

Mathematical Conventions

For the sake of readability, explicit function dependencies are generally dropped whenever unambiguously clear from the context. The imaginary unit is denoted j, *i. e.*, $\mathrm{j}^2 = -1$.

Scalars, Vectors, and Matrices

Scalars are written in lower case standard letters (a), lowercase boldface letters are for vectors ($\boldsymbol{v}$), matrices are written in upper case bold letters ($\boldsymbol{M}$), sets are written in calligraphic ($\mathcal{S}$), and operators defining (system) input/output relations are in standard capital letters (G). The i^{th} element of a vector $\boldsymbol{v}$ is denoted v_i and the elements of a matrix $\boldsymbol{M}$ are referred to as m_{ij} (i^{th} row, j^{th} column). The zero matrix $\mathbf{0}$, the all-one matrix $\mathbf{1}$ and the identity matrix $\boldsymbol{I}$ are taken of appropriate dimensions given from the context. The dimension may be stated explicitly by subscripts, *e. g.*, $\mathbf{0}_{m\times n} \in \mathbb{R}^{m\times n}$ denotes a zero matrix with m rows and n columns and $\mathbf{0}_n \triangleq \mathbf{0}_{n\times n}$.

General Sets

$\emptyset$	empty set
$\mathbb{N}$	set of all natural numbers
$\mathbb{Z}_0^+$	set of all non-negative integer numbers
$\mathbb{R}_0^+$	set of all non-negative real numbers
$\mathbb{R}^n$	n-dimensional Euclidian space, with $n = 1$ if n is omitted

$\mathbb{R}^{m\times n}$	set of real-valued $m \times n$ matrices
$\mathcal{B}_p^{m\times n}(\gamma)$	set of γ-bounded $m \times n$ real matrices w. r. t. the induced p-norm, *i. e.*, $\mathcal{B}_p^{m\times n}(\gamma) \triangleq \{\boldsymbol{M} : \boldsymbol{M} \in \mathbb{R}^{m\times n}, 0 \leq \|\boldsymbol{M}\|_p \leq \gamma\}$
$\mathcal{B}_{\max}^{m\times n}(b)$	set of real-valued $m\times n$ matrices with elementwise b-bounded entries
$\mathcal{C}^p$	set of p times continuously differentiable functions
ℓ_∞	set of all bounded sequences $\boldsymbol{x}(k)$ over $\mathbb{Z}_0^+$, satisfying $\|\boldsymbol{x}(t)\|_{\ell_\infty} < \infty$
$\mathcal{L}_\infty$	set of all bounded signals $\boldsymbol{x}(t)$ over $\mathbb{R}_0^+$, satisfying $\|\boldsymbol{x}(t)\|_{\mathcal{L}_\infty} < \infty$
$\mathcal{L}_2$	set of all finite-energy signals $\boldsymbol{x}(t)$ over $\mathbb{R}_0^+$, satisfying $\|\boldsymbol{x}(k)\|_{\mathcal{L}_2} < \infty$
$\mathcal{RH}_\infty$	set of all proper and real rational stable transfer matrices
$\mathcal{RH}_\infty^{n,m}$	set of all proper and real rational stable $n \times m$ transfer matrices
$\mathcal{RH}_2$	set of all strictly proper and real rational stable transfer matrices

Operators

iff	if and only if
$\triangleq$	equality by definition
$\neq$	inequality
$\otimes$	Kronecker product
$\mathrm{col}(\boldsymbol{x}_1, \boldsymbol{x}_2)$	vertical concatenation of two column vectors $\boldsymbol{x}_1, \boldsymbol{x}_2$, *i. e.*, $\left[\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top\right]^\top$
$\boldsymbol{M}^\top$	transpose of a matrix $\boldsymbol{M}$
$\boldsymbol{M}^{-1}$	inverse of a square matrix $\boldsymbol{M}$
$\boldsymbol{M} \succ (\succeq)\ 0$	a symmetric matrix $\boldsymbol{M}$ is positive (semi-)definite
$\boldsymbol{M} \prec (\preceq)\ 0$	a symmetric matrix $\boldsymbol{M}$ is negative (semi-)definite
$\lambda(\boldsymbol{M})$	an eigenvalue of a matrix $\boldsymbol{M}$
$\mathrm{sp}(\boldsymbol{M})$	the spectrum of a matrix $\boldsymbol{M}$, *i. e.*, the eigenvalues of $\boldsymbol{M}$
$\rho(\boldsymbol{M})$	the spectral radius of a matrix $\boldsymbol{M}$, *i. e.*, $\rho(\boldsymbol{M}) = \max\{\lvert\lambda\rvert : \lambda \in \mathrm{sp}(\boldsymbol{M})\}$
$\sigma_{\max}(\boldsymbol{M})$	the largest singular value of a matrix $\boldsymbol{M}$
$\lvert\cdot\rvert$	absolute value, for sets the cardinality
$\lceil\cdot\rceil$	round a number up to the next largest integer, *i. e.*, $\lceil x\rceil = \min\{n \in \mathbb{Z} : n \geq x\}$
$\|\boldsymbol{v}\|_p$	vector p-norm ($1 \leq p < \infty$) of $\boldsymbol{v} \in \mathbb{R}^n$, $\|\boldsymbol{v}\|_p \triangleq (\sum_{i=1}^n \lvert v_i\rvert^p)^{1/p}$
$\|\boldsymbol{v}\| \triangleq \|\boldsymbol{v}\|_2$	Euclidian vector norm of $\boldsymbol{v} \in \mathbb{R}^n$, $\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v}^\top\boldsymbol{v}}$
$\|\boldsymbol{v}\|_\infty$	maximum norm of $\boldsymbol{v} \in \mathbb{R}^n$, $\|\boldsymbol{v}\|_\infty = \max_{1\leq i\leq n}\lvert v_i\rvert$
$\|\boldsymbol{M}\|_p$	induced matrix norm, $\|\boldsymbol{M}\|_p = \sup_{\boldsymbol{v}\neq 0} \frac{\|\boldsymbol{M}\boldsymbol{v}\|_p}{\|\boldsymbol{v}\|_p}, \quad 1 \leq p \leq \infty$
$\|\boldsymbol{M}\| \triangleq \|\boldsymbol{M}\|_2$	spectral norm of a matrix $\boldsymbol{M}$, $\|\boldsymbol{M}\|_2 = \sigma_{\max}(\boldsymbol{M})$
$\mathrm{diag}(\boldsymbol{v})$	the diagonal $n \times n$ matrix where the $i^{\mathrm{th}}(i = 1, 2, \dots n)$ entry on the diagonal is equal to the i^{th} entry of the vector $\boldsymbol{v} \in \mathbb{R}^n$
$\mathrm{blkdiag}(\boldsymbol{M}_1, \dots, \boldsymbol{M}_n)$	the block-diagonal matrix where the $i^{\mathrm{th}}(i = 1, 2, \dots n)$ matrix block on the diagonal is given by $\boldsymbol{M}_i$
$\mathrm{conv}(\boldsymbol{M}_1, \dots, \boldsymbol{M}_n)$	set of convex combinations of vertex matrices $\boldsymbol{M}_i$
$\limsup$	limes superior
$\|\boldsymbol{x}\|_{\mathcal{L}_p}$	the $\mathcal{L}_p$ signal norm of a Lebesgue measurable function $\boldsymbol{x}(t)$ defined on $\mathbb{R}_0^+$, $\|\boldsymbol{x}\|_{\mathcal{L}_p} \triangleq (\int_0^\infty \|\boldsymbol{x}(t)\|^p \,\mathrm{d}t)^{1/p}$, for $1 \leq p < \infty$

$\|\boldsymbol{x}\|_{\ell_p}$	the ℓ_p signal norm of a sequence $\boldsymbol{x}(k)$ defined on $\mathbb{Z}_0^+$, $\|\boldsymbol{x}\|_{\ell_p} \triangleq (\sum_{i=0}^{\infty} \|\boldsymbol{x}(i)\|^p)^{1/p}$, for $1 \leq p < \infty$
$\|\boldsymbol{x}\|_{\mathcal{L}_\infty}$	the $\mathcal{L}_\infty$ signal norm of a Lebesgue measurable function $\boldsymbol{x}(t)$ defined on $\mathbb{R}_0^+$, $\|\boldsymbol{x}\|_{\mathcal{L}_\infty} \triangleq \operatorname{ess\,sup}_{0 \leq t \leq \infty} \|\boldsymbol{x}(t)\|$
$\|\boldsymbol{x}\|_{\ell_\infty}$	the ℓ_∞ signal norm of a sequence $\boldsymbol{x}(k)$ defined on $\mathbb{Z}_0^+$, $\|\boldsymbol{x}\|_{\ell_\infty} \triangleq \sup_{i=1,2,\dots} \|\boldsymbol{x}(i)\|$
$\|G\|_\infty$	for a linear system, the $\mathcal{H}_\infty$ norm; if G is a continuous (discrete-time) nonlinear mapping, the induced $\mathcal{L}_2$ (ℓ_2) gain of G is meant
δ	represents the derivative operator $\delta \boldsymbol{x}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t)$ in continuous-time and the one-step shift operator $\delta \boldsymbol{x}(t) = \boldsymbol{x}(t+1)$ in discrete-time
$G_1 \sim G_2$	input/output equivalence of systems G_1 and G_2
$\left[\begin{array}{c\|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{C} & \boldsymbol{D} \end{array}\right]$	state-space realization of a discrete- or continuous-time system
(G_1, G_2)	the feedback interconnection of two systems G_1 and G_2 through common input/output variables
$\mathcal{F}_\ell(G, K)$	the lower linear fractional transformation of a partitioned operator $G = \left[\begin{smallmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{smallmatrix}\right]$ with another mapping K of appropriate dimensions, $\mathcal{F}_\ell(G, K) \triangleq G_{11} + G_{12} K (I - G_{22} K)^{-1} G_{21}$
$\mathcal{F}_u(G, \Delta)$	the upper linear fractional transformation of a partitioned operator $G = \left[\begin{smallmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{smallmatrix}\right]$ with another mapping Δ of appropriate dimensions, $\mathcal{F}_u(G, \Delta) \triangleq G_{22} + G_{21} \Delta (I - G_{11} \Delta)^{-1} G_{12}$

Super-/Subscripts

Superscripts

$\tilde{(\cdot)}$	a factor related to a left coprime factorization
$(\cdot)^\star$	optimal value
$\undertilde{(\cdot)}$	error quantity
$\hat{(\cdot)}$	estimated or imprecisely known quantity

Subscripts

$(\cdot)_\mathrm{K}$	variable is related to a controller
$(\cdot)_\mathrm{Q}$	variable is related to a Q-parameter
$(\cdot)_\mathrm{fb}$	quantity is related to feedback control (exclusively)
$(\cdot)_\mathrm{ff}$	quantity is related to feedforward control action
$(\cdot)_\mathrm{cl}$	quantity is related to closed loop
$(\cdot)_\mathrm{nom}$	nominal value
$(\cdot)_0$	a nominal system, usually plant models or initial controllers
$(\cdot)_\mathrm{des}$ or $(\cdot)_\mathrm{d}$	desired value
$(\cdot)_\mathrm{ref}$ or $(\cdot)_\mathrm{r}$	some reference value
$(\cdot)_\mathrm{dist}$	a quantity acting as disturbance

$(\cdot)_{\mathrm{aug}}$	augmented model
$(\cdot)_{\mathrm{noise}}$	related to noise
$(\cdot)_{\mathrm{cmd}}$	commanded value
$(\cdot)_{\mathrm{msr}}$	measured (on hardware)
$(\cdot)_{\mathrm{sat}}$	saturated (limited) value
$(\cdot)_{\mathrm{max}}$	maximum value of some quantity
$(\cdot)_{\mathrm{LQN}}$	associated with a local Q-network implementation
$(\cdot)_{\mathrm{LPV}}$	associated with a LPV realization
$(\cdot)_{\mathrm{C}}$	quantity in Cartesian coordinates
$(\cdot)_{\mathrm{j}}$	quantity in joint coordinates

Symbols and Variables

Dynamic Systems

t	time instant in continuous or discrete time domain
T_{s}	sampling time used for discretization
$\mathcal{T}$	set representing time, usually in continuous time $\mathcal{T} = \mathbb{R}_0^+$ and in discrete time $\mathcal{T} = \mathbb{Z}_0^+$
$\boldsymbol{f}$	nonlinear vector function to describe system dynamics
$\boldsymbol{g}$	nonlinear vector function to represent general output equation
$\boldsymbol{x}$	state of a system with n dimensions
$\boldsymbol{x}_0$	initial state
$\boldsymbol{u}$	control inputs with n_{u} dimensions
$\boldsymbol{y}$	measured signals with n_{y} dimensions
$\boldsymbol{w}$	exogenous input with n_{u} dimensions
$\boldsymbol{z}$	performance quantities with n_{z} dimensions
$\boldsymbol{w}_\Delta$	input perturbation resulting from uncertainty in generalized plant
$\boldsymbol{z}_\Delta$	vector exciting uncertainty in generalized plant
$\boldsymbol{A}$	state matrix of a (linear) system
$\boldsymbol{B}$	input matrix of a system
$\boldsymbol{C}$	(measured) output matrix
$\boldsymbol{D}$	feedthrough matrix
$\boldsymbol{P}$	matrix defining a (quadratic) Lyapunov function
$\begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$	partitioned operator of appropriate size mapping two input to two output signals

Controller Parameterization

G	generalized plant
G_0	nominal plant for controller/parameterization design
$\mathcal{G}$	a set of plants $G \in \mathcal{G}$
K_0	nominal (feedback) controller

N	factor related to right coprime factorization of a plant
M	factor related to right coprime factorization of a plant
$\tilde{N}$	factor related to left coprime factorization of a plant
$\tilde{M}$	factor related to left coprime factorization of a plant
V	factor related to right coprime factorization of a controller
U	factor related to right coprime factorization of a controller
$\tilde{V}$	factor related to left coprime factorization of a controller
$\tilde{U}$	factor related to left coprime factorization of a controller
Q	the Q-parameter, also dubbed ("plug-in") Q-filter
$\mathcal{K}$	set of controllers in standard control loop
$\mathcal{Q}$	set of controllers defined by their parameters Q in a parameterization
J	central system of a parameterization; also called generator system
$\boldsymbol{r}$	n_{y}-dimensional vector of (residual) signals serving as input to the Q-parameter, generated from J
$\boldsymbol{s}$	n_{u}-dimensional output vector of the Q-filter
T	pre-stabilized closed-loop system resulting from interconnection of G and J
J_G	central system of a dual Youla parameterization w. r. t. nominal plant G
S	dual Youla parameter
$\boldsymbol{x}_{\mathrm{J}}$	state of generator system J
$\boldsymbol{F}$	state feedback gain
$\boldsymbol{F}_{\mathrm{G}}$	state feedback design matrix stabilizing system G
$\boldsymbol{F}_{\mathrm{K}}$	state feedback design matrix stabilizing controller K
$\boldsymbol{L}$	observer gain
$\boldsymbol{D}_{\mathrm{K},0}$	a nominal static feedback controller
γ_{Q}	upper bound of $\mathcal{H}_\infty$-norm of parameters Q
γ_{S}	upper bound of $\mathcal{H}_\infty$-norm of dual parameters S
Δ_N	coprime factor uncertainty related to N
Δ_M	coprime factor uncertainty related to M
$\mathcal{K}_{\mathrm{R}}$	set of robust controllers in standard control loop
Δ	operator representing uncertainty in a generalized plant formulation

Switched, Interpolated and Adaptive Systems

$\boldsymbol{\theta}$	parameter vector
$\mathcal{P}$	set of admissible scheduling parameters
$\boldsymbol{\alpha}$	scheduling/interpolation vector
$\mathcal{A}$	set of piecewise continuous admissible arbitrary interpolation vectors
N_{sys}	number of switched or interpolated systems
$\mathcal{I}$	set of indices of systems
σ	switching signal determining the active mode at each time instant
N_{K}	number of switched or interpolated controllers
$\mathcal{J}$	set of several central systems $J \in \mathcal{J}$
$\boldsymbol{\Xi}$	matrix of adaptively varied parameters
$\boldsymbol{D}_{\mathrm{K},i}$	ith static feedback controller
$\boldsymbol{F}_{\mathrm{N}}$	static feedback gain robust w. r. t. normalized coprime factor uncertainty

T_r	dynamic stable reference model with state $\boldsymbol{x}_\mathrm{r}$
$\bar{\boldsymbol{D}}_\mathrm{K}$	control gain obtained from averaging other gains
C_z	cost index related to variables $\boldsymbol{z}$
$\boldsymbol{R}_\mathrm{z}$	weighting matrix for cost of the variables $\boldsymbol{z}$
ϑ_i	adaptation factor for ith submodel
μ_i	update rate for ith submodel
$\boldsymbol{\Gamma}$	state variable of gradient dynamics
γ_i	ith gradient vector
t_s	an instant in time when switching takes place
t_th	a (threshold) number of time steps
t_N	a finite number of timesteps

Robotics

n	number of generalized coordinates of a robotic manipulator
$\boldsymbol{q}$	vector of the generalized coordinates of a robotic system
$\boldsymbol{x}$	state vector of a robotic manipulator, $\boldsymbol{x} \triangleq \mathrm{col}(\boldsymbol{q}, \dot{\boldsymbol{q}})$
$\boldsymbol{f}_\mathrm{e}$	external wrench applied to end effector
$\boldsymbol{M}$	generalized mass matrix
$\boldsymbol{K}_\mathrm{P}$	a stiffness matrix or a related feedback gain
$\boldsymbol{K}_\mathrm{D}$	a damping matrix or a related feedback gain
$\boldsymbol{J}$	Jacobian matrix
$\boldsymbol{x}_\mathrm{C}$	robot position in Cartesian coordinates
$\boldsymbol{f}_\tau$	input vector in Cartesian coordinates
$\boldsymbol{\tau}$	torque input vector (joint coordinates)
$\boldsymbol{n}$	vector that summarizes the nonlinear terms
$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$	vector of Coriolis and centripetal forces
$\boldsymbol{f}(\dot{\boldsymbol{q}})$	vector of friction terms
$\boldsymbol{g}(\boldsymbol{q})$	vector of gravitational forces
$\boldsymbol{e}$	vector of position and velocity tracking errors
$\tilde{\boldsymbol{n}}_\mathrm{C}$	error of Coriolis/centripetal vector
$\tilde{\boldsymbol{n}}_\mathrm{f}$	error of friction vector
$\tilde{\boldsymbol{n}}_\mathrm{g}$	error of gravity vector
$\boldsymbol{\Delta}_\mathrm{M}$	uncertainty matrix due to inertia modeling error
$\boldsymbol{\psi}$	inverse additive disturbances due to non-inertia modeling error
$M_\mathrm{l}, M_\mathrm{u}$	lower and upper bound related to norm of inverse inertia
C_u	upper bound related to Coriolis matrix
F_u	upper bound related to friction
g_u	upper bound related to gravity vector
q_max	manipulator workspace bound
v_max	manipulator velocity bound
C_dist	upper bound related to disturbance input
W	upper bound related to measurement noise
Q_d	upper bound on reference trajectory
$\alpha,\ \alpha_\mathrm{M}$	upper bound related to uncertainty caused by inertia modeling error

α_{Ψ}	upper bound related to uncertainty caused by nonlinear signal uncertainty
Φ	bounding function concerning manipulator nonlinearity vector
$\boldsymbol{\nu}$	additional input term related to robust manipulator control
$\boldsymbol{\eta}$	vector nonlinear (uncertainty) term ignoring structure of uncertainty
$\boldsymbol{w}_{\Delta}^{\mathrm{U}}$	vector of internal model uncertainty
$\boldsymbol{w}_{\Delta}^{\mathrm{M}}$	vector of inertia-induced uncertainty
$\boldsymbol{w}_{\Delta}^{\psi}$	vector of signal uncertainty
$\mathcal{D}_{\Delta}$	set of uncertainty perturbation operators
$\boldsymbol{\Delta}_{\psi}$	uncertain matrix related to nonlinearities uncertainty
$\mathcal{S}_{\Delta}$	set of uncertain dual Youla operators
n_{S}	number of samples
$\boldsymbol{\delta u}_{\mathrm{a}}$	additional input from state of the parameterization central system
$\boldsymbol{\delta u}$	input that is additive to the nominal input
m_{p}	mass of payload
$\boldsymbol{k}_{\mathrm{j}}$	stiffness vector exposed by the FRI
$\boldsymbol{d}_{\mathrm{j}}$	damping parameter exposed by the FRI
$\boldsymbol{\tau}_{\mathrm{FRI}}$	superimposed torque accessible via the FRI
$\overline{\boldsymbol{m}}$	mean values of the moments of inertia
$\Delta\boldsymbol{M}$	coordinate-dependent part of inertia matrix

Reinforcement Learning

$\mathscr{S}$	state space
$\mathscr{A}$	action space
$\mathscr{S}\mathscr{A}$	state-action space, $\mathscr{S}\mathscr{A} \triangleq \mathscr{S} \times \mathscr{A}$
P_s^a	state transition function
r	reward function
R_t	reward value at time t
$\mathbf{s}_t$	a state in $\mathscr{S}$ at time t
a_t	an action in $\mathscr{A}$ at time t
γ	discount factor, $0 \leq \gamma \leq 1$
G_t	return, the future accumulated discounted reward
π	a policy determining the actions of an agent
$\mathcal{V}_{\pi}$	state value function
$\mathcal{Q}_{\pi}$	state-action value function
$\mathcal{F}$	feature set
$\boldsymbol{\phi}(\cdot,\cdot)$	vector of basis functions $\phi(\cdot,\cdot)$
σ	(scalar) variance of a radial basis function
Σ	(matrix) variance of radial basis functions
N_{ϕ}	number of basis functions
$\boldsymbol{\theta}$	parameter vector in a function approximation
$\boldsymbol{\Lambda}, \boldsymbol{\lambda}$	matrix and vector in temporal difference learning
Ψ_j	Chebyshev polynomials of the first kind of degree j
κ	(Mercer) kernel function
$\boldsymbol{\varPhi}$	feature map defined by kernel

$\mathcal{H}$	high-dimensional feature space
$\mathcal{D}$	dictionary of previous data samples
N_{K}	number of elements in dictionary, $N_{\mathrm{K}} \triangleq \lvert\mathcal{D}\rvert$
δ	approximation error related to approximate linear dependency
μ	coherence of a dictionary
μ_0	a threshold concerning the coherence of a dictionary
ε	exploration factor
Λ	an entry in the matrix $\mathbf{\Lambda}$
λ	an entry in the vector $\boldsymbol{\lambda}$
K_θ	policy improvement interval
p_{e}	trust radius in basis function weighting
t_{exec}	execution time for one trial
t_{total}	execution time of all trials
N_{eval}	number of independent evaluation runs
N_{test}	number of test runs
$\boldsymbol{\nu}$	some signal to query relevant information, e. g., positions
$\mathcal{V}$	query space, set of possible values for $\boldsymbol{\nu}$
$\boldsymbol{\gamma}_i$	grid point, center of ith radial basis function (RBF)
n_γ	number of grid points
$\boldsymbol{\xi}$	vector of parameter functions determining coefficients in FIR-like, block-diagonal filter architecture
n_{p}	number of parameter functions
$\Sigma_{\mathrm{fb}}, \Sigma_{\mathrm{ff}}$	(initial) covariances used in black-box optimization algorithm

1 Introduction

Agile and highly flexible, customized production systems are a key technological trend in the ongoing development of Industry 4.0 [83]. In the field of robotics, this trend is reflected by a rapidly increasing interest in the branch of collaborative robotics, *i. e.*, the development of machines that are no longer restricted to cages in the industrial production hall but perform tasks simultaneously with a human co-worker. While more and more commercial applications are available on an increasing market [244], many scientific and technological barriers must still be overcome in order to realize the vision of intelligent and autonomous robots, seamlessly collaborating and sharing the workspace with humans.

Working towards the envisioned level of autonomy, robots are being equipped with artificial intelligence (AI) capabilities. A lot of current robotics research therefore revolves around how to employ machine learning (ML) methods to enable robots to learn tasks, instead of having to perform explicit, inflexible, and expensive expert programming. In this learning context, robot programming by demonstration (PbD) [25, 37] aims to teach skills to robots by guiding the robot kinesthetically and deriving a mathematical representation of the demonstrated task; this approach is also called imitation learning or learning from demonstration. Alternatively, robots may be equipped with mechanisms to learn a task on their own [70, 115] by reinforcement learning (RL) [228]. To this end, the robotic system attempts to perform the desired task in trial-and-error fashion, aiming to maximize a reward quantity that assesses its performance.

Control theory and AI—synergetic perspectives on decision making. The aforementioned trial-and-error paradigm of reinforcement learning has its roots in artificial intelligence and is one approach to solving the sequential decision making problem of how to choose actions given the state, such that the long-term performance is optimized. In other words, the policy governing the behavior of the agent constitutes a state feedback control law. The control community in turn has developed extensive knowledge not only about decision making under uncertainty, but also about modeling of dynamic systems, feedback control, stability, robustness, and so on. These properties, however, are not in the focus of the computational intelligence driven research on reinforcement learning which allows for model-free, data-driven, and broad applicability to unknown environments while aiming for high convergence rates of the actor towards the optimal control policy. These two perspectives on the feedback loop are contrasted in Fig. 1.1.

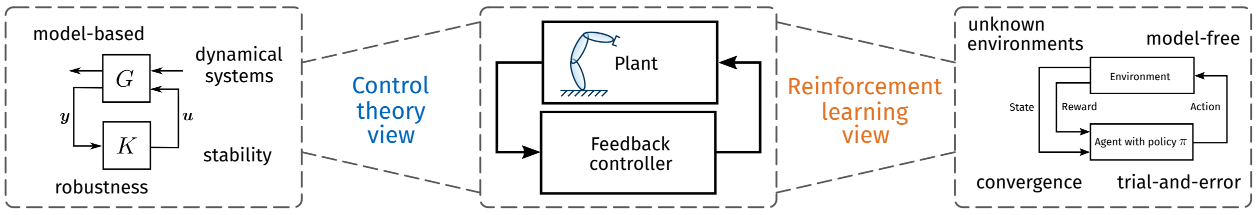


Figure 1.1: Juxtaposition of two perspectives on sequential decision making of intelligent agents with contrasting focus and priorities.

Research is contributed to the field of reinforcement learning from diverse backgrounds. For example, standard resources on reinforcement learning exist with perspectives from artificial intelligence [101, 228, 34, 253], operations research [181], and control [249]. Due to the close connection of the underlying Markov decision process (MDP) theory [182], also research from optimal control includes methods classifiable as reinforcement learning: the corresponding field is called approximate (adaptive) dynamic programming (ADP) [133, 135], and techniques dubbed neuro-dynamic programming [22] are similar. Thus, the methods from the field of control that are most clearly connected to reinforcement learning are optimal [7] and adaptive control [11]. Surveys have been appearing since the nineties to cover the intersection of these fields and explore their synergies [227, 133, 134, 104].

Due to the greatly different perception of the loop in Fig. 1.1 and priorities between control and AI, beyond approximate dynamic programming considerably less research exists that attempts to bridge the gap between control and reinforcement learning. Buşoniu *et al.* only lately emphasize "[. . .] the fact that AI researchers focus almost exclusively on performance with respect to a freely chosen reward function, while control objectives revolve around stability." [35] Recent publications [35, 109, 146, 187] however reflect the growing interest to explore synergies between the stability-centric view of systems and control and the performance as well as generality of reinforcement learning.

Robot learning control. In the robotics control community, the capability to learn has long been recognized as an important means to cope with the uncertainty arising in the real world; for example, Gullapalli *et al.* [70] already in the nineties use a multilayer (artificial) neural network (NN) and a reinforcement learning approach for robot skill acquisition. Considerable progress has since been made, *e. g.*, concerning scaling to higher dimensionality as reviewed in [201], learning rate and data efficiency [46], or model learning [216, 159]. Moreover, substantially more algorithmic approaches from reinforcement learning have been explored for robotic learning [115, 117]. However, compared to problem settings typical to the machine learning domain such as information retrieval, time series forecasting or speech recognition, feedback control is a less benign environment. The danger of instability clearly poses a threat and must be avoided in robotic learning to avoid damage in the system itself or to the environment around it. This may quickly become an issue particularly when dynamical systems are used to represent learned controllers for some skill, when learning in feedback, or when working with hardware.

From a control-theoretical perspective, concerning the stability of the robot control loop, we can classify contemporary skill learning methods into three major categories.

The first category of acquiring robotic motion skills is the very widespread approach to only learn in the feedforward control path and resort to a fixed, stabilizing feedback controller. For example, the popular dynamic movement primitive (DMP) [92, 93] consists of a stable second order system attracting to the goal and a parametric open-loop so-called forcing term which allows to represent smooth movements. Although the DMP contains a stable dynamical system inspired by feedback, integration over time yields a desired trajectory which is then tracked by any standard robot feedback controller. Some work [64] incorporates the actual robot state as a signal into the DMP; however, care should then be taken because such a step in general voids the by-design stability philosophy of the DMP approach. Other popular methods in this category, effectively acting as trajectory generators, are based on hidden Markov models or on Gaussian mixture models (GMMs) over time, see for example [38].

A remarkable trend throughout the last decade in robot learning research has been driven by the so-called "Dynamical Systems" approaches to programming by demonstration, see *e. g.* [25, 107, 188, 214, 55], leading to our second category. We sort methods into this category if the desired robot motion is generated (from a dynamical system) given the current state of the robot, as opposed to the trajectory-based representations of motion over time or over a time-like latent variable. The dynamical systems based approaches can be constructed from a small set of training data, easily generate new motions in unseen areas, are robust to perturbations, and react instantly to changes in the environment because the desired trajectory is constantly recalculated from the systems representation [188]. Again, a Gaussian mixture model is a typical representation to encode these skills and care must be taken to avoid the potential for instability of the dynamic model, *e. g.*, by enforcing a common Lyapunov function during the learning process [107]. Another quite versatile approach to ensuring stability of the motion model works indirectly by learning potential functions that act as a control Lyapunov function to correct when needed the motion commands of any regression model [106].

The methods previously categorized are concerned with representations and learning of robot motions. However, for collaborative robots working in uncertain environments, the energy exchange with the environment during physical interaction must be considered in order to learn the desired interactive behavior. Consequently, there is a need to understand and refine the role of impedance control [84] in this context: because of the difficulty and inflexibility of static robot programming, both impedance as well as motion parameters should be learnable from demonstrated data. However, the stability of the resulting feedback control structure is substantially harder to analyze, forming the third category of approaches.

First, this is because the feedback gains that realize active impedance are variable and it is well-known that ad-hoc gain scheduling bears the potential of instabilities [212]. Many approaches reported in the literature [32, 68, 121, 188, 267] simply ignore these potential pitfalls. Kronander and Billard therefore propose a method to verify impedance profiles [123] and a number of control approaches are also reported that are specifically tailored to the robotics problem domain, such as a biomimetic adaptive controller [263] or a control law mimicking energy reservoir ideas [54]. Another common approach helping to avoid stability issues is to collect trajectory and feedback data from demonstrations [68, 39, 121, 136, 157, 267], effectively biasing the subsequent learning process towards admissible behavior. While these approaches may oftentimes work in practice, particularly when damping is high and variations are slow, stability might nonetheless be lost in continued learning processes.

Second, the stability problem becomes even more severe when variable impedance control is combined with a dynamical systems approach to implicitly define motions. For example, learning variable impedance control may refer to learning both trajectories and stiffness/damping gain schedules with a combined DMP [32]. Another similar approach is proposed by Yang *et al.* [264], creating two DMPs for encoding motion and impedance characteristics measured from human muscle activation. Alternatively, the impedance profile is encoded in an additional dimension in the DMP, yielding a so-called compliant movement primitive [47]. All these methods benefit from the feedforward nature of DMP approaches but the general drawbacks of ad-hoc gain scheduling remain. Similarly, Rey *et al.* [188] ensure that the learned motion generation relies on a stable dynamical system, yet the learned stiffness schedules constitute a state-dependent gain scheduling feedback control law. Therefore, all these approaches work due to using admissible sets of demonstrations to derive the desired gain schedules. Nonetheless, care must be taken if some reinforcement learning module should be used to improve the policy further over time. Some work explicitly takes the stability problem into account when working with dynamical systems feedback motion planning. To this end, Khansari-Zadeh *et al.* [105] propose to unify motion generation and impedance control in a single asymptotically stabilizing Gaussian mixture model and provide a corresponding learning method [108]. Another approach is built around ensuring passivity [122]. However, even if a programming by demonstration approach can be chosen to provide expert training data, the robotic system may have to continuously optimize after initial learning in order to achieve satisfactory performance [115].

In summary, to realize the vision of collaborative robot manipulators with the capability to safely and autonomously learn feedback motion planning and interaction skills, more research is required for methods that ensure stability under a wide range of operating conditions.

(Closed-loop stable) machine learning control. Beyond the scope of robotics, an extensive body of literature exists on intelligent control methods [208], including control using genetic algorithms [120], neural network control [89], iterative learning control [30], fuzzy and neuro-fuzzy control [31], *etc.* When referring to learning control in this thesis, machine learning control is the only subfield of interest and the reader should refer to the literature otherwise. In particular reinforcement learning as outlined above as well as kernel methods [204] have been receiving a lot of attention from the controls community throughout the last decade, see for example [178] and the references therein.

The number of references that consider stability with reinforcement learning is much more limited. While RL control aims to handle general optimization objectives (*e. g.*, performance ranking by a human supervisor) and usually permits failing during the learning process due to the trial-and-error approach, adaptive control is subject to strict performance constraints [201]. Hence, the field of adaptive control offers a viable source of inspiration to address stability also for learning control methods. Using adaptive approximations [53], for example the work of Nakanishi *et al.* [155] falls into this category.

Concerning stability concepts for standard reinforcement learning methods, some efforts have been made beyond the approximate dynamic programming literature. Early research in this direction is limited to heuristics that should help to yield stabilizing policies. For example, Perkins and Barto [176] define different control laws which render stability in the sense of Lyapunov to the closed-loop system and the agent is only allowed to choose which one to apply in each specific point in time. A similar approach is reported for continuous actions spaces [56]. Thus, although [176, 56] resort to stabilizing policy elements, both essentially learn switching rules. However, switching between controllers may induce instability even if all controllers separately are stabilizing [137]. A congeneric problem applies to the actor-critic approach of Rosenstein and Barto [192] who connect a feedforward NN in parallel to a nominal controller, the so-called supervisor. The composition of the two elements is faded from the nominal controller to the actor NN over time, hence the method interpolates control outputs and eventually leaves stability to the authority of the NN actor.

A number of safe learning methods yield probabilistic stability or safety assertions to reinforcement learning controllers. For example, Berkenkamp and Schoellig [19] employ a Gaussian process (GP) to estimate uncertainties to adjust a robust controller. In [20], given an a priori stable control policy, an estimation of the region of attraction is iteratively expanded by means of a GP, allowing for safe policy updates within that region. A multitude of probabilistic reinforcement learning methods focuses on feedforward control, see [225] and the references therein.

Some work aiming for deterministic stability guarantees in conjunction with reinforcement learning agents exists as well. Ng and Kim [158] propose for linear systems a stable adaptive control algorithm for online variation of a state feedback controller by an arbitrary learning agent, based on a monitoring approach for time-varying feedback gains and rejection of control matrices that lead to instability. Kretchmar *et al.* [118] use a feedforward NN as actor in parallel to a nominal controller and employ integral quadratic constraintss (IQCs), a robustness tool known in the controls community [149], to ensure stability of the overall interconnection throughout the learning process. In [8], this idea is extended to recurrent NNs. Unfortunately, the method requires to re-evaluate the IQCs during the learning process and consequently the execution time to prove stability may become prohibitively high for problem settings such as the control of physical robot manipulators. Jin and Lavaei [99] more recently leverage IQCs to derive conditions solvable by semidefinite programming to certify via a smoothness argument the bounded-input bounded-output (BIBO) stability of the feedback loop under a deep RL policy. Similarly to the work of [20], Gillula *et al.* [66] develop a framework for quadcopter control which only allows to perform the control dictated by the machine learning algorithm if the system state is not near to the unsafe set which is approximately computed via reachability analysis. Contrary to the statistical approach of [20], problem-specific knowledge about the system dynamics is required in [66].

In summary, more methods of intelligent control are needed that ensure stability throughout the learning process in closed feedback loops.

Motivation. The research reported in this dissertation is driven by the following overarching question:

How may machine learning be employed for performance optimization of a (robotic) control system, such that stability is always preserved despite learning in the closed feedback loop?

In order to tackle this question, the specific research approach in this thesis is to employ machine learning in conjunction with a technical tool commonly known as the *parameterization* *of stabilizing controllers*. That is, contrary to purely data-driven learning, available *a priori* dynamic models of process and control are not dismissed but serve to structure the solution space to account for the effects of feedback. Hence, we aim to combine both perspectives depicted in Fig. 1.1 and exploit the domain knowledge available in robotics to realize an architecture for stable learning of feedback for robot manipulator control.

1.1 Challenges

The introduction above provides an overview of the vast fields of learning control and robot learning, summarizing the general background and motivation for this work. It is noted that this thesis contributes results for selected specific research aspects in this area: the central tool to address stability is the parameterization of stabilizing controllers, which will play a paramount role in the following chapters. The machine learning methods considered mainly belong to the class of reinforcement learning (RL) algorithms. Due to the difficulty outlined above of applying RL in feedback configurations on physical systems, another focus is on applicability to robotic hardware. The following questions portray the challenges addressed in this thesis.

Challenge 1. *How may model-based stabilizing controller parameterizations be constructed that are suitable for learning control on robotic manipulators?*
We will see that the established standard interpretations of the Q-parameterization of all stabilizing controllers may be unfavorable for implementation purposes on real robotic hardware. An additional central challenge is to understand how much prior model knowledge is required to obtain a useful parameterization. Once such a parameterization is obtained, how is the trade-off between model accuracy required for the parameterization and the search space available for learning characterized mathematically? How is the approach connected to robustness, such that we can use RL to adapt the control parameters on hardware? In order to develop answers to these questions, a substantial amount of the thesis resolves around the parameterization itself.

Challenge 2. *How may reinforcement learning methods be employed in the stabilizing parameterization approach to robotic learning control?*
In addition to the parameterization, some challenges related to the learning methods need to be addressed. Specifically, what conditions must be imposed on RL methods to leverage the benefits of the parameterization? What is a method of RL that is amenable to such modifications? How is the interplay between a RL learning mechanism and the parameterization generally characterized?

Challenge 3. *How is the free parameter (space) designed in particular robotic control applications?*
While challenge 1 refers to the structure of the control scheme, a related challenge is then how to find suitable parameters Q in the parameterization. How can the parameter be chosen when deploying learning to this controller structure? Given the proportional-derivative (PD) controller widespread in robotic control, how to recover PD-like behavior in the parameterization to allow for more intuitive tuning on hardware?

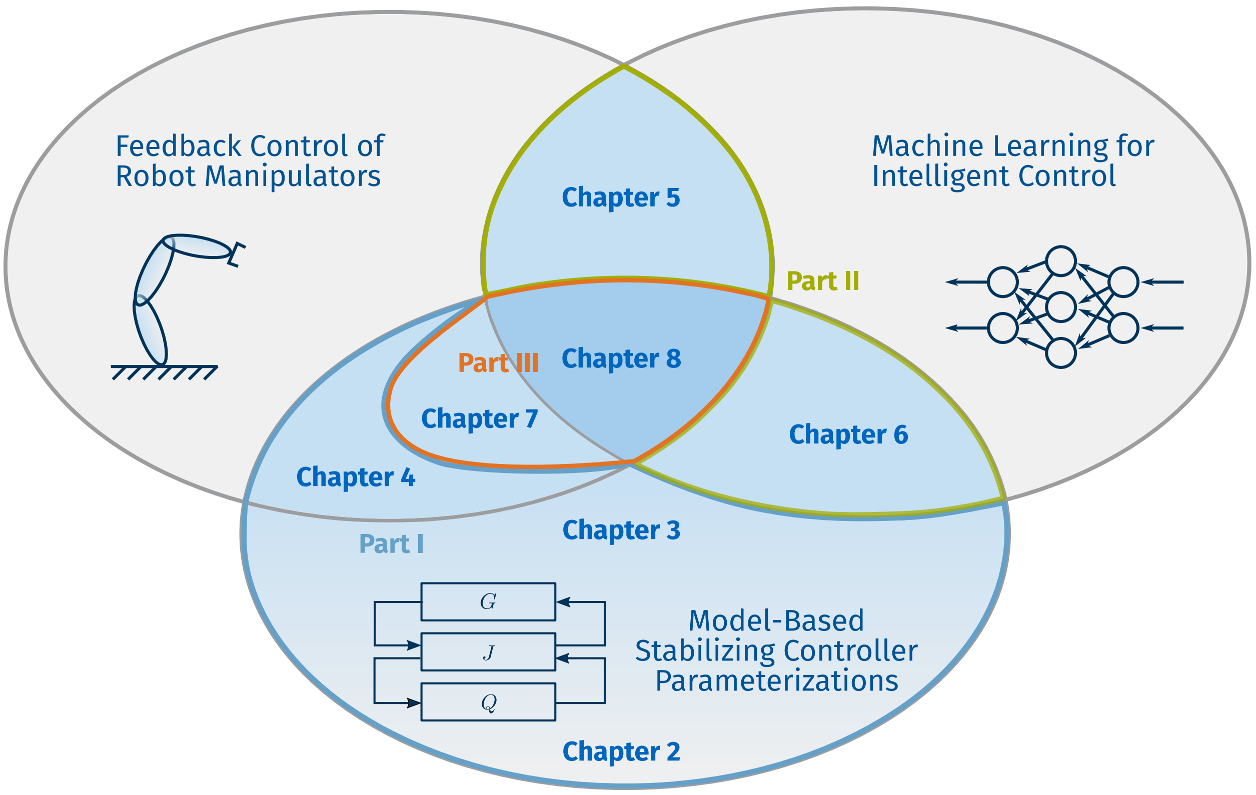

Feedback Control of Robot Manipulators
Machine Learning for Intelligent Control
Chapter 5
Part II
Part III
Chapter 8
Chapter 7
Chapter 6
Chapter 4
Chapter 3
Part I
G
J
Q
Model-Based Stabilizing Controller Parameterizations
Chapter 2

Figure 1.2: Overview of the structure of this thesis. The scientific background of the presented interdisciplinary research can be classified into the three technical areas *Model-Based Stabilizing Controller Parameterizations*, *Feedback Control of Robot Manipulators*, and *Machine Learning for Intelligent Control*. The methods presented in this thesis contribute mainly at the intersections of these areas.

1.2 Main Contributions and Outline

The main contribution of this thesis is to develop methods for an integration of machine learning with the parameterization of stabilizing controllers, characterizing a general framework that bridges model-free learning approaches with model-based control. Moreover, performance enhancement by reinforcement learning in feedback loops is considered particularly for robot manipulator control. Consequently, as shown in Fig. 1.2, the thesis covers specific aspects of robotics, machine learning, and control. The contributions are interdisciplinary and mainly where the technical areas *model-based stabilizing controller parameterizations*, *feedback control of robot manipulators*, and *machine learning for intelligent control* intersect. The methods are considered both theoretically as well as in case studies that summarize our experiments conducted on robotic hardware in the lab.

It is noted that Roberts *et al.* [190] consider the parameterization of stabilizing controllers in a basic performance comparison between several standard feedback controller structures in conjunction with a simple RL algorithm. While Roberts *et al.* conclude that the parameterization performed best in their simulation study, none of the challenges outlined in Sec. 1.1 has been conclusively dealt with.

Next, the contributions of this thesis are listed in greater detail, in order of the organization of the thesis. The interested reader may refer to Appendix A for some background information on dynamical systems in general and on the coprime factorization approach to control in particular. Part I considers stabilizing controller parameterizations and their usage in adaptive control with particular attention to manipulator control. Given these perspectives of the parameterization and robotics, intelligent control methods are developed in Part II. First, a RL algorithm is presented that can be used in conjunction with the parameterization while being tailored towards robotics; next, the interplay between RL and the parameterizations are investigated more generally. Part III of the thesis presents laboratory case studies to eventually provide experimental evidence for the efficacy of the control strategies studied in this thesis, integrating elements of all three technical disciplines.

Part I: Model-Based Stabilizing Controller Parameterizations for Robotics

Introduction to the stabilizing controller parameterization (Chapter 2). Chapter 2 first presents introductory material about the class of all stabilizing controllers as well as the dual parameterization of stabilized plants, as well as the resulting double parameterization. We then review the most relevant applications in control to establish the relation of the methods that will be developed in the subsequent chapter w. r. t. the state of the art.

Adaptive control in the parameterization with multiple modes (Chapter 3). Given the background on existing parameterizations of Chap. 2, adaptive control methods that are based on local modes and the parameterization will be developed in this chapter.

The first focus in Sec. 3.1 is on the derivation of a parameterization that will allow to be implemented on robotic hardware in a later part of the thesis. Stability of a linear system under fast switching or blending of a set of controllers can be ensured by an appropriate observer-based state-space realization. In this chapter, the more specific problem is considered of arbitrary interpolation of a set of state feedback gains based on an initial static state feedback. First, the dynamic augmentation generating this parameterization is derived as well as the associated parameters for local recovery of predefined static controllers. By further simplification, a simple and intuitive structure is obtained with only a single design matrix. We propose to exploit this remaining degree of freedom to maximize robustness in terms of coprime factor uncertainty. The resulting parameterization is comparatively simple to implement in both continuous and discrete time and the robotics problem of active variable impedance control serves to illustrate utility of this parameterization.

Section 3.2 is concerned with some theoretical properties of the parameterization in switching plants, as these may occur in robotic contact situations. Specifically, we will investigate the problem of designing an adaptive performance enhancement control law for an arbitrarily fast switching linear plant given a switching compensator. Switching is assumed to be uncontrolled and the characteristics of the reference or disturbance signals are changing over time. Stability despite switching and adaptation are given by construction, using a parameterization of all quadratically stabilizing compensators. It is then shown that the so-called adaptive-Q control methodology is well suited to enhance performance online for switching

linear cases. A simulation example of an unstable switching plant illustrates the efficacy of the method and the performance enhancement when tracking a bounded reference signal with unknown characteristics.
Parts of this chapter were previously published in the *IEEE Control Systems Letters* [58] and in the *IEEE Conference on Decision and Control* [59].

Robust manipulator control: double-Youla approach (Chapter 4). The simple parameterization over state feedback of the previous chapter assumed the availability of a perfectly accurate dynamical process model. In practical robotic settings, this assumption is hard to fulfill. Therefore, in this chapter, the following question is tackled: Given some initial engineering knowledge, *i. e.*, a simplified dynamic model of the robot and a nominal controller, *how* and *how much* can the feedback controller be modified on the real system without sacrificing stability of the closed loop? Intuitively, the less accurate the model and the more uncertain the nominal loop, the more restricted any such modification will have to be. The method proposed in this chapter allows to *quantify* this trade-off. Consequently, by the proposed framework, the admissible search space can be systematically tightened to contain only robustly stabilizing controllers for robot manipulators. Thus, the overall contribution of this chapter is to provide a novel robot manipulator control framework to assure stability to the loop in spite of online modifications of the feedback controller.
The material of this chapter was published in the *International Journal of Robust and Nonlinear Control* [60].

Part II: Model-Free Learning Control from the Robotics and Controller Parameterization Viewpoints

In the first part of the thesis, the stabilizing controller parameterizations are developed with a special focus on being implementable on robotic systems. In Part II, results are reported concerning the quest of combining such a parameterization approach with RL algorithms.

Automated continuous online least-squares policy iteration (Chapter 5). First, in this chapter, we consider a specific class of RL, namely least-squares policy iteration (LSPI) algorithms. This choice was motivated by taking into account that the algorithm should be analyzable in the framework of stabilizing parameterizations. Moreover, for robot control problem settings, it is oftentimes characteristic that the algorithms have to learn online through interaction with the system while it is operating, and that both state and action spaces are continuous. Least-squares policy iteration (LSPI) based approaches are therefore particularly hard to employ in practice, and parameter tuning is a tedious and costly enterprise. In order to mitigate this problem, we derive an automatic online LSPI algorithm that operates over continuous action spaces and does not require an a-priori, hand-tuned value function approximation architecture. To this end, we first show how the kernel least-squares policy iteration algorithm can be modified to handle data online by recursive dictionary and learning update rules. Next, borrowing sparsification methods from kernel adaptive filtering, the continuous action-space approximation in the online least-squares policy iteration algorithm can be ef-

ficiently automated as well. We then propose a similarity-based information extrapolation for the recursive temporal difference update in order to perform the dictionary expansion step efficiently in both algorithms. The performance of the proposed algorithms is compared with respect to their batch or hand-tuned counterparts in a simulation study. The novel algorithms require less prior tuning and data is processed completely on the fly, yet the results indicate that similar performance can be obtained as by careful hand-tuning. Therefore, engineers from both robotics and AI can benefit from the proposed algorithms when an LSPI algorithm is faced with online data collection and tuning by experiment is costly.
The material of this chapter appeared previously in the journal of the IFAC *Engineering Applications of Artificial Intelligence* [61].

A synergistic perspective of reinforcement learning and model-based controller parameterizations (Chapter 6). While the previous chapter considered LSPI algorithms from the perspective of robotics, in this chapter RL algorithms in general are investigated from the viewpoint of the model-based stabilizing controller parameterization. Contrary to standard RL, we incorporate prior dynamical model knowledge to construct the parameterization and leverage RL in a stability-by-design paradigm. The main contribution of this chapter is to systematically unravel for the first time the general interplay and the relations between RL and the Q-parameterization. To this end, we first give a control-relevant classification of RL. Then, five different architectures are outlined of how to employ RL in the parameterization. It will be shown that, contrary to intuition, the affine property of the Q-parameterization may actually pose an obstacle in combination with RL, depending on the class of learning algorithm. Next, model uncertainty is taken into consideration and it will be shown how to impose the required norm bounds over any value-based critic-only RL algorithm. The chapter concludes by discussing the overall framework by means of a reference workflow guiding through the construction of combined stabilizing parameterizations reinforcement learning controllers. The student thesis [205] partly contributed to the results presented in this chapter.

Part III: Laboratory Case Studies

In the third part, experimental evidence will be reported for the efficacy of the control strategies developed in part I and II of the thesis.

Active variable impedance control (Chapter 7). The first case study concerns the implementation of the parameterization developed in Chap. 3 on hardware. The overall contribution is to show that we can construct parameterizations for active variable impedance control suitable for deployment to a KUKA lightweight robot (LWR) IV+. The key idea is that although the dynamics of the underlying robotic control system is not exactly known, the nominal gains constitute a design degree of freedom to decrease uncertainty in a suitable controller parameterization. It will be shown that the desired control modes computed from the Q-parameters can be recovered using the superimposed torque interface of the robot. Our study experimentally confirms that the novel control architecture based on the Q-parameterization leads to admissible behavior under interpolation conditions that lead to instability when naively implemented. This result implies that in future work, RL can be employed straightforwardly to learn stiffness schedules without the risk of instability due to interpolation-induced hidden coupling. The chapter concludes with an outlook to a new way of implementing variable impedance feedback motion planning skills based on the methods proposed in the thesis. The student theses [165, 194] partly contributed to the case study presented in this chapter.

Episodic performance enhancement in tracking control (Chapter 8). The second case study contains elements of all three technical areas of the thesis. The novel framework is employed to safely enhance the performance of the tracking controller with machine learning. To this end, first we discuss how to obtain a suitable dynamic model from prior knowledge about the robot manipulator without excessive modeling effort. Next, a learning framework is constructed upon a parameterization tailored for PD control and an episodic learning approach. In other words, one of the architectures discussed in Chap. 6 was implemented, based on the parameterization derived in Chap. 4. Simulation studies as well as an experiment on a two degree of freedom (DoF) robotic manipulator were conducted. The results confirm that the novel methods constitute a stability framework applicable to real-world robots, such that performance enhancement is feasible by learning directly on hardware.
The contributions of this chapter were presented at the *IEEE International Conference on Robotics and Automation* [57].

Part I

Model-Based Stabilizing Controller Parameterizations for Robotics

2

Introduction to the Stabilizing Controller Parameterization

This chapter provides an introductory overview of the parameterization of stabilizing controllers and reviews its utility in selected hybrid, adaptive, and learning control approaches.

First, in Sec. 2.1, a concise summary of the controller parameterization based on coprime factorizations is given. This parameterization is a classic result in control theory, used in a wide range of applications. It is one of the core concepts used in this thesis; therefore, the chapter is supposed to provide the necessary background. The reader may also refer to Appendix A for basic concepts of dynamical systems, the stability concepts of relevance in this thesis, and the representation of matrix transfer functions by means of coprime factorizations. Next, in Sec. 2.2, the dual parameterization of plants stabilized by a given controller is summarized as well as the combined primal and dual parameterization, yielding a framework to simultaneously deal with both plant and controller uncertainty. In Sec. 2.3, the role of the controller parameterization is summarized for the problem of interpolation and arbitrary switching among controllers. Finally, some selected approaches that make use of the parameterization in the context of adaptive and learning control are reviewed in Sec. 2.4. The introductory expositions presented in this chapter were partly used in previous publications [57, 58, 59, 60].

2.1 Class of All Stabilizing Feedback Controllers

The class of all stabilizing feedback controllers for a plant can be described by means of an interconnection between an initial stabilizing controller and a stable parameter system. Note that this parameter system is often, though not consistently, denoted Q in the control literature. The resulting parameterization is therefore oftentimes called *Q-parameterization*, and it is well covered by standard reference literature [6, 269, 231, 246]. The parameterization is also dubbed *Youla parameterization*, *Kučera parameterization*, *Youla-Kučera parameterization*, *Youla-Jabr-Bongiorno-Kučera* by some authors; arguably, both YOULA *et al.* [265] and KUČERA [124] independently described the parameterizations. Closely related is also VIDYASAGAR's *factorization approach* to control [246] which is summarized in Appendix A.3. In what follows, a concise and intuitive explanation of the parameterization is given. For more complete treatments and detailed expositions, the interested reader is referred to [6, 269, 231, 246].

Consider a continuous- or discrete-time system G such that

$$\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} G_{zw} & G_{zu} \\ G_{yw} & G_{yu} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{u} \end{bmatrix} \tag{2.1}$$

and assume G_{zw}, G_{zu}, $G_{yw} \in \mathcal{RH}_\infty$. Then, if the feedback loop (G_0, K_0) formed by $G_0 = G_{yu}$ and a (nominal) controller K_0 is well-posed and internally stable, the closed-loop matrix transfer function $T_{zw} = G_{zw} + G_{zu}K_0 \left(I - G_{yu}K_0\right)^{-1} G_{yw}$ is also stable.

As shown next, for linear systems, the set of *all* controllers that stabilize a given system G can be expressed in terms of a stable parameter system Q and two stable rational matrix factors of the system G and a controller K_0 that satisfy a Bezout identity.

2.1.1 Controller Parameterization by Coprime Factorization

To parameterize all stabilizing controllers by a factorization [246], G_0 and K_0 are written in terms of double coprime factorizations

$$G_0 = \tilde{M}_0^{-1}\tilde{N}_0 = N_0 M_0^{-1} \text{ and } K_0 = \tilde{V}_0^{-1}\tilde{U}_0 = U_0 V_0^{-1}, \tag{2.2}$$

$$\text{where } N_0, M_0, \tilde{N}_0, \tilde{M}_0, V_0, U_0, \tilde{V}_0, \tilde{U}_0 \in \mathcal{RH}_\infty, \tag{2.3}$$

such that the left and right factors have no unstable pole-zero cancellations, *i. e.*, chosen to satisfy the double Bezout identity

$$\begin{bmatrix} \tilde{V}_0 & -\tilde{U}_0 \\ -\tilde{N}_0 & \tilde{M}_0 \end{bmatrix} \begin{bmatrix} M_0 & U_0 \\ N_0 & V_0 \end{bmatrix} = \begin{bmatrix} M_0 & U_0 \\ N_0 & V_0 \end{bmatrix} \begin{bmatrix} \tilde{V}_0 & -\tilde{U}_0 \\ -\tilde{N}_0 & \tilde{M}_0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{2.4}$$

The following result is summarized *e. g.* from [269, Th. 12.17] and [231, Th. 2.5.1].

Proposition 2.1 (Q-parameterization). Given coprime factorizations (2.3) of the plant G_0 and the controller K_0 that fulfill (2.4), the set $\mathcal{K}$ of stabilizing controllers for G_0 can be characterized in terms of an arbitrary stable parameter Q as $\mathcal{K} = \{K(Q) \mid Q \in \mathcal{Q} \subseteq \mathcal{RH}_\infty\}$,

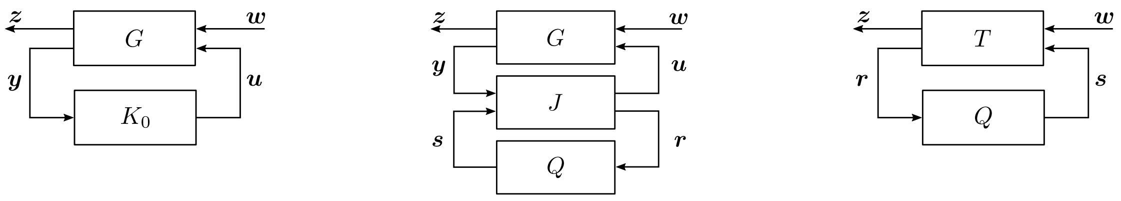
$$K(Q) = U(Q)V(Q)^{-1}, \text{ where } U(Q) = U_0 + M_0 Q, \quad V(Q) = V_0 + N_0 Q \tag{2.5}$$

to obtain a right factored form, or

$$K(Q) = \tilde{V}^{-1}(Q)\tilde{U}(Q), \text{ where } \tilde{V}(Q) = \tilde{V}_0 + Q\tilde{N}_0, \quad \tilde{U}(Q) = \tilde{U}_0 + Q\tilde{M}_0 \tag{2.6}$$

for a left stable linear fractional form. Reformulating from the definition, using the Bezout identity, (2.5) can be reformulated as

$$K(Q) = \tilde{V}_0^{-1}\tilde{U}_0 + \tilde{V}_0^{-1}Q\left(I + V_0^{-1}N_0 Q\right)^{-1} V_0^{-1} \triangleq \mathcal{F}_\ell\left(J, Q\right), \tag{2.7}$$



(a) Standard feedback loop **(b)** Q-parameterization **(c)** Closed loop and Q-parameter

Figure 2.1: Reformulation of a feedback controller in terms of a lower fractional transformation of an initial stabilizing controller and a stable parameter system Q.

where $Q \in \mathcal{RH}_\infty$ such that $(I + V_0^{-1}N_0Q)(\infty)$ is invertible and the generator system J can be implemented as

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} \tilde{V}_0^{-1}\tilde{U}_0 & \tilde{V}_0^{-1} \\ V_0^{-1} & -V_0^{-1}N_0 \end{bmatrix} = \begin{bmatrix} U_0V_0^{-1} & \tilde{V}_0^{-1} \\ V_0^{-1} & -V_0^{-1}N_0 \end{bmatrix}. \tag{2.8}$$

Proof: See [269, p. 324f] and [231, p. 41f]. ■

In words, by a Q-parameterization shown in Fig. 2.1b, every linear internally stabilizing controller K can be implemented as a lower linear fractional transformation (LFT) $K(Q) = \mathcal{F}_\ell\left(J, Q\right)$. That is, some central system J is interconnected with another stable filter $Q \in \mathcal{RH}_\infty$ such that

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{r} \end{pmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{s} \end{pmatrix}, \quad \boldsymbol{s} = Q\boldsymbol{r}. \tag{2.9}$$

Note that the entry J_{11} corresponds to the nominal controller K_0.

Proposition 2.2 (All stabilizing controllers). With Q varying over $\mathcal{RH}_\infty$, Prop. 2.1 characterizes the set of *all* possible proper stabilizing controllers for G_0.

Proof: See [269, p. 324f] and [231, p. 41f]. ■

The relationship between K and Q is bijective, and the nominal controller K_0 is recovered by $Q = 0$. The pre-stabilized system obtained by interconnecting G with J is henceforth denoted T, yielding the interconnection shown in Fig. 2.1c. The corresponding signal mapping can be shown [231, p. 47] to be

$$\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{r} \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} G_{zw} + G_{zu}U_0\tilde{M}_0G_{yw} & G_{zu}M_0 \\ \tilde{M}_0G_{yw} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{s} \end{bmatrix}. \tag{2.10}$$

Hence, in the standard parameterization, the closed-loop system relating the exogenous inputs $\boldsymbol{w}$ to the performance signals $\boldsymbol{z}$ is

$$T_{zw}(Q) = G_{zw} + G_{zu}U_0\tilde{M}_0G_{yw} + G_{zu}M_0Q\tilde{M}_0G_{yw}. \tag{2.11}$$

Note that (2.11) is of the well-known [28, 269, 51, 231] affine form

$$T_{zw}(Q) = T_{11} + T_{12}QT_{21}, \tag{2.12}$$

leading to *convexity* for many standard control objectives in the parameter Q.

Remark 2.1 *(Q-parameterization closed-loop properties, ideal case).*

- Note from (2.10) that $T_{22} = 0$, *i. e.*, the gain from the output of the system Q to its input is zero in the *closed-loop* system. This property can be intriguing: the signal $\boldsymbol{r}$ driving the plug-in filter Q is not affected by the specific choice of Q, because the signal does not depend on $\boldsymbol{s}$. In consequence, the feedback loop is "unwrapped" based on the nominal model and controller, leading to the convenient formula (2.12).
- Due to the affine form of (2.12), the parameterization approach allows for efficient optimization-based design of the feedback controller, for both offline optimal controller design [28] and online, adaptive versions [231]. Therefore, when using a machine learning approach to optimize performance, the cost function will be better suited for numerical search as well by employing a Q-parameterization to implement the controller, *cf.* [190].

$\triangleleft$

2.1.2 State-Space Interpretation of Stabilizing Controllers

While the Q-parameterization above is constructed by means of the coprime factors involved, in this section, state-space interpretations of the parameterization of stabilizing controllers are given. Therefore, although originally developed in the frequency domain, the parameterization can be calculated and set up using state-space models. The viewpoint from state-space allows for a more transparent understanding of the controller structure and can be advantageous from a numerical controller design perspective. To proceed in state space, let the plant be given by

$$G : \left[\begin{array}{c|cc} \boldsymbol{A} & \boldsymbol{B}_1 & \boldsymbol{B}_2 \\ \hline \boldsymbol{C}_1 & \boldsymbol{D}_{11} & \boldsymbol{D}_{12} \\ \boldsymbol{C}_2 & \boldsymbol{D}_{21} & \boldsymbol{D}_{22} \end{array}\right], \tag{2.13}$$

with the inherited realization of the control channel $G_{yu} : \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B}_2 \\ \hline \boldsymbol{C}_2 & \boldsymbol{D}_{22} \end{array}\right]$. Internal stability of the controlled general setup is then equivalent to internal stability of the feedback control loop with G_{yu}, provided the realization is stabilizable and detectable from $\boldsymbol{u}$ and $\boldsymbol{y}$, respectively.

Proposition 2.3 (Internal stability via G_{yu} [269]). Let $(\boldsymbol{A}, \boldsymbol{B}_2)$ and $(\boldsymbol{A}, \boldsymbol{C}_2)$ be stabilizable and detectable pairs, respectively. Then, the system $\mathcal{F}_\ell\,(G, K)$ is internally stable iff the loop (G_{yu}, K) is internally stable.

Proof: See Lemma 12.2 in [269]. ■

State-estimate feedback (observer-based) initial controller. The most common state-space form of the Q-parameterization is the Youla-Kučera parameterization based on a state-estimate (observer-based) central feedback controller K_0. This parameterization can be derived directly in state space, see *e. g.* [269, Ch. 12.3], as well as from suitable coprime factors as in [231, Ch. 2.5] or [269, Ch. 12.6].

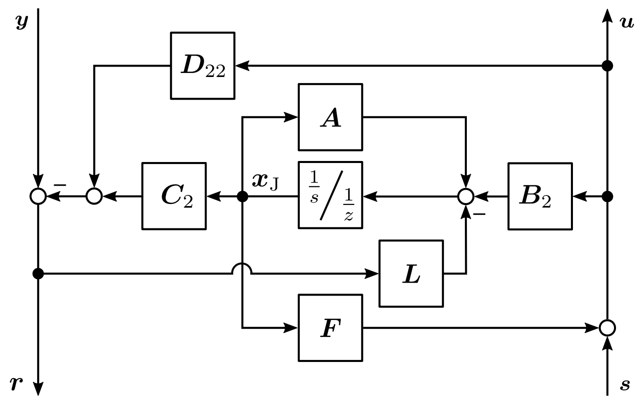


Figure 2.2: In the most common state-space interpretation of stabilizing controllers, the central system J is an observer-based (state-estimate) feedback controller. The signal $\boldsymbol{r}$ then simply corresponds to the residual between measured quantities $\boldsymbol{y}$ and their estimates. The filtered signal $\boldsymbol{s} = Q\boldsymbol{r}$ is input to both the state estimate controller and the actual plant.

To construct the central controller, let $\boldsymbol{F}$ and $\boldsymbol{L}$ be such that $\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A} + \boldsymbol{L}\boldsymbol{C}_2$ are stable. In order to generate the parameterization, the estimation residual is simply becoming the input $\boldsymbol{r} \in \mathbb{R}^{n_\mathrm{y}}$ to the block Q. The filtered $\boldsymbol{s} = Q\boldsymbol{r}$, $\boldsymbol{s} \in \mathbb{R}^{n_\mathrm{u}}$ is then an input to both the state estimate controller and the actual plant. The corresponding central system is given by

$$J : \begin{cases} \delta \boldsymbol{x}_\mathrm{J} = & (\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F} + \boldsymbol{L}\boldsymbol{C}_2 + \boldsymbol{L}\boldsymbol{D}_{22}\boldsymbol{F})\,\boldsymbol{x}_\mathrm{J} \\ & -\ \boldsymbol{L}\boldsymbol{y} + (\boldsymbol{B}_2 + \boldsymbol{L}\boldsymbol{D}_{22})\,\boldsymbol{s}, \quad \boldsymbol{x}_{\mathrm{J},0} = \boldsymbol{0} \\ \boldsymbol{u} = & \boldsymbol{F}\boldsymbol{x}_\mathrm{J} + \boldsymbol{s}, \\ \boldsymbol{r} = & -\ (\boldsymbol{C}_2 + \boldsymbol{D}_{22}\boldsymbol{F})\,\boldsymbol{x}_\mathrm{J} + \boldsymbol{y} - \boldsymbol{D}_{22}\boldsymbol{s}. \end{cases} \tag{2.14}$$

For the derivation of (2.14), the reader is referred to [269, Th. 12.8] or [231, Ch. 2.5]. The structure corresponding to (2.14) is depicted in Fig. 2.2.

Proposition 2.4 (State-estimate feedback based parameterization [269, Th. 12.8]). All controllers that internally stabilize G can be parameterized as $\mathcal{F}_\ell\,(J, Q)$, with J from (2.14) and any $Q \in \mathcal{RH}_\infty$ and $\boldsymbol{I} + \boldsymbol{D}_{22}Q(\infty)$ nonsingular.

Proof: See [269, p. 312]. ∎

Static initial controller. It is common to use the state-estimate feedback structure Fig. 2.2 to provide a state-space interpretation of all stabilizing controllers as summarized above and first discussed in [152]. However, an alternative parameterization can be constructed using a static controller $K_0 \triangleq \boldsymbol{D}_{\mathrm{K},0}$ as nominal feedback controller in the generator system J. Although such a parameterization based on static state feedback has been used in the context of $\mathcal{H}_2$-control at least since [195], this parameterization is a lot less prominent in the literature; it is typically not found in standard reference book that discuss the Youla parameterization, *e. g.*, [231, 269, 51, 28, 81].

Proposition 2.5 (Generator system based on static feedback). Let $\boldsymbol{F}$ be such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}$ is stable and $\boldsymbol{D}_{\mathrm{K},0}$ such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{D}_{\mathrm{K},0}$ is stable. For $(\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})^{-1}$ invertible, *i. e.*, a well-posed loop, a generator system based on $\boldsymbol{u}=\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{y}$ is

$$J:\begin{cases}\delta\boldsymbol{x}_{\mathrm{J}} = & (\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F})\,\boldsymbol{x}_{\mathrm{J}}+\boldsymbol{B}_2\boldsymbol{s}, \quad \boldsymbol{x}_{\mathrm{J},0}=\boldsymbol{0}\\ \boldsymbol{u} = & ((\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})\boldsymbol{F}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2)\,\boldsymbol{x}_{\mathrm{J}}\\ & +\ \boldsymbol{D}_{\mathrm{K},0}\,\boldsymbol{y}+(\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})\,\boldsymbol{s},\\ \boldsymbol{r} = & -\ (\boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F})\,\boldsymbol{x}_{\mathrm{J}}+\boldsymbol{y}-\boldsymbol{D}_{22}\boldsymbol{s}.\end{cases} \tag{2.15}$$

Proof: This parameterization follows from (2.8) with suitable coprime factors. To this end, state-space realizations are constructed for the case of a static initial controller by reducing the general formulae given in (A.21) for a controller $K_0:\left[\begin{array}{c|c}\star & \boldsymbol{0}\\ \hline \boldsymbol{0} & \boldsymbol{D}_{\mathrm{K},0}\end{array}\right]$, yielding

$$\begin{aligned}\begin{bmatrix}M_0 & U_0\\ N_0 & V_0\end{bmatrix} &: \left[\begin{array}{c|cc}\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{B}_2 & \boldsymbol{0}\\ \hline \boldsymbol{F} & \boldsymbol{I} & \boldsymbol{D}_{\mathrm{K},0}\\ \boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F} & \boldsymbol{D}_{22} & \boldsymbol{I}\end{array}\right],\\ \begin{bmatrix}\tilde{V}_0 & -\tilde{U}_0\\ -\tilde{N}_0 & \tilde{M}_0\end{bmatrix} &: \left[\begin{array}{c|cc}\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{Y}\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2 & -\boldsymbol{B}_2\boldsymbol{Y} & \boldsymbol{B}_2\boldsymbol{Y}\boldsymbol{D}_{\mathrm{K},0}\\ \hline \boldsymbol{F}-\boldsymbol{Y}\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2 & \boldsymbol{Y} & -\boldsymbol{Y}\boldsymbol{D}_{\mathrm{K},0}\\ \boldsymbol{Z}\boldsymbol{C}_2 & -\boldsymbol{Z}\boldsymbol{D}_{22} & \boldsymbol{Z}\end{array}\right],\end{aligned} \tag{2.16}$$

where $\boldsymbol{Y}\triangleq(\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})^{-1}$ and $\boldsymbol{Z}\triangleq(\boldsymbol{I}-\boldsymbol{D}_{22}\boldsymbol{D}_{\mathrm{K},0})^{-1}$. Removing uncontrollable states, the following factors are obtained:

$$U_0:\left[\begin{array}{c|c}\star & \boldsymbol{0}\\ \hline \boldsymbol{0} & \boldsymbol{D}_{\mathrm{K},0}\end{array}\right],\quad V_0:\left[\begin{array}{c|c}\star & \boldsymbol{0}\\ \hline \boldsymbol{0} & \boldsymbol{I}\end{array}\right],\quad N_0:\left[\begin{array}{c|c}\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{B}_2\\ \hline \boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F} & \boldsymbol{D}_{22}\end{array}\right].$$

$$\tilde{V}_0:\left[\begin{array}{c|c}\boldsymbol{A}+\boldsymbol{B}_2(\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})^{-1}\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2 & -\boldsymbol{B}_2(\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})^{-1}\\ \hline \boldsymbol{F}-(\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})^{-1}\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2 & (\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})^{-1}\end{array}\right].$$

Inverting $\tilde{V}_0$ results in

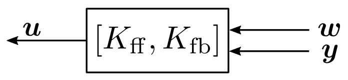
$$\tilde{V}_0^{-1}:\left[\begin{array}{c|c}\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{B}_2\\ \hline (\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})\,\boldsymbol{F}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2 & (\boldsymbol{I}-\boldsymbol{D}_{\mathrm{K},0}\boldsymbol{D}_{22})\end{array}\right],$$

and $-V_0^{-1}N_0$ gives

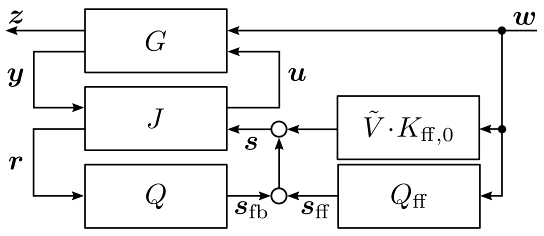
$$-V_0^{-1}N_0:\left[\begin{array}{c|c}\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{B}_2\\ \hline -\boldsymbol{C}_2-\boldsymbol{D}_{22}\boldsymbol{F} & -\boldsymbol{D}_{22}\end{array}\right].$$

The realization (2.15) is eventually obtained by stacking the systems according to (2.8) and removal of one uncontrollable stable mode. ■

A number of special cases of the general parameterization (2.15) occurred in the literature in different contexts. Particularly, special cases of (2.15) were used for $\mathcal{H}_2$-control [195], switching controllers [164], intermittent redesign [150] and robotic learning control [57].



(a) General two-degree-of-freedom controller



(b) Parameterization over two-degree-of-freedom controllers

Figure 2.3: Two-degree-of-freedom controllers and structure of corresponding parameterization of all stabilizing controllers.

2.1.3 Parameterization of Two-Degree-of-Freedom Controllers

For the study of the interplay between learning and (feedback) control in the later chapters of this thesis, it is crucial to clearly distinguish feedback from feedforward control action. Let the exogenous signal $\boldsymbol{w}$ contain measurable quantities, *e. g.*, some desired output trajectory. Feedforward signals can then be processed independently from the feedback controller, *i. e.*, $\boldsymbol{u} = \boldsymbol{u}_{\text{ff}} + \boldsymbol{u}_{\text{fb}} = [K_{\text{ff}}, K_{\text{fb}}] \left[\boldsymbol{w}^\top, \boldsymbol{y}^\top\right]^\top$. This situation is depicted in Fig. 2.3a.

Remark 2.2 *(Feedforward control elements).* "Feedforward elements link measured disturbances to manipulated inputs." [217, p. 421] For the developments in later parts of this thesis, the following property of measured disturbances is emphasized: by definition, the gain from measured quantities $\boldsymbol{y}$, performance quantities $\boldsymbol{z}$, and manipulated inputs $\boldsymbol{u}$ to exogenous inputs (measured disturbances) $\boldsymbol{w}$ is zero. This statement is rather obvious but has a profound consequence in the selection of algorithmic classes of reinforcement learning. This matter will be discussed in Chap. 6.1.2. $\triangleleft$

For analysis, it is convenient to consider the two-degree-of-freedom controller as a single controller $K = [K_{\text{ff}}, K_{\text{fb}}]$ in feedback connection with an augmented plant $\left[0,\ G^\top\right]^\top$ [231, p. 32f]. All two-degree-of-freedom controllers stabilizing G are obtained [231, p. 49f, p. 53] as all one-degree-of-freedom controllers for $\left[0,\ G^\top\right]^\top$. As spelled out in [231, p. 49f], using augmented coprime factors, the class of all stabilizing feedforward/feedback controllers can then be constructed as depicted in Fig. 2.3b.

For simplicity, in the remainder of this thesis, we assume that the nominal feedforward controller $K_{\text{ff},0}$ is stable, *i. e.*, $K_{\text{ff},0} \in \mathcal{RH}_\infty$. Consequently, to parameterize all stabilizing feedforward/feedback controllers, only a feedforward component $\boldsymbol{s}_{\text{ff}} = Q_{\text{ff}}\boldsymbol{w}$ has to be added to the output of the corresponding feedback controllers, *i. e.*, $\boldsymbol{s} = \boldsymbol{s}_{\text{ff}} + \boldsymbol{s}_{\text{fb}} = [Q_{\text{ff}}, Q] \left[\boldsymbol{w}^\top, \boldsymbol{r}^\top\right]^\top$ as in Fig. 2.3b. Internal stability of the resulting control loop is exclusively determined by the feedback loop, provided $\boldsymbol{w}$ is bounded and $K_{\text{ff},0}, Q_{\text{ff}} \in \mathcal{RH}_\infty$.

Summary. The significance of the parameterization for learning-based control is due to the fact that, once a single stabilizing feedback controller is known, it is possible to obtain *all stabilizing* feedback controllers for a given plant G by variation of a *stable* parameter. The problem of finding stabilizing feedback controllers K is transformed to a search only over stable operators Q. This is a much simpler constraint to enforce during learning. This advantage, however, generally comes at the cost of needing a dynamical process model in the implementation of the feedback controller.

2.2 Plant Uncertainty Description in the Parameterization

2.2.1 Dual Youla Parameterization

Interchanging the role of the plant and the controller in the derivation, one can parameterize all plants that are stabilized by a given controller K_0 [6]. A widespread convention in the literature is to denote this free parameter system S [231, 160]. The resulting parameterization is also referred to as *dual Youla* parameterization [160] in the remainder.

Proposition 2.6 (S-parameterization [231, Th. 3.4.1]). Given coprime factorizations (2.3) of the nominal plant G_0 and the controller K_0 that fulfill (2.4), the set of (all) proper plants stabilizable by K_0 is characterized in terms of an arbitrary stable operator S as $\mathcal{G} = \{G(S) \mid S \in \mathcal{S} \subseteq \mathcal{RH}_\infty\}$, where

$$G(S) = N(S)M(S)^{-1}, \quad \text{with} \quad N(S) = N_0 + V_0 S, \quad M(S) = M_0 + U_0 S, \quad \text{or} \tag{2.17}$$

$$G(S) = \tilde{M}(S)^{-1}\tilde{N}(S), \quad \text{with} \quad \tilde{M}(S) = \tilde{M}_0 + S\tilde{U}_0, \quad \tilde{N}(S) = \tilde{N}_0 + S\tilde{V}_0. \tag{2.18}$$

Proof: The coprime factors of $G(S)$ and of K_0 satisfy the Bezout identity, hence the result is dual to Prop. 2.1. A detailed derivation is spelled out in [231, Chap. 3.4]. ■

The relation between any S and the corresponding $G(S)$ is bijective, and $S \notin \mathcal{RH}_\infty$ implies that $(G(S), K_0)$ does not constitute a stabilizing loop. Analogously to the controller parameterization, the dual setup can be reformulated in terms of a LFT in S; precisely, using (2.4), from (2.17) it follows [231, p. 70]

$$G(S) = G_0 + \tilde{M}_0^{-1} S (I + M_0^{-1} U_0 S)^{-1} M_0^{-1} = \mathcal{F}_\ell\left(J_G, S\right), \tag{2.19}$$

where the central system is

$$J_G = \begin{bmatrix} G_0 & \tilde{M}_0^{-1} \\ M_0^{-1} & -M_0^{-1} U_0 \end{bmatrix}. \tag{2.20}$$

It is emphasized in particular that the transfer function matrix from the input $\boldsymbol{s}$ to the output $\boldsymbol{r}$ in the pre-stabilized loop of Fig. 2.1b is the parameter S [234, Rem. 8]. Note that the nominal system G_0 is recovered by $S = 0$, corresponding to the ideal case assumed in the previous section.

2.2.2 Double-Youla Parameterization

Using both parameterizations of plant $G(S)$ and controller $K(Q)$, a *double-Youla* [50] parameterization results, *i. e.*, a description for the set of loops formed by a nominal plant/controller pair and the perturbations of the closed loop described by the parameter systems S and Q. Concerning the stability of the resulting loop, the nominal interconnection cancels out provided it is stable [234, 231]; thus, in order to establish robust stability under both plant and controller uncertainty, the loop (Q, S) is decisive.

Proposition 2.7 (Double-Youla parameterization [234, Th. 2.1]). Let (G_0, K_0) be a stabilizing nominal plant-controller pair, $G(S)$ be the class of plants described by (2.17)–(2.18), and $K(Q)$ be the class of controllers (2.5)–(2.6). Then $(G(S), K(Q))$ is well-posed and internally stable if and only if the feedback system (Q, S) is well-posed and stabilizing, *i. e.*, there exists $\begin{bmatrix} I & -Q \\ -S & I \end{bmatrix}^{-1} \in \mathcal{RH}_\infty$.

Proof: See [234] and [231, p. 76ff]. ■

Proposition 2.7 immediately allows to characterize a set $\mathcal{Q}$ of maximally allowed perturbations $Q \in \mathcal{Q}$ of the controller K subject to robust stability of the closed loop for some set $\mathcal{S}$ of permissible models $S \in \mathcal{S}$.

Proposition 2.8 (Stable plant/controller perturbations). Consider a set of plants $\mathcal{G}$ and a set of controllers $\mathcal{K}$ given by

$$\begin{aligned} \mathcal{G}(G_0, K_0, \gamma_\mathrm{S}) &\triangleq \{(N_0 + V_0 S)(M_0 + U_0 S)^{-1} \mid \|S\|_\infty \leq \gamma_\mathrm{S}\}, \\ \mathcal{K}(G_0, K_0, \gamma_\mathrm{Q}) &\triangleq \{(U_0 + M_0 Q)(V_0 + N_0 Q)^{-1} \mid \|Q\|_\infty \leq \gamma_\mathrm{Q}\}. \end{aligned} \tag{2.21}$$

Then all plants in $\mathcal{G}$ are stabilized by all controllers in $\mathcal{K}$ if and only if $\gamma_\mathrm{S} \cdot \gamma_\mathrm{Q} < 1$.

Proof: Application of the small-gain theorem [48] on the (Q, S) loop [231, p. 166f, p. 224]. ■

For the developments later in this thesis, it is important to note that the open-loop transfer function matrix from the input $\boldsymbol{s}$ to the output $\boldsymbol{r}$ in the closed (pre-stabilized) feedback loop of Fig. 2.1b is precisely the parameter S [231, p. 79]. Also note that the closed-loop T_{zw} is not affine anymore if $S \neq 0$ but instead becomes [161, Th. 7]

$$\begin{aligned} T_{zw} = G_{zw} - G_{zu} M_0 G_{sw} + \Big(G_{zu}(M_0 + U_0 S) + G_{zr} S \Big) \\ \cdot (I - QS)^{-1} \Big((\tilde{U}_0 + Q\tilde{M}_0) G_{yw} + G_{sw} \Big). \end{aligned} \tag{2.22}$$

Summary. When implementing controllers based on a Youla parameterization, the dual parameterization of stabilized plants is a complementary and handy tool. In this thesis, the dual parameter is not explicitly implemented as if the predominant goal were to identify the plant, for example in closed-loop system identification. Instead, while the controller parameterization is used for implementation, conceptually the dual parameter will serve for analysis of the loop. Hence, a double-parameterization approach is used in order to narrow down the search space for learning of the controller.

2.3 Controller Switching and Interpolation

The concept to switch or blend controllers is common in practice, particularly if the plant is nonlinear and the controller needs to be adapted along the operating conditions. Research on gain scheduling correspondingly has a long history, see [131, 197] and the references therein. Switching-based adaptive control schemes, in turn, employ some logic-based rule to entirely swap controllers during operation. Thus, a supervisor [82] dictates which controller is active, usually based on some decision logic given the performance signals $\boldsymbol{z}$ or some pre-processed version thereof. The piecewise constant signal $\sigma : \mathcal{T} \mapsto \mathcal{I}$ then rules which of the N_K controllers is active, where $\mathcal{I} \triangleq \{1, \ldots, N_\mathrm{K}\}$ denotes the index set and $\mathcal{T}$ the time set.

Switching or interpolation of controllers can lead to instability even if each controller separately stabilizes the system [137]. Therefore, ensuring stability under switching or interpolation of the family of controllers is crucial and a wealth of approaches exists, see for example [137, 138] and the references therein. One specific way to ensure stability when switching between linear controllers is to exploit the state-space realization of each controller as a design degree of freedom [213]: the class of all stabilizing controllers is particularly useful as reviewed next.

2.3.1 State-Space Realizations for Arbitrary Switching and Interpolation

Given that the notion of quadratic stability is similar w. r. t. arbitrary switching and convex interpolation between controllers (*cf.* Prop. A.5 in Appendix A.2), results exploiting the Youla parameterization can be found in both the literature on gain scheduling and on switching systems. A specific realization useful for interpolation is by using a construction within the class of all stabilizing controllers, *i. e.*, in terms of the stable parameter system Q. Stabilizing controller interpolation then reduces to the simpler task of interpolating stable systems [161]. In similar spirit, the so-called J-Q-interpolation constitutes an important stability preserving scheme for gain scheduling [222]. Hespanha and Morse [80] prove that a similarity transformation of the parameters Q is sufficient to construct a common quadratic Lyapunov function (CQLF), ensuring stability under *arbitrary* switching between linear controllers.

2.3.2 Switching Stabilization of Linear Switching Systems

In this section, the case is considered when also the plant is switching according to some external rule. This problem is of interest because the stabilization results of [80] for arbitrary switching between controllers for a linear system extend to the case when also the plant is switching [27]. Therefore, some results of Blanchini *et al.* [27] on switching stabilization of linear switching systems are reviewed. While more general conditions for switching stability are given in [27], only quadratic stability is considered in this thesis.

Consider the problem of stabilizing the time-varying system

$$\begin{aligned} \delta\boldsymbol{x} &= \boldsymbol{A}_{\sigma(t)}\boldsymbol{x} + \boldsymbol{B}_{\sigma(t)}\boldsymbol{u}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y} &= \boldsymbol{C}_{\sigma(t)}\boldsymbol{x}, \end{aligned} \tag{2.23}$$

whose dynamics can switch arbitrarily between N_sys modes as governed by a switching signal $\sigma : \mathcal{T} \mapsto \mathcal{I},\ \mathcal{I} = \{1, 2, ..., N_\mathrm{sys}\}$. The index $i = \sigma(t)$ is called active mode at time instant t. Let $\boldsymbol{x} \in \mathbb{R}^n$ denote the state-space vector with initial state $\boldsymbol{x}_0$, $\boldsymbol{u} \in \mathbb{R}^{n_\mathrm{u}}$ and $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{y}}$ are the input and output signals, and the set of associated switching controllers is $\mathcal{K} = \{K_1, K_2, ..., K_{N_\mathrm{sys}}\}$. This situation is visualized in Fig. 2.4.

We assume the standard case of stabilizability and detectability of all modes.

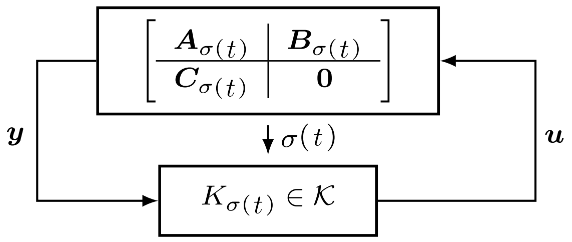

$\left[\begin{array}{c|c} \boldsymbol{A}_{\sigma(t)} & \boldsymbol{B}_{\sigma(t)} \\ \hline \boldsymbol{C}_{\sigma(t)} & \boldsymbol{0} \end{array}\right]$
$\sigma(t)$
$\boldsymbol{y}$
$\boldsymbol{u}$
$K_{\sigma(t)} \in \mathcal{K}$

Figure 2.4: Switching control of a linear switching system. The signal $\sigma(t)$ dictates the plant/controller pair that is active at time instant t.

Assumption 2.1. For all $i \in \mathcal{I}$, the pairs $(\boldsymbol{A}_i, \boldsymbol{B}_i)$ are stabilizable and $(\boldsymbol{A}_i, \boldsymbol{C}_i)$ are detectable. $\diamond$

Concerning the switching signal, it is assumed that there is no delay in communicating the currently active mode from plant to controller; moreover, non-zenoness is assumed for the continuous-time case (implicit in the discrete-time domain).

Assumption 2.2. Switching occurs uncontrolled, the number of switching instants is finite on every finite interval, and switching is instantaneous for both the plant and the controller, *i. e.*, for all time instants t, the currently active mode $\sigma(t)$ is known, but not dictated by the controller. $\diamond$

The following proposition summarizes results from [27] and provides linear matrix inequality (LMI) conditions that are efficiently solvable by semidefinite programming.

Proposition 2.9 ([27]). There exists a linear switching compensator $\mathcal{K}$ for the switching plant (2.23) assuring switching quadratic stability to the closed-loop system iff there exists symmetric positive definite matrices $\boldsymbol{P}_\mathrm{F} \in \mathbb{R}^{n \times n}$, $\boldsymbol{P}_\mathrm{L} \in \mathbb{R}^{n \times n}$, $\boldsymbol{U}_i \in \mathbb{R}^{n_\mathrm{u} \times n}$ and $\boldsymbol{Y}_i \in \mathbb{R}^{n \times n_\mathrm{y}}$ such that

- in the discrete-time case:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{P}_\mathrm{F} & (\boldsymbol{A}_i\boldsymbol{P}_\mathrm{F} + \boldsymbol{B}_i\boldsymbol{U}_i)^\top \\ \boldsymbol{A}_i\boldsymbol{P}_\mathrm{F} + \boldsymbol{B}_i\boldsymbol{U}_i & \boldsymbol{P}_\mathrm{F} \end{bmatrix} \succ 0, \\ \begin{bmatrix} \boldsymbol{P}_\mathrm{L} & (\boldsymbol{P}_\mathrm{L}\boldsymbol{A}_i + \boldsymbol{Y}_i\boldsymbol{C}_i)^\top \\ \boldsymbol{P}_\mathrm{L}\boldsymbol{A}_i + \boldsymbol{Y}_i\boldsymbol{C}_i & \boldsymbol{P}_\mathrm{L} \end{bmatrix} \succ 0; \end{aligned} \tag{2.24}$$

- in the continuous-time case:

$$\begin{aligned} \boldsymbol{A}_i\boldsymbol{P}_\mathrm{F} + \boldsymbol{P}_\mathrm{F}\boldsymbol{A}_i^\top + \boldsymbol{B}_i\boldsymbol{U}_i + \boldsymbol{B}_i^\top\boldsymbol{U}_i^\top \prec 0, \\ \boldsymbol{A}_i^\top\boldsymbol{P}_\mathrm{L} + \boldsymbol{P}_\mathrm{L}\boldsymbol{A}_i + \boldsymbol{Y}_i\boldsymbol{C}_i + \boldsymbol{C}_i^\top\boldsymbol{Y}_i^\top \prec 0. \end{aligned} \tag{2.25}$$

Note that, just as in the case of a single plant under switching control [80], the controller family $\mathcal{K}$ cannot be arbitrary, but must be suitably realized. If the LMIs (2.24) respectively (2.25) are solvable, a set of stabilizing compensators $\mathcal{K}$ is given by the observer-based controllers $\mathcal{J} = \{J_1, \dots, J_{N_\mathrm{sys}}\}$ constructed as

$$J_i : \begin{cases} \delta\hat{\boldsymbol{x}} = & (\boldsymbol{A}_i + \boldsymbol{L}_i\boldsymbol{C}_i + \boldsymbol{B}_i\boldsymbol{F}_i)\hat{\boldsymbol{x}} - \boldsymbol{L}_i\boldsymbol{y} + \boldsymbol{B}_i\boldsymbol{s} \\ \boldsymbol{u} = & \boldsymbol{F}_i\hat{\boldsymbol{x}} + \boldsymbol{s}, \end{cases} \tag{2.26}$$

where

$$\boldsymbol{F}_i = \boldsymbol{U}_i \boldsymbol{P}_{\mathrm{F}}^{-1} \quad \text{and} \quad \boldsymbol{L}_i = \boldsymbol{P}_{\mathrm{L}}^{-1} \boldsymbol{Y}_i. \tag{2.27}$$

A key result in [27] is that the set of all switching quadratically stabilizing compensators is given by an observer-based pre-compensator $\mathcal{J}$ and an input injection $\boldsymbol{s}$, in coherent extension of the LTI theory [269]. With the switching pre-compensator $J_{\sigma(t)}$ in place, the estimation error $\boldsymbol{r}(k) = \boldsymbol{C}_{\sigma(t)}\hat{\boldsymbol{x}}(k) - \boldsymbol{y}(k)$ is fed into a proper stable switched system $Q_{\sigma(t)} \in \mathcal{Q} = \{Q_1, Q_2, \dots, Q_{N_{\mathrm{sys}}}\}$ to determine

$$\boldsymbol{s}(t) = Q_{\sigma(t)} \boldsymbol{r}(t). \tag{2.28}$$

In order to ensure stability of the overall loop for arbitrary switching σ, each of the parameters

$$Q_i = \left[\begin{array}{c|c} \boldsymbol{A}_{\mathrm{Q},i} & \boldsymbol{B}_{\mathrm{Q},i} \\ \hline \boldsymbol{C}_{\mathrm{Q},i} & \boldsymbol{D}_{\mathrm{Q},i} \end{array}\right]$$

must be realized such that set of systems $\mathcal{Q}$ is switching stable [27]. Corresponding similarity transformations always exist.

Summary. Taking advantage of the parameterization of stabilizing controllers, realizations of linear controllers can be constructed that preserve quadratic stability even when blending controllers or switching arbitrarily among them. Assuming full knowledge about the switching signal, the scheme extends to switching linear systems in analogous fashion, *i. e.*, using an appropriate observer-based realization.

2.4 Adaptive-Q and Learning-Q Control

Usage of the Q-parameterization also has a long history in adaptive and learning control. In this section, we briefly summarize the main ideas of the adaptive-Q and learning-Q methods dating back to the 1990s; for a more comprehensive treatise, see [231] and the references therein.

Principle of adaptive-Q control. *Adaptive-Q* control [153, 233, 251, 231, 232] is easiest understood as an online variant of Q-Design [28], *i. e.*, taking advantage of the affine parameterization for numerical optimization of the controller. The underlying philosophy is to first robustly stabilize the plant under control and then improve performance online by limited adaptation within the parameterization of all stabilizing controllers. Direct adaptive-Q control is therefore most suitable when model uncertainty is limited, but an existing controller must be re-tuned online because the design criterion is altered or the characteristics of the reference signal is not known during the design stage. Otherwise, indirect [234] and iterative approaches exist as well, including a plant uncertainty model characterized by a dual Youla operator. For nonlinear plants, adaptive-Q methods were developed to enhance the robustness and performance of a linear time-varying feedback controller along the trajectory $(\boldsymbol{u}^{\star}, \boldsymbol{x}^{\star})$ of a smooth nonlinear system [94].

To briefly illustrate the direct adaptive-Q approach, consider in discrete time the standard observer-based parameterization shown in Fig. 2.2. Let Q be realized as a stable dynamic system with state $\boldsymbol{x}_\mathrm{Q} \in \mathbb{R}^{n_\mathrm{q}}$ and adaptively adjusted output

$$Q : \begin{cases} \boldsymbol{x}_\mathrm{Q}(k+1) = \boldsymbol{A}_\mathrm{Q}\boldsymbol{x}_\mathrm{Q}(k) + \boldsymbol{B}_\mathrm{Q}\boldsymbol{r}(k),\ \boldsymbol{x}_\mathrm{Q}(0) = \boldsymbol{x}_{\mathrm{Q},0}, \\ \qquad \boldsymbol{s}(k) = \Xi(k)\boldsymbol{x}_\mathrm{Q}(k). \end{cases} \tag{2.29}$$

The matrix $\boldsymbol{A}_\mathrm{Q} \in \mathbb{R}^{n_\mathrm{q}\times n_\mathrm{q}}$ is a design parameter and $\Xi(k) \in \mathbb{R}^{n_\mathrm{u}\times n_\mathrm{q}}$ is the adaptation matrix with variable parameters. Due to the affine form (2.12) of the closed-loop in the parameter Q, choosing a suitable law to determine $\Xi(k)$ online, the system (2.29) constitutes an (adaptive) notch filter in the closed loop. Therefore, the control system performance can be enhanced using a suitable generalized plant setup. Examples of practical applications of adaptive-Q control include suppression of vibration caused by unknown disturbances in rotor/magnetic bearing systems [41] and active jitter reduction in precision optical systems [147]. Kalyanam and Tsao [102] give an elaborate experimental case study on the application of adaptive-Q control in a hard-disk drive track-following servo problem. Luo *et al.* [142] more recently report the real-time optimization of a motor controller using, in essence, an adaptive-Q control approach.

Learning-Q control. Leveraging the Q-parameterization in the context of learning dates back to the 1990s, when Moore and co-workers [94, 95] coined the term *learning-Q* control[1] for the approach. The classic learning-Q method refers to an adaptive-Q approach along predefined trajectories of a nonlinear system; however, the function approximation used in the filter system Q is designed to depend on the current state. Some form of functional learning is then used to adapt the parameters of the approximation architecture in the filter system: "The major objective of the learning-Q method is to learn from one trajectory, or set of trajectories information which will enhance the control performance for a new trajectory." [95]

Summary. The methods investigated in this thesis, *i. e.*, construction of a parameterization of stabilizing controllers and performance enhancement by means of RL, constitute an *advanced form of a learning-Q* approach, with a particular emphasis on robotics.

2.5 Conclusion

In this chapter, the methodological foundations for the developments in the remainder of the thesis were reviewed. While the construction of stabilizing controllers is typically stated in a coprime factor (frequency-domain) description, it is more convenient to work directly in state space or to construct suitable factors in state space for practical implementations. The observer-based interpretation of stabilizing controllers is well-known, and additionally the derivation of a parameterization based on static state feedback was shown. In the next chapter, a similar approach will be used to derive a simple parameterization to implement variable state feedback controllers, for instance for the purpose of robot control.

[1]Not to be confused with the famous RL algorithm *Q-learning* [252], *cf.* Sec. 5.1.

2.6 Bibliographical Notes

In addition to the applications reviewed above, the Q-parameterization is ubiquitously used in control theory, *e. g.*, in robust control [269, 51], computer aided (optimization-based) controller design [28], gain scheduling [221, 184, 23], model predictive [237], hybrid [80], and adaptive [231, 126] control. Vidyasagar's classic book [246] on the factorization approach provides an algebraic view on control system synthesis and stabilizing controllers. In the context of robotics, the factorization approach has been introduced to the robust control of robots in [218] in a one-degree-of-freedom and in [226] for two-degree-of-freedom controller design and is nowadays covered in textbooks [132, 127] as well. Anderson's survey [6] constitutes an excellent overview of the primary and dual Youla parameterizations up to the 1990s. In this section, we briefly address some relevant work that has appeared in the wealth of related literature since then.

The parameterization has been recognized as a useful approach to explore arbitrarily switching and interpolated controllers. Hespanha and Morse [80] leverage the parameterization for hybrid control and prove that a similarity transformation of the parameters Q is sufficient to construct a CQLF, ensuring stability under arbitrary switching between linear controllers. As summarized in Sec. 2.3.2 above, Blanchini *et al.* [27] extend the result of [80] to the case when also the plant is switching. Parameterizations also exist for polytopic linear parameter-varying (LPV) systems [256, 185], allowing for example to construct switched LPV controllers [24]. Under certain conditions, gain scheduling is admissible in the parameterization even on endogenous signals [184]. Also based on [80], a hybrid regulator is proposed in [40] to enforce output regulation using an estimation of the frequencies in an unknown exogenous system. Several structures to implement switching controllers based on the Youla parameterization are compared in [156]. Stability under switching, however, does not yet ensure performance. This drawback is resolved by the parameterization of Hencey and Alleyne [78] that preserves suboptimal $\mathcal{H}_\infty$ performance and suppresses transient peaks effectively. Similarly, this result was shown to extend to switching linear systems [257] as well.

The parameterizations above all consider specific factorizations or construct the central controller as needed. However, if an existing controller must remain implemented, parameterizations can also be built upon only the terminal connections [239]. This requirement will also be imposed on the parameterization developed in the next chapter. Thereby, one can also set up a parameterization without the need to factorize the initial stabilizing controller K_0 explicitly, a fact exploited in the framework of so-called plug-and-play control [16]. An architecture for Youla-like parameterizations based on reduced order models is presented in [163]. Another important result is the parameterization over stabilizing structured controllers, allowing to employ convex optimization in decentralized problems; see [196] and the references therein. While the Youla parameterization has also been used to derive anti-windup controllers for some time, the relation of the Youla parameterization itself compared to established, well-known anti-windup structures has been very recently explored [162], allowing to include an anti-windup element directly via the parameterization. Finally, it is noted that not all the parameterizations above yield closed-loop maps that are affine in Q.

The dual parameterization is most known for its use in closed-loop system identification in the Hansen scheme [71]. Parameterizations of the dual form nowadays have been recognized to play a key role to reduce conservatism in the robust design of advanced motion control systems, see [168]. The dual parameterization is also used for active fault diagnosis [223], closed-loop system identification of LPV systems [17], and multiple-model adaptive control [18]. A constrained model predictive control scheme based on the primary and dual Youla parameters is proposed in [237]. A robust switching control scheme for scalar plants is presented in [14], leveraging the dual Youla parameter to show that switching among robust controllers is possible while ensuring robust stability under arbitrary switching.

Finally, on the one hand, it can be shown that all stabilizing controllers for a linear plant can be constructed using the observer-based structure [2]. On the other hand, some of the stabilizing parameterizations listed above actually cannot recover *all* stabilizing controllers. In the context of this thesis, this difference is largely irrelevant. In order to optimize and adapt the parameter system Q using learning methods, in practice, a restriction of the search space has to be made in order to limit the complexity of the learning problem.

3 Adaptive Control by Parameterization and Multiple Modes

The previous chapter set the foundation for the contributions of this thesis by providing a concise background on the parameterization of stabilizing controllers and its use in control. This chapter presents novel contributions to adaptive control methods that leverage the parameterization. The overarching approach is to consider multiple modes in the parameterization, first only in the controller and later also in the plant. This chapter presents results previously published in [58, 59] and is structured in two larger parts.

In the first part of this chapter in Sec. 3.1, a feedback controller architecture is derived to simplify arbitrary interpolation by means of the controller parameterization with a special focus on applicability to robotic hardware as required in part III of the thesis. After motivating the developments by the variable impedance control problem in Sec. 3.1.1 and comparing to other state-of-the-art parameterizations in Sec. 3.1.2, we proceed to give a formal problem statement in Sec. 3.1.3. The main result of this section, the parameterization and the choice of parameters are presented in Sec. 3.1.4–Sec. 3.1.6. In Sec. 3.1.7 the result is discussed in relation to previous parameterizations before we return to the active variable impedance control problem in Sec. 3.1.8 as an example for the utility of the method.

In Sec. 3.2, the second part of this chapter, the starting point is different in that now also the plant is assumed to be switching among several modes. This section is to explore the synergy between the control of switching linear systems and the adaptive-Q method. To this end, in Sec. 3.2.1 the motivation for this problem is outlined before formalizing it in Sec. 3.2.2. The major result of this second part of this chapter is given in Sec. 3.2.3, a design methodology to enhance the control of switching linear systems with adaptive fine-tuning of the distinct controller modes. The approach is illustrated by simulation of an example system in Sec. 3.2.4.

With Sec. 3.3 we conclude by giving an outlook on possible future work building on the methods presented in this chapter.

3.1 Arbitrary Interpolation of State Feedback Controllers

Certain control systems demand for switching or interpolation between state feedback. This section is to make the parameterization approach easily accessible for interpolated control problems where state information is available. Before we proceed to the general problem setting, the active variable impedance control problem is reviewed which motivates the development of such a parameterization. Active impedance control requires a robot manipulator to achieve by regulation some desired stiffness/damping characteristics, respectively, a specific mechanical impedance. Hence, as will be shown next, to implement variable impedance it is required to vary the feedback gains.

3.1.1 The Active Variable Impedance Control Problem

Consider a torque-controlled, fully actuated robot manipulator with revolute joints. The goal of (Cartesian) impedance control [84] is to shape the mechanical impedance of the manipulator such that the response of the end effector to external wrench $\boldsymbol{f}_\mathrm{e} \in \mathbb{R}^6$ becomes similar to that of a mass-spring-damper system:

$$\boldsymbol{M}(\ddot{\boldsymbol{x}}_\mathrm{C,d} - \ddot{\boldsymbol{x}}_\mathrm{C}) + \boldsymbol{K}_\mathrm{D}(\dot{\boldsymbol{x}}_\mathrm{C,d} - \dot{\boldsymbol{x}}_\mathrm{C}) + \boldsymbol{K}_\mathrm{P}(\boldsymbol{x}_\mathrm{C,d} - \boldsymbol{x}_\mathrm{C}) = \boldsymbol{f}_\mathrm{e}. \tag{3.1}$$

The positive definite matrices $\boldsymbol{M}, \boldsymbol{K}_\mathrm{D}, \boldsymbol{K}_\mathrm{P} \in \mathbb{R}^{6\times6}$ denote the desired inertia, damping, and stiffness, and $\boldsymbol{x}_\mathrm{C,d} \in \mathbb{R}^6$ is the virtual reference trajectory in the generalized coordinates $\boldsymbol{x}_\mathrm{C} \in \mathbb{R}^6$. For simplicity, only the case is considered where the inertia of the manipulator is not affected; in this case, the impedance control can be implemented without measurement of external forces. In order to achieve (3.1), it is then sufficient to feed the deviation of the states $\boldsymbol{x} = \left[\boldsymbol{x}_\mathrm{C}^\top, \dot{\boldsymbol{x}}_\mathrm{C}^\top\right]^\top$ back as

$$\boldsymbol{f}_\tau = \left[\boldsymbol{K}_\mathrm{P}, \boldsymbol{K}_\mathrm{D}\right](\boldsymbol{x}_\mathrm{d} - \boldsymbol{x}) = \left[\boldsymbol{K}_\mathrm{P}, \boldsymbol{K}_\mathrm{D}\right]\boldsymbol{x}_\mathrm{d} - \left[\boldsymbol{K}_\mathrm{P}, \boldsymbol{K}_\mathrm{D}\right]\boldsymbol{x} \triangleq \boldsymbol{u}_\mathrm{ff} + \boldsymbol{u}_\mathrm{fb}, \tag{3.2}$$

where $\boldsymbol{f}_\tau \in \mathbb{R}^6$ is the input vector, realized via the Jacobian $\boldsymbol{J}$ of the manipulator by the controlled joint torques $\boldsymbol{\tau} = \boldsymbol{J}^\top \boldsymbol{f}_\tau$. For a *variable* desired impedance, $\boldsymbol{K}_\mathrm{D}(t, \boldsymbol{x}), \boldsymbol{K}_\mathrm{P}(t, \boldsymbol{x})$ are chosen depending on time- or even state and the feedback term $\boldsymbol{u}_\mathrm{fb}$ in (3.2) becomes an instance of *interpolated state feedback*.

It is well-known [197] that standard linear analyses, *e. g.*, assessment of the closed-loop eigenvalues over time, are in general not sufficient to conclude stability of the resulting closed loop system, as shown exemplarily in Remark 3.1. This concern is nonetheless frequently ignored [32, 68, 121, 188] in favor of simplicity in implementation: it is mitigated by using demonstrations to bias towards admissible behavior [121, 32, 39]. High damping on hardware and slow variations as in [136, 157] in practice further alleviate the potential for instabilities caused by ad-hoc gain scheduling. Current solutions which consider stability are tailored specifically to the robotics problem domain [54, 108] or do not provide a synthesis method [123].

Remark 3.1 *(Eigenvalue assessment in the linear time-varying (LTV) case).* In general, stability of the matrix $\boldsymbol{A}(t)$ is not sufficient to check uniform asymptotic stability for LTV systems. Consider the following counterexample from [103]:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_\mathrm{V}(t)\boldsymbol{x}(t), \quad \text{with } \boldsymbol{A}_\mathrm{V}(t) = \begin{bmatrix} -1 + 1.5\cos^2(t) & 1 - 1.5\sin(t)\cos(t) \\ -1 - 1.5\sin(t)\cos(t) & -1 + 1.5\sin^2(t) \end{bmatrix}.$$

Although the eigenvalues are $\mathrm{eig}(\boldsymbol{A}_\mathrm{V}(t)) = -\frac{1}{4} \pm \frac{1}{4}\sqrt{7}\mathrm{j}$ for all t, there are initial states $\boldsymbol{x}_0$ for which the solution $\boldsymbol{x}(t) = \begin{bmatrix} e^{0.5t}\cos(t) & e^{-t}\sin(t) \\ -e^{0.5t}\sin(t) & e^{-t}\cos(t) \end{bmatrix}\boldsymbol{x}_0$ is unbounded. $\triangleleft$

3.1.2 Comparison to the State-of-the-Art Parameterizations

With these issues in mind, we report a novel interpolation scheme for state feedback controllers addressing the following requirements.

R1) The closed loop must be stable under arbitrary interpolation or switching, and the state feedback controllers must be recovered in the design points.

R2) The scheme must be based on a static state feedback controller and use the terminal connections, *i. e.*, the initial state feedback controller cannot be replaced[1].

R3) The interpolated controller must be simple to implement and allow for a transparent interpretation of its components.

One common drawback of the parameterizations reviewed in Chap. 2 is a potentially high dynamic order. In addition, the complexity of the schemes seems to restrict the scope of applications. In this section, a complementary parameterization is provided, *i. e.*, an architecture fulfilling the requirements R1–R3. To this end, the general YK switching framework is specialized to the state feedback case and the dynamic parameters Q are derived that recover state feedback behavior in the design points. In contrast to many works that do not provide guidelines how to systematically exploit the freedom in the choice of the coprime factorization, [231]–[80],[185], [239]–[156], we also propose a design for the free parameter matrix to simplify the structure and to allow for a transparent and intuitive interpretation of the scheme. In this section, the formulae for all design steps involved are provided in both continuous- and discrete-time domains.

3.1.3 Formal Problem Setting

Consider the nominal system G with measurable state $\boldsymbol{x} \in \mathbb{R}^{n}$, control input $\boldsymbol{u} \in \mathbb{R}^{n_\mathrm{u}}$ and output $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{y}}$

$$G : \begin{cases} \delta\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_2\boldsymbol{u}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{z} = \boldsymbol{C}_1\boldsymbol{x} + \boldsymbol{D}_{12}\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{x}, \end{cases} \tag{3.3}$$

such that some desired performance is expressed by the quantities $\boldsymbol{z} \in \mathbb{R}^{n_\mathrm{z}}$. Assuming stabilizability of $(\boldsymbol{A}, \boldsymbol{B}_2)$, let a set

$$\mathcal{K} = \{K_1, K_2, \ldots, K_{N_\mathrm{K}}\} \tag{3.4}$$

of static state feedback controllers $K_i \; : \; \boldsymbol{u} \; = \; \boldsymbol{D}_{\mathrm{K},i}\boldsymbol{x}, \; \boldsymbol{D}_{\mathrm{K},i} \in \mathbb{R}^{n_\mathrm{u} \times n}$ be given such that $\boldsymbol{A} + \boldsymbol{B}_u\boldsymbol{D}_{\mathrm{K},i}$ is stable for all $i = 1, \ldots, N_\mathrm{K}$.

[1]The user interface of some robotic hardware only allows to modify the control torque by adding to some implemented, ever-present state feedback. One robot that can be considered in this category is the KUKA LWR IV+ that will serve as experimental platform in the case study in Chap. 7.

For the interpolated controller $K(\boldsymbol{\alpha}(t))$, let the variable $\boldsymbol{\alpha}(t) \in \mathbb{R}^{N_\mathrm{K}}$ describe the extent of how much each controller of the family $\mathcal{K}$ contributes at instant t. We allow for the set of piecewise continuous *arbitrary interpolation signals* [77]

$$\mathcal{A} = \left\{ \boldsymbol{\alpha}(t) : \mathbb{R}_0^+ \mapsto [0,1]^{N_\mathrm{K}} \;\middle|\; \textstyle\sum_{i=1}^{N_\mathrm{K}} \alpha_i = 1,\ \alpha_i \geqslant 0 \right\}, \tag{3.5}$$

covering both arbitrarily fast switching and blending of controllers. The controller interpolation criteria from [77] are adopted accordingly to characterize admissible controllers.

Definition 3.1 *(Admissible interpolated state feedback).* Given a set $\mathcal{K}$ of local state feedback controllers, an admissible interpolated controller $K(\boldsymbol{\alpha})$ satisfies the following controller interpolation criteria [77]:

1. $K(\boldsymbol{\alpha})$ is stabilizing for all $\boldsymbol{\alpha} \in \mathcal{A}$.
2. $K(\boldsymbol{\alpha}) \sim K_i$ for constant $\alpha_i = 1$, $\forall i = 1, \dots, N_\mathrm{K}$.
3. $K(\boldsymbol{\alpha})$ is a continuous function of $\boldsymbol{\alpha}(t)$. $\diamond$

We are now ready to give the formal problem statement for arbitrary interpolation of state feedback controllers subject to requirements R1–R3.

Problem 3.1 *(Arbitrary Interpolation of State Feedback).* Consider an initial controller $K_0 :$ $\boldsymbol{u} = \boldsymbol{D}_{\mathrm{K},0}\boldsymbol{x}$ such that $\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{D}_{\mathrm{K},0}$ is stable. Given a set $\mathcal{K}$ of static state feedback controllers, construct a dynamic augmentation of K_0 such that $K(\boldsymbol{\alpha})$ is an admissible interpolated state feedback controller and $\boldsymbol{D}_{\mathrm{K},0}$ is the only free design parameter. $\diamond$

3.1.4 Parameterization for Arbitrary Interpolation

In general, under arbitrary interpolation of the set of feedback gains (3.4) by

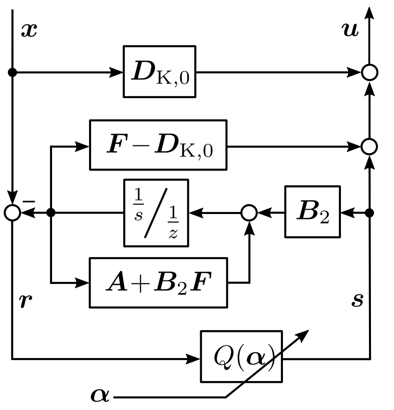
$$K(\boldsymbol{\alpha}) : \quad \boldsymbol{u} = \left(\sum_{i=1}^{N_\mathrm{K}} \alpha_i \boldsymbol{D}_{\mathrm{K},i} \right) \boldsymbol{x}, \tag{3.6}$$

stability cannot be guaranteed. In order to solve Prob. 3.1, first the realization of an admissible interpolated state feedback controller is derived in Sec. 3.1.4 before we discuss the choice of parameters in Sec. 3.1.6 to further simplify the scheme.

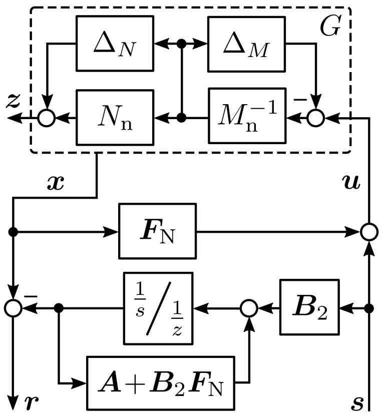
Theorem 3.1 (Interpolation of State Feedback Controllers). Consider an LTI plant (3.3), a stabilizing state feedback controller $\boldsymbol{u} = \boldsymbol{D}_{\mathrm{K},0}\,\boldsymbol{x}$, a set of stabilizing static controllers (3.4), and a matrix $\boldsymbol{F} \in \mathbb{R}^{n_\mathrm{u} \times n}$ such that $\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F}$ is stable. An admissible interpolated controller $K(\boldsymbol{\alpha})$ is given by interconnecting

$$\begin{aligned} J : &\begin{cases} \delta \boldsymbol{x}_\mathrm{J} = (\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F})\boldsymbol{x}_\mathrm{J} + \boldsymbol{B}_2\boldsymbol{s}, \quad \boldsymbol{x}_\mathrm{J}(0) = \boldsymbol{0}, \\ \boldsymbol{u} = \boldsymbol{D}_{\mathrm{K},0}\,\boldsymbol{x} + (\boldsymbol{F} - \boldsymbol{D}_{\mathrm{K},0})\boldsymbol{x}_\mathrm{J} + \boldsymbol{s}, \\ \boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x}_\mathrm{J} \end{cases} \\ \text{with} \quad & \quad \boldsymbol{s} = Q(\boldsymbol{\alpha})\boldsymbol{r}, \end{aligned} \tag{3.7}$$

where the interpolated system $Q(\boldsymbol{\alpha})$ of parameters $\mathcal{Q} = \{Q_i \,|\, Q_i \in \mathcal{RH}_\infty, i = 1, \dots, N_\mathrm{K}\}$ is realized such that all $Q \in \mathcal{Q}$ share a CQLF, *i. e.*,



(a) Structure of interpolation based on static state feedback controller $D_{K,0}$



(b) Simplified structure and model of the nominal closed state feedback loop contained in J

Figure 3.1: Proposed parameterization for arbitrary interpolation of state feedback controllers.

$$\begin{aligned} &\exists\, P_Q \in \mathbb{R}^{n\times n},\ P_Q = P_Q^\top \succ 0 : \\ &\quad \forall x \neq 0 : V(x) = x^\top P_Q x > 0 \text{ and } \forall \alpha \in \mathcal{A} : && (3.8) \\ &\text{(continuous-time)} \quad P_Q A_Q(\alpha) + A_Q^\top(\alpha) P_Q \prec 0, && (3.9a) \\ &\text{(discrete-time)} \quad A_Q(\alpha)^\top P_Q A_Q(\alpha) - P_Q \prec 0. && (3.9b) \end{aligned}$$

The dynamic parameters Q_i corresponding to the local state feedback controllers K_i are given by

$$Q_i \colon \begin{cases} \delta x_Q = (A + B_2 D_{K,i}) x_Q + B_2 \left(D_{K,0} - D_{K,i}\right) r, \\ x_Q(0) = \mathbf{0}, \\ s_i = \left(F - D_{K,i}\right) x_Q + \left(D_{K,i} - D_{K,0}\right) r. \end{cases} \tag{3.10}$$

Proof: The proof is provided in Appendix B.4. ■

The interpolated $Q(\alpha)$ can be realized to satisfy (3.8)–(3.9) by the standard procedures summarized in the next section.

Discussion. In order to construct an admissible P_Q for (3.8), all Q_i in (3.10) have to be stable. Therefore, the requirement that all $D_{K,i}$ must stabilize the plant model cannot be relaxed. The order of the resulting controller is $2n$ or $(N_K + 1)n$, depending on the scheme to implement the interpolated $Q(\alpha)$ according to (3.12) or (3.11), respectively. Requirement R1 is fulfilled directly given Thm. 3.1. As the original controller $D_{K,0}$ stays in place and the dynamic augmentation to realize J only requires to measure x and to add to u, R2 is satisfied as well. The structure of the interpolation scheme is depicted in Fig. 3.1a.

3.1.5 Implementation of Q-filter

Two approaches are predominant in the literature to implement the interpolated system $Q(\boldsymbol{\alpha})$ subject to (3.8)–(3.9).

Implementation as local Q-network. It is straightforward to interpolate the output of stable parallel systems $Q_i \in \mathcal{RH}_\infty$, *i. e.* $Q_{\mathrm{LQN}}(\boldsymbol{\alpha}) = \sum_{i=1}^{N_\mathrm{K}} \alpha_i Q_i \in \mathcal{RH}_\infty$. This is sometimes termed local Q-network (LQN) [184, 78]. Formally, a realization is

$$\boldsymbol{A}_\mathrm{Q} = \mathrm{diag}(\boldsymbol{A}_{\mathrm{Q},1}, \dots, \boldsymbol{A}_{\mathrm{Q},N_\mathrm{K}}), \quad \boldsymbol{B}_\mathrm{Q} = \left[\boldsymbol{B}_{\mathrm{Q},1}^\top, \dots, \boldsymbol{B}_{\mathrm{Q},N_\mathrm{K}}^\top\right]^\top, \tag{3.11a}$$

$$\boldsymbol{C}_\mathrm{Q} = [\alpha_1 \boldsymbol{C}_{\mathrm{Q},1}, \dots, \alpha_{N_\mathrm{K}} \boldsymbol{C}_{\mathrm{Q},N_\mathrm{K}}], \boldsymbol{D}_\mathrm{Q} = \textstyle\sum_{k=1}^{N_\mathrm{K}} \alpha_i \boldsymbol{D}_{\mathrm{Q},i}\,, \tag{3.11b}$$

yielding an admissible $\boldsymbol{P}_\mathrm{Q}$ in (3.8) since $\boldsymbol{A}_\mathrm{Q}$ is a constant block diagonal matrix of stability matrices.

Implementation with shared states. The interpolated controller can also be implemented as a polytopic LPV system. By a similarity transformation of all $\boldsymbol{A}_{\mathrm{Q},i}$, a common Lyapunov matrix $\boldsymbol{P}_\mathrm{Q} = \boldsymbol{S}_\mathrm{Q}^\top \boldsymbol{S}_\mathrm{Q} \in \mathbb{R}^{n\times n}$ exists as first shown in the switching literature [80, App. A.1]. As $Q_i \in \mathcal{RH}_\infty$ by construction, there exist associated $\boldsymbol{P}_{\mathrm{Q},i}$ obtained by solving (3.9) with $\alpha_i = 1, \forall i = 1, \dots, N_\mathrm{K}$. Denote by nonsingular $\boldsymbol{S}_{\mathrm{Q},i}$ a Cholesky factorization such that $\boldsymbol{S}_{\mathrm{Q},i}^\top \boldsymbol{S}_{\mathrm{Q},i} = \boldsymbol{P}_{\mathrm{Q},i}$. Then, $Q(\boldsymbol{\alpha})$ can be realized as a system of order n as

$$Q_{\mathrm{LPV}}(\boldsymbol{\alpha}) : \left[\begin{array}{c|c} \sum_{k=1}^{N_\mathrm{K}} \alpha_i \bar{\boldsymbol{A}}_{\mathrm{Q},i} & \sum_{k=1}^{N_\mathrm{K}} \alpha_i \bar{\boldsymbol{B}}_{\mathrm{Q},i} \\ \hline \sum_{k=1}^{N_\mathrm{K}} \alpha_i \bar{\boldsymbol{C}}_{\mathrm{Q},i} & \sum_{k=1}^{N_\mathrm{K}} \alpha_i \bar{\boldsymbol{D}}_{\mathrm{Q},i} \end{array}\right], \tag{3.12}$$

where $\bar{\boldsymbol{A}}_{\mathrm{Q},i} \triangleq \boldsymbol{S}_\mathrm{Q}^{-1} \boldsymbol{S}_{\mathrm{Q},i} \boldsymbol{A}_{\mathrm{Q},i} \boldsymbol{S}_{\mathrm{Q},i}^{-1} \boldsymbol{S}_\mathrm{Q}$, $\bar{\boldsymbol{B}}_{\mathrm{Q},i} \triangleq \boldsymbol{S}_\mathrm{Q}^{-1} \boldsymbol{S}_{\mathrm{Q},i} \boldsymbol{B}_{\mathrm{Q},i}$, $\bar{\boldsymbol{C}}_{\mathrm{Q},i} \triangleq \boldsymbol{C}_{\mathrm{Q},i} \boldsymbol{S}_{\mathrm{Q},i}^{-1} \boldsymbol{S}_\mathrm{Q}$, and $\bar{\boldsymbol{D}}_{\mathrm{Q},i} \triangleq \boldsymbol{D}_{\mathrm{Q},i}$.

3.1.6 Choice of Parameters

Free parameters in the parameterization (3.7)–(3.10) are the initial static controller $\boldsymbol{D}_{\mathrm{K},0}$ and the virtual gain $\boldsymbol{F}$ which is used to construct the coprime factorizations (B.11) of the plant and controllers. In general, these gains affect the transient behavior under varying $\boldsymbol{\alpha}$ and are a degree of freedom to be chosen by the designer.

One specific choice is particularly beneficial and key to satisfy requirement R3. By choosing the virtual gain $\boldsymbol{F}$ equal to the static initial controller,

$$\boldsymbol{F} = \boldsymbol{D}_{\mathrm{K},0}, \tag{3.13}$$

the structure of the generator system (3.7) is further simplified. The control input is then simply $\boldsymbol{u} = \boldsymbol{D}_{\mathrm{K},0}\boldsymbol{x} + \boldsymbol{s}$. Note that the dynamic augmentation in (3.7) models the effect of the filtered signal $\boldsymbol{s}$ on the *closed* controlled nominal loop. Therefore, the principle of function of the parameterization in Fig. 3.1 becomes very transparent.

It remains to design the gain $\boldsymbol{D}_{\mathrm{K},0}$. The most obvious choice is to take some $\boldsymbol{D}_{\mathrm{K},0} \in \mathcal{K}$. However, we propose to employ a specific LQR gain.

Corollary 3.1 (Choice of initial gain). Consider the solution $\boldsymbol{F}_{\mathrm{N}}$ to the standard linear quadratic regulator (LQR) problem of designing $\boldsymbol{u} = \boldsymbol{F}_{\mathrm{N}}\boldsymbol{x}$ such that the continuous- or the discrete-time cost functional $\int_0^\infty \boldsymbol{z}^\top \boldsymbol{z} + \boldsymbol{u}^\top \boldsymbol{u}\,\mathrm{d}t$ respectively $\sum_0^\infty \left(\boldsymbol{z}^\top \boldsymbol{z} + \boldsymbol{u}^\top \boldsymbol{u}\right)$ is minimized. Then, choosing $\boldsymbol{D}_{\mathrm{K},0} = \boldsymbol{F}_{\mathrm{N}}$ yields a central controller which is maximally robust w. r. t. normalized coprime factor uncertainty $\Delta_N, \Delta_M \in \mathcal{RH}_\infty$, *i. e.*, the robust stability margin of the nominal loop is maximized. □

Proof: The central controller of the parameterization in Thm. 3.1 is $\mathcal{F}_\ell\,(J, Q = 0)$. By construction, $\boldsymbol{F}$ is stabilizing; hence, (3.7) reduces to $\mathcal{F}_\ell\,(J, 0) = \boldsymbol{D}_{\mathrm{K},0}$. The finding of $\boldsymbol{u} = \boldsymbol{F}_{\mathrm{N}}\boldsymbol{x}$ being a maximally robust state feedback controller for the plant $G = (N_{\mathrm{n}} + \Delta_N)\,(M_{\mathrm{n}} + \Delta_M)^{-1}$ with normalized right coprime factor uncertainty is due to [110]. ■

The resulting generator system of the proposed parameterization is visualized in Fig. 3.1b.

3.1.7 Relation to Other Parameterizations

Some remarks are in order so as to put the proposed parameterization in context to the literature.

Given R2, parameterization (3.7)–(3.10) can be seen as an instance of those in [239] specialized to state feedback. Compared to the classic parameterization based on a state-estimate controller [269, 231], the order of the interpolated controller $K(\boldsymbol{\alpha})$ according to Theorem 3.1 is lower, as the order of the elements in $\mathcal{Q}$ for controller recovery of a family $\mathcal{K}$ of desired *static* controllers is higher than n when using an observer-based central controller. By (3.13), the system J in (3.7) reduces to the generator systems utilized in [195] and [150] for purposes other than interpolation. In particular, [195] is relevant w. r. t. achievable robustness as detailed in the following remark.

Remark 3.2 *(On Robustness)*. The generator system J of (3.7) can be used to construct the set of $\mathcal{H}_2$-optimal controllers— in this case, some specific $\boldsymbol{D}_{\mathrm{K},0} = \boldsymbol{F}$ and certain $Q \in \mathcal{RH}_2$ come into place [195, Th. 1]. By centering the parameterization on $\boldsymbol{F}_{\mathrm{N}}$ according to Corollary 3.1 as depicted in Fig. 3.1b, one might similarly parameterize a set of sub-optimal coprime factor uncertainty robust controllers. To this end, just as in the more general output feedback case [148], an appropriate restriction of $\|Q_i\|_\infty$ would have to be enforced. The requirement R1 of local controller recovery conflicts, however, as $\boldsymbol{D}_{\mathrm{K},i}$ is only assumed to be stabilizing and consequently $\|Q_i\|_\infty$ becomes arbitrarily large. ◁

3.1.8 Illustrative Study: The Variable Impedance Control Problem

To illustrate the utility of the parameterization, let us revisit the exemplary variable impedance control problem from [54, 123] with a single translational degree of freedom. The system consists of a mass M of nominally $M_{\mathrm{nom}} = 10\,\mathrm{kg}$ in free motion. For brevity, all quantities are normalized to SI units in what follows. Consequently, as a plant model we have $G_{yu}(s) = \frac{1}{10}\,s^{-2}$ and a minimal realization with $\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{B}_2 = [0, 0.1]^\top$ is taken. The initial state is $\boldsymbol{x}(0) = [x(0), \dot{x}(0)]^\top = [10, 0]^\top$, the virtual reference trajectory is $x_{\mathrm{d}}(t) = 10\sin(0.1t)$ and the desired variable stiffness and damping are

$$K_{\mathrm{P}}(t) = 12 + 10\sin(t) \triangleq K_{\mathrm{c}} + K'(t), \quad K_{\mathrm{D}}(t) = 1. \tag{3.14}$$

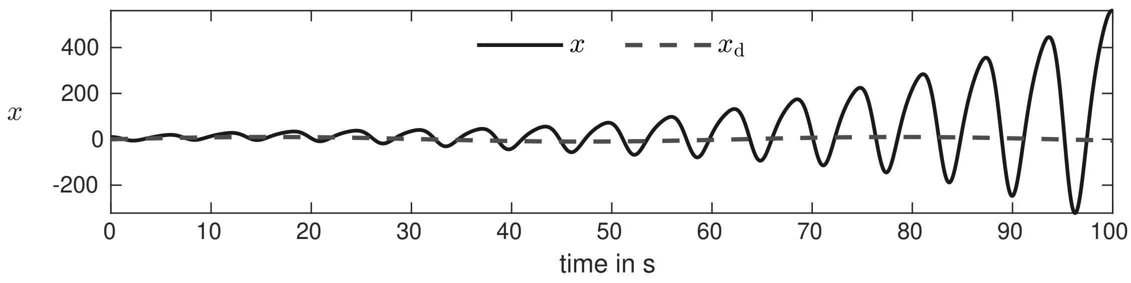


Figure 3.2: Simulation of a mass under standard implementation of impedance control with variable gains.

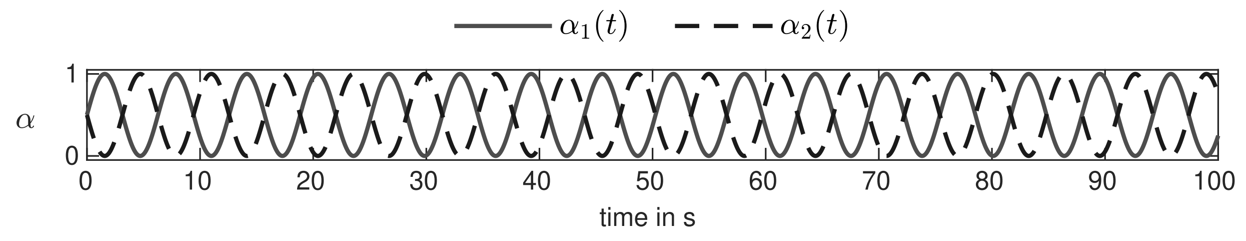
A simulation of the system with standard impedance control, *i. e.*, feedback of position and velocity errors with direct interpolation (3.6) of the gains, results in the trajectory shown in Fig. 3.2 for the first 100 s. The increasing magnitude of the oscillation results only from the hidden coupling induced by time-varying $K'(t)$, as the error system reduces to an exponentially stable LTI system for all constant $K'(t) = K' \in [-10, 10]$.

This problem can be circumvented by the architecture of Sec. 3.1.4; for simplicity, the one-degree-of-freedom setup is used for design while the state error is used during implementation. First, the desired gains are reformulated as a convex combination of the two matrices $\boldsymbol{D}_{\mathrm{K},1} \triangleq -\max\left[K_\mathrm{P}(t), K_\mathrm{D}(t)\right] = -\left[22, 1\right]$ and $\boldsymbol{D}_{\mathrm{K},2} \triangleq -\min\left[K_\mathrm{P}(t), K_\mathrm{D}(t)\right] = -\left[2, 1\right]$. The interpolation signal such that $-\left[K_\mathrm{P}(t), K_\mathrm{D}(t)\right] = \alpha_1 \boldsymbol{D}_{\mathrm{K},1} + \alpha_2 \boldsymbol{D}_{\mathrm{K},2}$ is then given by $\alpha_1 = \frac{1}{2}\left(\sin(t) + 1\right)$ and $\alpha_2 = 1 - \alpha_1$. Next, one needs to design the initial gain $\boldsymbol{D}_{\mathrm{K},0}$. It may seem natural to choose $\boldsymbol{D}_{\mathrm{K},i} \in \mathcal{K}$ or average $\bar{\boldsymbol{D}}_\mathrm{K} \triangleq \frac{\boldsymbol{D}_{\mathrm{K},1} + \boldsymbol{D}_{\mathrm{K},2}}{2} = -\left[12, 1\right]$. As by Cor. 3.1, we also calculate $\boldsymbol{D}_{\mathrm{K},0} = \boldsymbol{F}_\mathrm{N}$ by simply taking $\boldsymbol{C}_1 = \boldsymbol{I}$ and $\boldsymbol{D}_{12} = \boldsymbol{0}$, *i. e.*, the performance variable is $\boldsymbol{z} = \boldsymbol{x}$ and equally weighs position and velocity. This results in $\boldsymbol{D}_{\mathrm{K},0} \approx \left[-1.00, -4.58\right]$. The dynamic parameters Q_1 and Q_2 for controller recovery of $\boldsymbol{D}_{\mathrm{K},1}$ and $\boldsymbol{D}_{\mathrm{K},2}$ can now be calculated by (3.10). The resulting controllers are labeled A–D as summarized in Tab. 3.1. In order to realize the interpolated $Q(\boldsymbol{\alpha})$ to satisfy (3.8), both (3.11) and (3.12) with $\boldsymbol{P}_\mathrm{Q} = \boldsymbol{I}$ are considered, yielding $Q_\mathrm{LQN}(\boldsymbol{\alpha})$ of order 4 and $Q_\mathrm{LPV}(\boldsymbol{\alpha})$ of order 2, respectively.

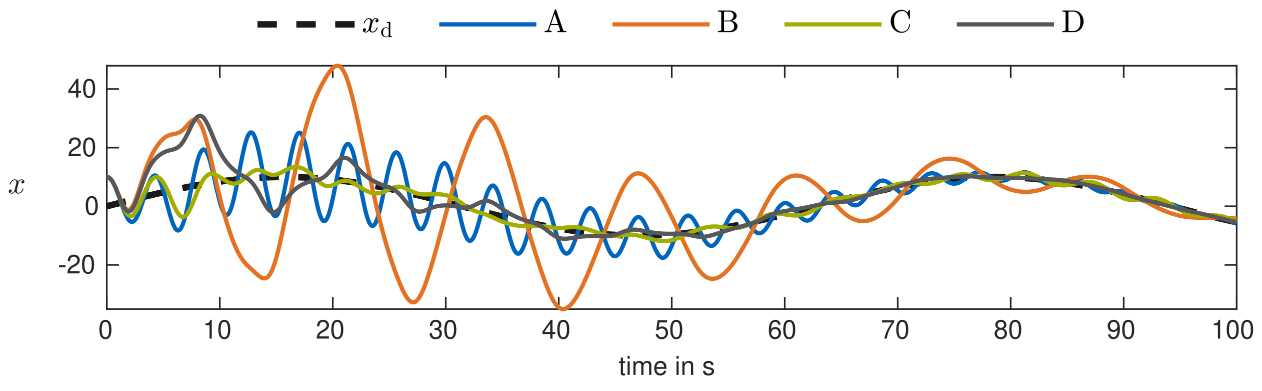
The simulation results are shown in Fig. 3.3. Controllers A–D achieve asymptotic stabilization of the error system, effectively avoiding the scheduling-induced instability of Fig. 3.2. Opposed to [54], the simple controller of this section does not result in high-frequency chattering of the control input $\boldsymbol{u}$. The analysis of [123] in turn only allows to check if an impedance

Table 3.1: Controller configurations used for the illustrative simulation study of a point mass under variable impedance control

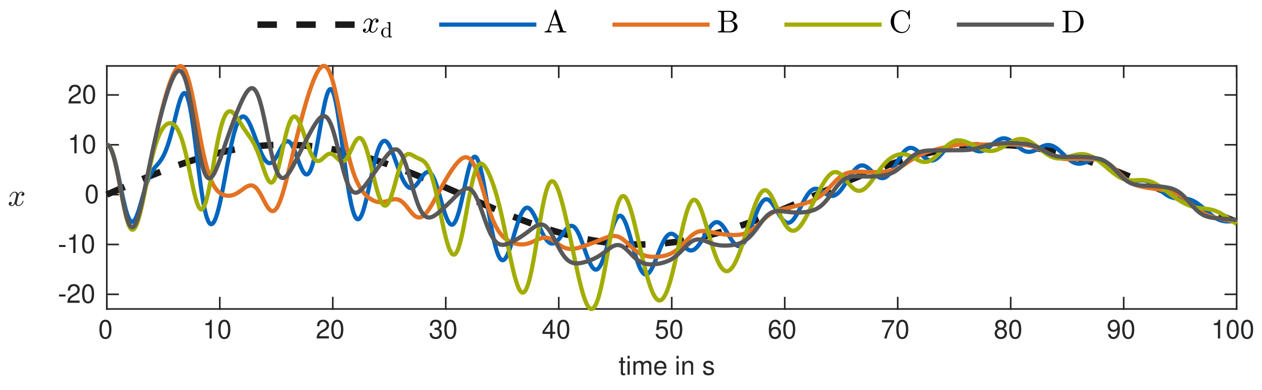
Label	$\boldsymbol{D}_{\mathrm{K},0}$	$\lVert Q_1 \rVert_\infty$	$\lVert Q_2 \rVert_\infty$	$\lVert S \rVert_\infty$ for M/M_nom		
				0.1	1	5
A	$\boldsymbol{D}_{\mathrm{K},1}$	0	894.7	1.203	0	1.808
B	$\boldsymbol{D}_{\mathrm{K},2}$	270.4	0	2.404	0	5.060
C	$\bar{\boldsymbol{D}}_\mathrm{K}$	68.16	226.6	1.347	0	2.270
D	$\boldsymbol{F}_\mathrm{N}$	317.0	18.97	0.399	0	1.267



(a) Interpolation signal



(b) Implementation with Q_{LQN} (3.11)



(c) Simulation with Q_{LPV} (3.12)

Figure 3.3: Simulation of the mass-spring damper system with varying impedance, implemented by the controller of Fig. 3.1 with single gain (3.13). Labels A–D refer to the settings according to Tab. 3.1.

profile is suitable for ad-hoc implementation by (3.6). Indeed, the stiffness profile (3.14) is disqualified in [123], whereas the parameterization approach reported here constitutes a straightforward synthesis method to render admissible behavior to the system, implicitly altering the transient behavior as required.

3.1.9 Additional Discussion and Implementation Aspects

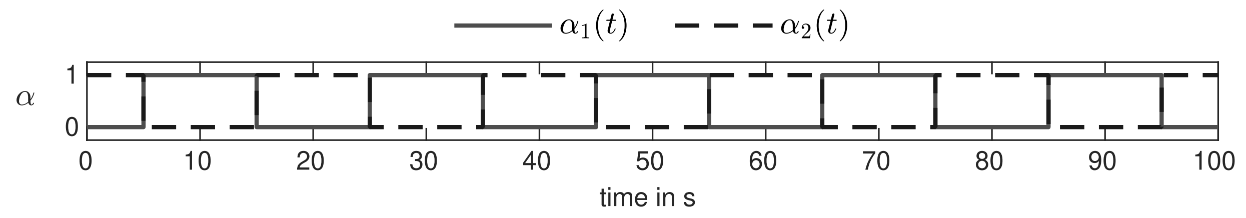
Recall that $K(\boldsymbol{\alpha})$ reduces to the corresponding static feedback for frozen $\boldsymbol{\alpha}$. Therefore, the effect of the stabilizing parameterization becomes discernible only when $\boldsymbol{\alpha}$ is varying. The central controller $\boldsymbol{D}_{\mathrm{K},0}$, the coprime factor stabilizing gain $\boldsymbol{F}$, and the realization of interpolated $Q(\boldsymbol{\alpha})$ all affect the transient behavior.

It depends on the particular application if the implementation by either Q_{LQN} or Q_{LPV} is preferable. In our simulation studies, we could not observe a general advantage of the higher-order Q_{LQN} over Q_{LPV} except for a more straightforward implementation. However, the state dimensionality of Q_{LQN} may become an issue particularly if $|\mathcal{K}|$ is large.

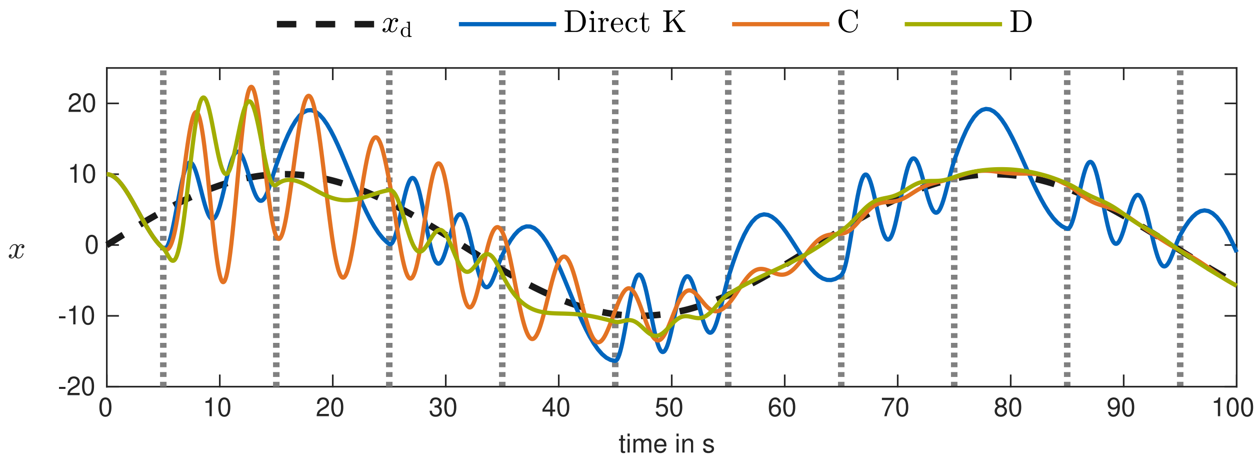
Due to the parameterization architecture, the difference of the feedback controllers w. r. t. the nominal loop is separated by (B.12) into the corresponding plug-in filters $\mathcal{Q}$. Therefore, one could aim to keep $\max_{Q_i \in \mathcal{Q}} \|Q_i\|_\infty$ low by choice of the parameters. With equal gains (3.13), the only decisive factor in (3.10) is $(\boldsymbol{D}_{\mathrm{K},i} - \boldsymbol{D}_{\mathrm{K},0})$. This point of view suggests to use $\boldsymbol{D}_{\mathrm{K},0} = \bar{\boldsymbol{D}}_{\mathrm{K}}$ (controller C), instead of $\boldsymbol{D}_{\mathrm{K},0} \in \mathcal{K}$ (controllers A and B).

Given the requirement of local controller recovery, in general no assertions can be made concerning robustness— the maximum coprime factor uncertainty margin of the central gain $\boldsymbol{F}_{\mathrm{N}}$ is lost once the local recovery filters Q_1 and Q_2 are employed, *cf.* Remark 3.2. Nonetheless, controller D constitutes a reasonable compromise of transient performance (see Figs. 3.3 and 3.4) and robustness: in order to ensure the effectiveness of the parameterization, the dual Youla operator S [231] characterizing the model mismatch should be stable and build a stable feedback loop with Q. As summarized in Tab. 3.1, if for example the mass M is varied, the setup with gain $\boldsymbol{F}_{\mathrm{N}}$ yields a lower $\mathcal{H}_\infty$ norm to the dual Youla parameter computed according to [160, Th. 3.1]. Guaranteed robust arbitrary interpolation *and* controller recovery, however, would require that the controllers individually ensure robust stability w. r. t. the dual Youla parameter. In this case, a robust arbitrary switching scheme comes in reach as recently shown in a SISO setting [14].

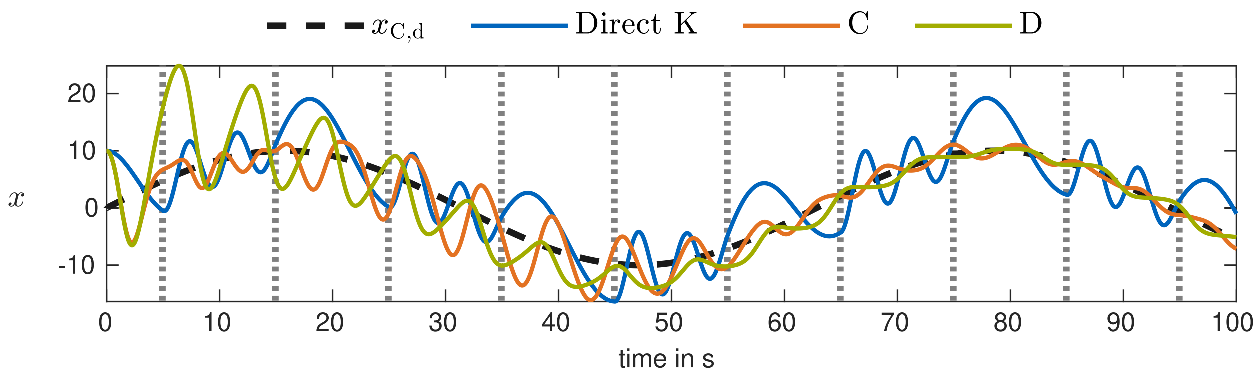
When switching controllers, peaks occur in the proposed scheme just as in the standard YK parameterization. Recall that controller D is based on the LQR criterion of Cor. 3.1, hence with an implicit penalty on the performance variables $\boldsymbol{z}$ respectively the control input $\boldsymbol{u}$. An example of good transient performance achievable with the scheme is shown in Fig. 3.4 for the mass-spring damper system. However, if the transient peaks are of primary concern and must be suppressed, requirement R3 of simple implementation should be lifted in favor of the full dynamic $\mathcal{H}_\infty$ interpolation scheme of [78]. In this case, notably more engineering effort is required to tune the weighting filters involved in the $\mathcal{H}_\infty$ design; moreover, the dynamic order of the resulting controller is considerably higher.



(a) Switching signal



(b) Implementation with Q_{LQN} (3.11)



(c) Simulation with Q_{LPV} (3.12)

Figure 3.4: Transient behavior under switching. The switching instants are marked by vertical dotted lines for enhanced visibility. Direct K refers to (3.6) and labels C–D to the controller architecture with $\boldsymbol{D}_{\mathsf{K},0}$ according to Tab. 3.1.

3.2 Direct Adaptive-Q Control for Switching Linear Systems

In the previous section, a parameterization was derived with a focus on simplicity and ease of implementation. In this parameterization, multiple controllers can be switched or interpolated in order to adapt to changing control objectives; however, the plant dynamics must not change substantially. In this section, we consider arbitrarily switching linear systems as a starting point and extend the adaptive-Q method to achieve fine-tuning of the switching controller.

3.2.1 Motivation

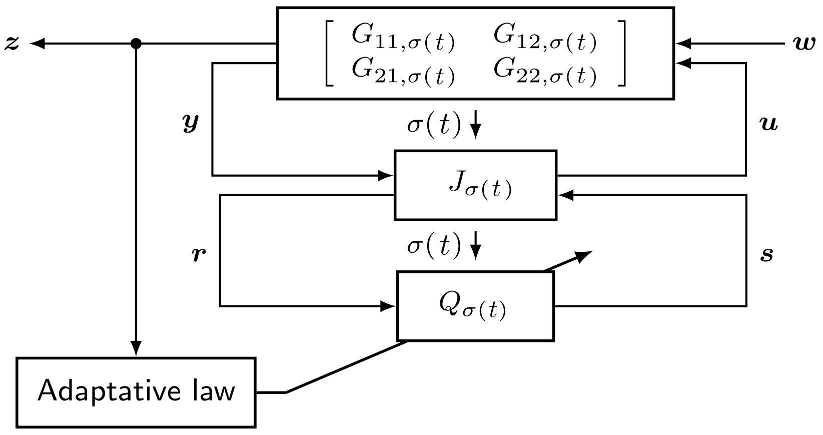
Linear switching systems constitute an important class of hybrid systems. On the one hand, the dynamics of many systems can be described or approximated by a hybrid system switching among a number of linear systems, see *e. g.* [137]. On the other hand, it is common to switch between several controllers to adapt to different operating conditions, as for example in [82]. Sufficient conditions for a linear plant to be stabilizable under arbitrary controller switching were given in [80]. The switching controller can be constructed using state reset maps, or in a suitable realization based on the theory of all stabilizing controllers [265, 125]. In [27] these results were generalized to the case when also the plant is switching. Under certain conditions, a switching, quadratically stabilizing observer-based controller can be designed by solving a set of linear matrix inequalities. Then, any set of desired controllers must be realized appropriately to ensure quadratic stability under arbitrary switching. Such a set of controllers is usually designed a priori for a range of operating conditions or family of disturbance signals. During operation, a suitable controller is then plugged into the loop.

In order to improve the performance of a switching controller, one approach is to calculate optimized reset maps to determine the initial state of the controller when switching [220, 79, 198]. These approaches tackle particularly the performance loss due to the transients induced by switching dynamic controllers. In this section, a complementary approach is developed, *i. e.*, an adaptation scheme to continuously adjust each controller while it is in operation. To this end, we employ the adaptive-Q control approach reviewed in Sec. 2.4.

The contribution of this section is to show that the use of an adaptive-Q control scheme is advantageous particularly in a switching context: this is due to the fact that both the adaptation scheme to enhance performance online, as well as the baseline design of a switching stabilizing compensator operate on the set of all stabilizing controllers. In other words, the switching stable controller realization is given in terms of an observer-based pre-compensator and an additional input injection by means of a stable filter; this filter in turn can be tuned online. The results on adaptive-Q control reported in the literature were so far mainly based on linear systems or time-varying linearizations of nonlinear systems along a particular trajectory [231].

3.2.2 Problem Statement

Throughout this section, we consider discrete-time systems only. We are now ready to state the problem setup, which is depicted in Fig. 3.5.



z
w
y
u
r
s
$\begin{bmatrix} G_{11,\sigma(t)} & G_{12,\sigma(t)} \\ G_{21,\sigma(t)} & G_{22,\sigma(t)} \end{bmatrix}$
$\sigma(t)$
$J_{\sigma(t)}$
$\sigma(t)$
$Q_{\sigma(t)}$
Adaptative law

Figure 3.5: The problem setup considered in this section: online adjustment of a switching compensator to enhance performance w. r. t. a cost index depending on $\boldsymbol{z}$, under the influence of a partially unknown external input $\boldsymbol{w}$.

Problem 3.2. Consider the discrete-time switching plant G_σ

$$\begin{bmatrix} \boldsymbol{x}(t+1) \\ \boldsymbol{z}(t) \\ \boldsymbol{y}(t) \end{bmatrix} = \left[\begin{array}{c|cc} \boldsymbol{A}_{\sigma(t)} & \boldsymbol{B}_{1,\sigma(t)} & \boldsymbol{B}_{2,\sigma(t)} \\ \hline \boldsymbol{C}_{1,\sigma(t)} & \boldsymbol{D}_{11,\sigma(t)} & \boldsymbol{D}_{12,\sigma(t)} \\ \boldsymbol{C}_{2,\sigma(t)} & \boldsymbol{D}_{21,\sigma(t)} & \boldsymbol{0} \end{array}\right] \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{w}(t) \\ \boldsymbol{u}(t) \end{bmatrix}, \tag{3.15}$$

where $\sigma(t) \in \mathcal{I}$ is arbitrarily switched. The state vector is $\boldsymbol{x} \in \mathbb{R}^n$ with initial state $\boldsymbol{x}_0$; $\boldsymbol{w} \in \mathbb{R}^{n_\mathrm{w}}$ and $\boldsymbol{u} \in \mathbb{R}^{n_\mathrm{u}}$ denote exogenous and control inputs, respectively; $\boldsymbol{z} \in \mathbb{R}^{n_\mathrm{z}}$ are the outputs subject to performance specifications and $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{y}}$ are the outputs available to the controller. Let Assumption 2.1 hold for the G_{22} channel and switching is according to Assumption 2.2. Design an adaptive switching compensator $\{\mathcal{J}, \mathcal{Q}\}$ with adjustable parameters $\Xi(t)$ to reduce the cost index

$$C_\mathrm{z}(\Xi) = \frac{1}{t_\mathrm{N}} \sum_{k=0}^{t_\mathrm{N}} \boldsymbol{z}(t)^\top \boldsymbol{R}_\mathrm{z} \boldsymbol{z}(t) \tag{3.16}$$

during operation, while assuring quadratic stability under arbitrary switching for partially unknown-but-bounded $\boldsymbol{w} \in \ell_\infty$ and a weighting matrix $\boldsymbol{R}_\mathrm{z} = \boldsymbol{R}_\mathrm{z}^\top \succeq 0$. ◇

3.2.3 Design Methodology

The result shown in this section is a procedure to solve Problem 3.2.

Procedure 3.1 *(Adaptive-Q for switching linear systems).*

1. Check that Assumption 2.1 is fulfilled, *i. e.*, the switching plant defined by the matrices $(\boldsymbol{A}_i, \boldsymbol{B}_{2,i}, \boldsymbol{C}_{2,i})$ is switching quadratically stabilizable. This is the case if (2.24) is solvable.

2. Calculate the feedback gain and state estimation matrices according to (2.27) and realize the pre-compensator $J_{\sigma(t)}$ as in (2.26). This ensures switching quadratic stability of G_{22}

according to Theorem 2.9 and consequently bounded-input bounded-output stability of the setup in Fig. 3.5 for $\mathcal{Q} = \emptyset$.

3. Add the input injection $\boldsymbol{s}$ according to (2.28) using a set $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_{N_{\text{sys}}}\}$, where each $Q_i \in \mathcal{RH}_\infty$ is realized as

$$Q_i \triangleq \left[\begin{array}{c|c} \boldsymbol{A}_{\text{Q},i} & \boldsymbol{B}_{\text{Q},i} \\ \hline \boldsymbol{C}_{\text{Q},i} & \boldsymbol{0} \end{array}\right], \tag{3.17}$$

$\boldsymbol{A}_{\text{Q},i} \in \mathbb{R}^{n_\text{q} \times n_\text{q}}$, $\boldsymbol{B}_{\text{Q},i} \in \mathbb{R}^{n_\text{q} \times n_\text{y}}$, $\boldsymbol{C}_{\text{Q},i} \in \mathbb{R}^{n_\text{u} \times n_\text{q}}$, such that

$$\boldsymbol{A}_{\text{Q},i}^\top \boldsymbol{P}_\text{Q} \boldsymbol{A}_{\text{Q},i} - \boldsymbol{P}_\text{Q} \prec 0, \quad \forall i \in \mathcal{I} \tag{3.18}$$

is solvable for a positive definite symmetric matrix $\boldsymbol{P}_\text{Q} \in \mathbb{R}^{n_\text{q} \times n_\text{q}}$. Thus, the set $\mathcal{Q}$ of additional filters shares a CQLF function, preserving switching stability of the closed loop.

4. During operation, interpret $\boldsymbol{C}_{\text{Q},i}$ as adjustable matrix parameter $\Xi_i(t)$, $\Xi_i(0) = \boldsymbol{C}_{\text{Q},i}$ for all $i \in \mathcal{I}$. Apply a least-squares update rule in order to adapt the parameter matrices $\Xi_i(t)$ such that the overall cost index (3.16) is being minimized.
One particular choice is given by a stochastic gradient descent such that the variables of the currently active Q-parameter are updated, *i. e.*

$$\forall i = 1, \dots, N_\text{sys} : \quad \forall m = 1, \dots n_\text{u}, \quad \forall n = 1, \dots, n_\text{q} :$$

$$\widehat{\Xi}_{i,mn}(t+1) = \Xi_{i,mn}(t) - \vartheta_i \mu_i \left.\frac{\partial \boldsymbol{z}(t)^\top}{\partial\, \Xi_{i,mn}}\right|_{\Xi_i(t)} \boldsymbol{R}_\text{z} \boldsymbol{z}(t), \tag{3.19}$$

$$\text{where } \vartheta_i = \begin{cases} 1 \text{ if } i = \sigma(t), \\ 0 \text{ otherwise.} \end{cases} \tag{3.20}$$

In order to guarantee that $\Xi_i(t)$ stays bounded for all t, project the parameters back on a ball $\mathcal{B}(0, \xi)$ after the update,

$$\Xi_i(t+1) = \begin{cases} \widehat{\Xi}_i(t+1) \text{ if } \|\widehat{\Xi}_i(t+1)\| < \xi, \\ \widehat{\Xi}_i(t+1) \dfrac{\xi}{\|\widehat{\Xi}_i(t+1)\|} \text{ otherwise.} \end{cases}$$

In the following, the factors ϑ_i and μ_i are called *submodel adaptation factors* and *submodel update rates*, respectively. The stability of the resulting closed loop is analyzed as follows.

Theorem 3.2 (Stability of switching adaptive-Q control). Let a switching quadratically stabilizing compensator be given as the interconnection of a baseline controller $\mathcal{J}$ according to (2.26)-(2.27), and a switching quadratically stable $\mathcal{Q}$ according to (3.17)-(3.18), as depicted in Fig. 3.5. As long as $\Xi_i(t)$ is bounded $\forall i \in \mathcal{I}, \forall t$, quadratic stability is guaranteed despite parameter adjustment using a switching adaptive-Q law.

Proof: The proof is provided in Appendix B.5. ■

Theorem 3.2 implies that adjusting the output equations of the Q-parameters does not compromise stability when there is a mechanism in the adaptive law to ensure that $\Xi_i(t)$ remains finite. This offers an additional degree of freedom usable in switching control: just as in the linear adaptive-Q case [231], for each mode i suitable zeros can be placed in the closed-loop transfer function $T_{\boldsymbol{w}\to\boldsymbol{z}}$ by adaptation of Ξ_i.

It remains to give an expression for the gradient vector in (3.19). It can be calculated analytically for the nominal model, but depends on the signal interconnection used in the generalized plant (3.15). When employing the standard adaptive-Q scheme reviewed in Sec. 2.4, we have $\boldsymbol{y} = \boldsymbol{y}_\mathrm{P} - \boldsymbol{y}_\mathrm{ref}$, $\boldsymbol{w}^\top = \left[\boldsymbol{d}^\top, \boldsymbol{y}_\mathrm{ref}^\top\right]$ and $\boldsymbol{z}^\top = \left[(\boldsymbol{y}_\mathrm{P} - \boldsymbol{y}_\mathrm{ref})^\top, \boldsymbol{u}^\top\right]$, where $\boldsymbol{y}_\mathrm{P}$ is the actual plant output. Analogously as shown in [231, Chapter 6.3] for a single plant without switching, the dynamics of the ith gradient vector $\gamma_{i,mn}^\top(t) \triangleq \left.\frac{\partial \boldsymbol{z}(t)^\top}{\partial\,\Xi_{i,mn}}\right|_{\Xi_i(t)} \in \mathbb{R}^{n_\mathrm{y}+n_\mathrm{u}}$ can then be found from (B.13) as

$$\left[\begin{array}{c} \Gamma_{i,mn}(t+1) \\ \hline \gamma_{i,mn}(t) \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{A}_i + \boldsymbol{B}_{2,i}\boldsymbol{F}_i & \boldsymbol{B}_{2,i}\mathbb{1}_{mn} \\ \hline \boldsymbol{C}_{2,i} & \boldsymbol{0} \\ \boldsymbol{F}_i & \mathbb{1}_{mn} \end{array}\right] \left[\begin{array}{c} \Gamma_{i,mn}(t) \\ \hline \boldsymbol{x}_\mathrm{Q}(t) \end{array}\right] \tag{3.21}$$

with

$$\Gamma_{i,mn}(0) = 0. \tag{3.22}$$

Note that due to the affine nature of the closed-loop dynamics w.r.t. to the functions Q_i, the transient behavior of each gradient vector γ_i is independent of $\Xi_i(t)$.

Convergence. Next we show that the cost index (3.16) is improved by the adaptation. For brevity, we only summarize the result for linear systems and the specific adaptive-Q algorithm used. For a complete analysis based on averaging theory and a proof of convergence in the linear case, the reader is referred to [231, Chapter 6.4] and the remarks therein.

Proposition 3.1 (Parameter Convergence). Consider an interval $t_1 \leq t \leq t_2$ such that $\sigma(t) = i$. Assume the disturbance signal $\boldsymbol{w}(t)$ is bounded and such that there exists a unique $\Xi_i^\star$ fulfilling $C_\mathrm{z}(\Xi_i^\star) \leq C_\mathrm{z}(\Xi_i)$ for all Ξ_i. Then there exists a submodel update rate $\mu_i^\star > 0$, such that for all $\mu_i \in (0, \mu_i^\star)$ the system state is bounded and near optimal performance

$$\limsup_{t,t_2\to\infty} \|\Xi_i(t) - \Xi_i^\star\| \leq c_i,$$

is achieved exponentially, where c_i is a constant depending on μ_i.

Proof: With (3.20), adaptation is only applied to the active mode i. Thus, it suffices to consider segments without switching and the proposition is a reformulation of the analysis given in [231, Theorem 4.3], where also a detailed expression for c_i can be found. ■

Therefore, near optimal parameters $\Xi_i \to \Xi_i^\star$ are recovered after each switching instant if switching occurs sufficiently slow, *i. e.*, the convergence of the parameters Ξ_i in mode i is not disrupted by the next switch. The dynamics (3.21),(3.22) then approximate a steepest gradient for the currently active submodel and can be used to implement (3.19). The convergence rate depends on the submodel update rate μ_i being upper bounded by $\mu_i^\star$. It is consequently required that the adaptation is slow compared to the dynamics of the control loop in each mode i.

Discussion. The limitations of the proposed switching adaptive-Q algorithm are twofold. Firstly, quadratic stability is conservative as there are stable systems which are not quadratically stable and also systems stabilizable by switching which are not quadratically stabilizable [137]. Second, as in the linear case [231], direct adaptive-Q control may destabilize the loop if plant-model mismatch needs to be accounted for.

In the switching case, the achievable performance enhancement is more sensitive w. r. t. the submodel update rates μ_i than in the linear adaptive-Q algorithm. While a higher update rate $\mu_i < \mu_i^\star$ generally leads to faster convergence, the adaptive enhancement can lead to a worsening of the transients occurring due to switching. One simple way to mitigate this problem is to reset the gradient state $\mathbf{\Gamma}$ in (3.21) to zero after a mode switch at time t_s, where $\sigma(t_\mathrm{s}-1) \neq \sigma(t_\mathrm{s})$, and to disable the adaptation for a short time until the transients begin to decay. For this purpose, a suitable modification of the submodel adaptation factor (3.20) is

$$\vartheta_i = \begin{cases} 1 \text{ if } i = \sigma(t) \quad \wedge \quad t \geq t_\mathrm{s} + t_\mathrm{th}, \\ 0 \text{ otherwise}, \end{cases} \tag{3.23}$$

such that $t_\mathrm{th} \geq 1$ is a threshold design parameter to disable the adaptation for t_th steps after a mode switch.

Remark 3.3. In this section, the use of an adaptive-Q algorithm is investigated for switching linear systems. Naturally, the scheme can also be employed when a multi-controller is used for a single plant. In this case, $\mathcal{J}$ has only one element J, and the different controllers are obtained in an appropriate realization by switching in $\mathcal{Q}$, as derived in [80]. A main reason for using a multi-controller is the possibility to design several controllers for different operating conditions a priori (offline), and switch between them (online) in order to obtain a better overall performance [80, 137, 82, 220]. In this view, the adaptive-Q methodology allows to fine-tune such a controller online in order to enhance the performance further. $\triangleleft$

3.2.4 Numerical Example

We illustrate the technique described in the previous section with an academic example and highlight the achievable performance improvement.

Consider the following switching plant with one unstable and one stable mode, using a sampling time of $T_\mathrm{s} = 0.1\,\mathrm{s}$:

$$\widehat{G}_1(z) = \frac{-0.3z+1}{(z-1.25)(z+0.7)(z-0.15)}, \quad \widehat{G}_2(z) = \frac{0.4}{(z-0.5)^2(z+0.85)}. \tag{3.24}$$

The input is disturbed by an unknown additional signal d which cannot be measured. The characteristics of the reference signal y_ref is also unknown in advance. Furthermore, the output signal $y_\mathrm{P} \in \mathbb{R}$ of the actual switching plant $\widehat{G}_{\sigma(t)}$ is measured under additional white measurement noise y_noise, *i. e.*, the measured output y is the control error $y = y_\mathrm{P} - y_\mathrm{ref} + y_\mathrm{noise}$.

Let the exogenous signals of the generalized plant G be defined as

$$\boldsymbol{w} = \begin{bmatrix} d \\ y_{\text{ref}} \end{bmatrix}, \quad \boldsymbol{z} = \begin{bmatrix} y_{\text{P}} - y_{\text{ref}} + y_{\text{noise}} \\ u \end{bmatrix}. \tag{3.25}$$

With (3.25), the switching adaptive-Q design (3.17)–(3.23) can be used. We proceed according to procedure 3.1:

1. For a minimal realization of the augmented plant, the LMIs (2.24) can be solved using, *e. g.*, [141, 67]. We obtain the matrices $\boldsymbol{P}_{\text{F}} = \text{diag}(1.743, 5.089, 2.583)$ and

$$\boldsymbol{P}_{\text{L}} = \begin{bmatrix} 31.156 & -0.123 & -51.899 \\ -0.123 & 11.222 & -10.312 \\ -51.899 & -10.312 & 108.792 \end{bmatrix}.$$

2. Solving (2.27) yields the following feedback and estimation gains assuring switching quadratic stability:

$$\boldsymbol{F}_1 = \begin{bmatrix} -0.3500 \\ -0.3962 \\ 0.1312 \end{bmatrix}^\top, \quad \boldsymbol{F}_2 = \begin{bmatrix} -0.1500 \\ -0.6000 \\ 0.4250 \end{bmatrix}^\top, \quad \boldsymbol{L}_1 = \begin{bmatrix} -1.9411 \\ -1.9754 \\ -0.5320 \end{bmatrix}, \quad \boldsymbol{L}_2 = \begin{bmatrix} -0.5566 \\ -2.1645 \\ -0.6439 \end{bmatrix}.$$

3. Next, a set $\mathcal{Q} = \{Q_1, Q_2\}$ corresponding to a set $\mathcal{K}$ of desired controllers could be designed as switching stable realizations according to [27, Procedure 3.1]. In this example however, we simply use

$$Q_1 = \left[\begin{array}{ccc|c} 0.8 & 0 & 0 & 1 \\ 0 & 0.4 & 0 & 1 \\ 0 & 0 & 0.05 & 1 \\ \hline 0.5 & 0.2 & 0.75 & 0 \end{array}\right], \quad Q_2 = \left[\begin{array}{ccc|c} 0.02 & 0 & 0 & 1 \\ 0 & -0.4 & 0 & 1 \\ 0 & 0 & 0.4 & 1 \\ \hline -2 & 2 & 3 & 0 \end{array}\right], \tag{3.26}$$

chosen such that the eigenvalues correspond roughly to the absolute values of the observer dynamics $(\boldsymbol{A}_1 + \boldsymbol{L}_1\boldsymbol{C}_1)$ and $(\boldsymbol{A}_2 + \boldsymbol{L}_2\boldsymbol{C}_2)$. Moreover, Q_1 and Q_2 share a common quadratic Lyapunov function characterized by

$$\boldsymbol{P}_{\text{Q}} = \begin{bmatrix} 2.7778 & 0 & 0 \\ 0 & 1.1905 & 0 \\ 0 & 0 & 1.0989 \end{bmatrix}.$$

4. The initial value of the adjustable parameters are

$$\begin{aligned} \Xi_1(0) &= \boldsymbol{C}_{\text{Q},1} = [0.5 \quad 0.2 \quad 0.75], \\ \Xi_2(0) &= \boldsymbol{C}_{\text{Q},2} = [-2 \quad 2 \quad 3]. \end{aligned}$$

The plant is subject to frequent switching, noisy measurement and a severe input disturbance, as depicted in Fig. 3.6. We used $\boldsymbol{R}_{\text{z}} = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}$ to calculate the cost indices (3.16) in all simulations presented in this section. The results of the simulation study are shown in Fig. 3.6 – 3.9 and are summarized in Tab. 3.2.

For a fair assessment of the performance enhancement, we begin with a set of constant, well-tuned Q-parameters. Using the controllers given by (3.26) reduces the cost already considerably, compared to the stabilizing baseline controller (2.26)-(2.27) (*cf.* Tab. 3.2). The performance of this controller is shown by the orange line in Fig. 3.7. It is evident that the controller yields a good tracking behavior as long as the disturbances are not severe. However, the additive input disturbance from 50 s on corrupts the performance particularly when the first, *i. e.* unstable, mode is active. On the other hand, the blue line in Fig. 3.7 shows the performance of the adaptive scheme which quickly compensates for the varying disturbance and recovers good tracking behavior in both modes. We used a switching adaptive-Q controller with state reset in (3.21) after each switch characterized by the parameters

$$\mu_1 = 2 \cdot 10^{-5},\ \mu_2 = 0.01,\ \xi = 10,\ k_{\text{th}} = 30. \tag{3.27}$$

The corresponding evolution of the parameters Ξ_i used in the output equation of the Q-filters are shown in Fig. 3.8.

As pointed out in Sec. 3.2.3, the adaptation may have a deteriorating effect on the control performance in face of the switching transients. Indeed, setting $t_{\text{th}} = 0$ leads to a higher overall cost index in our example, as shown in Fig. 3.9. Stability is not compromised by this effect but the benefits of the adaptation in the continuous phases may be outweighed by the loss due to the reinforcement of the transients after switching. This effect can be effectively countered by the modification (3.23), *i. e.*, for $t_{\text{th}} = 30$ we could use high submodel update rates μ_1 and μ_2 to achieve fast convergence to the reference signal despite switching and disturbances.

3.3 Conclusion

In the review of the stabilizing controllers parameterization in Chapter 2, it was shown that the parameterization approach to assure stability when switching or interpolating controllers has already received some attention theoretically. However, there is yet a lack of dissemination to a wider range of application domains and we have therefore formulated requirements from a practitioner's point of view in Sec. 3.1. Consequently, a relatively simple architecture was proposed tailored on the state feedback case. The resulting parameterization is simple to implement and features a transparent interpretation. Moreover, the novel scheme is using the terminal connections of the controller, which makes it suitable for implementation on actual robotic systems. Its utility was demonstrated by the variable impedance control problem.

Table 3.2: Simulation results of adaptive-Q approaches for switching control

Algorithm	**Final Cost** (3.16)
Quadratically stabilizing controller $\mathcal{J}$ ($Q_1 = Q_2 = 0$)	362
Control with constant Q_1, Q_2 from (3.26)	9.51
Switching adaptive-Q according to (3.27), $t_{\text{th}} = 0$	4.56
Switching adaptive-Q according to (3.27), $t_{\text{th}} = 30$	3.63

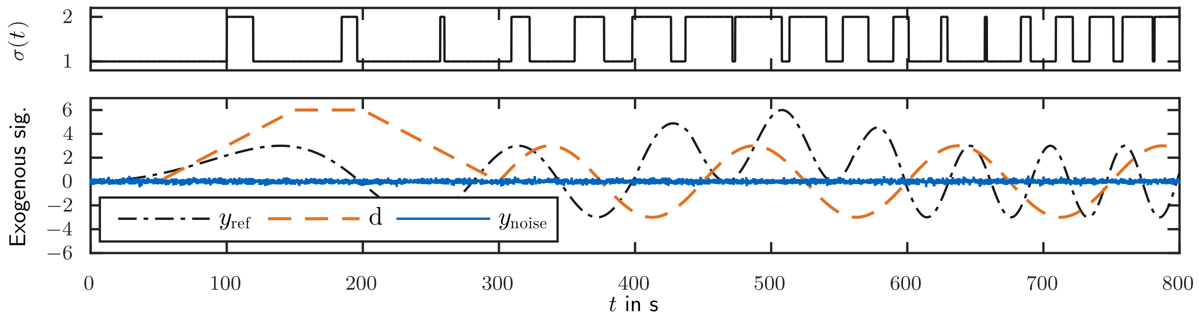


Figure 3.6: The switching signal σ, the reference signal y_{ref}, the input disturbance d and the additional noise y_{noise} at the output of the plant used during the simulation.

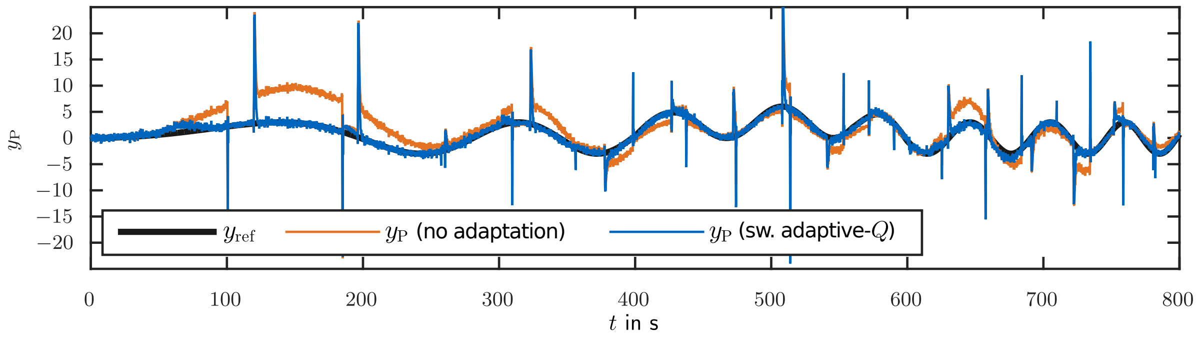


Figure 3.7: Output tracking behavior of the closed-loop system for a frequency-swept reference signal y_{ref}. The orange line shows the achievable performance using the Q-parameters from (3.26). The blue line indicates the plant output under switching adaptive-Q control with parameters according to (3.27) and with $t_{\mathrm{th}} = 30$ in (3.23).

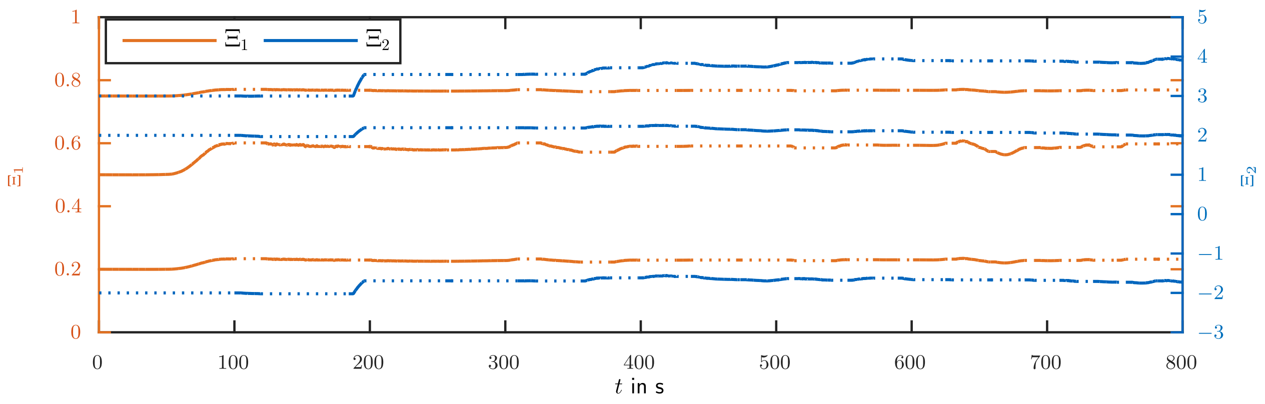


Figure 3.8: Time evolution of the parameters Ξ_i of the two modes of the switching control system. The solid/dotted lines indicate which subsystem is active/inactive.

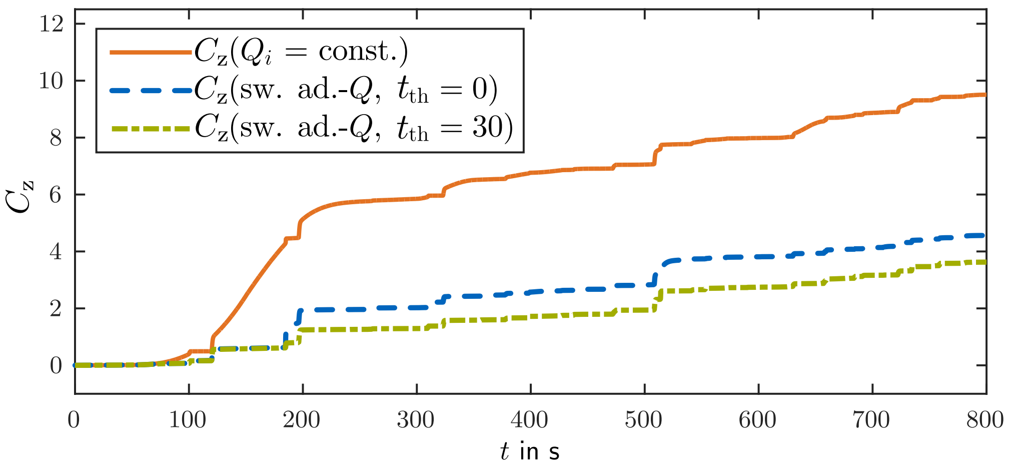


Figure 3.9: Time evolution of the cost index C_z for different simulation runs.

Finally, guidelines were provided for the selection of a more advanced scheme where necessary. A topic of future research is the online adaptation of the plug-in systems in the spirit of adaptive-Q control. Luo *et al.* recently propose in [142] an integrated process monitoring and control technique, allowing for online optimization by an adaptive-Q scheme in a Youla-like parameterization that is based on the terminal connections of PID controllers. Therefore, the method of [142] is applicable to industrial control systems where the pre-designed controllers cannot be modified. Thus, adding online adaptation capabilities to the state feedback based parameterization proposed in this chapter could lead to a number of interesting industrial applications including robot manipulator control.

In Sec. 3.2, the classic adaptive-Q methodology is extended to a class of hybrid systems operating under uncertain or unknown external disturbances. While the classic adaptive-Q method was mainly limited to the augmentation of robustly controlled linear plants, this section shows how it can be employed in controllers for switching plants: remarkably, performance enhancement of existing controllers realized according to [80, 27] can be achieved by online adjustment of the Q-parameters that are already in place in order to build a specific switching quadratically stabilizing controller. More sophisticated update laws could be used instead of the gradient descent algorithm (3.19), see for instance [75]. One limitation of the method is the occurrence of the switching transients, although their effect can be reduced using (3.23). Opposed to the nonadaptive case, where reset maps [220, 79, 198] can be used to optimally counteract the effect of the transients, it is not straightforward to extend that approach to the switching adaptive-Q scheme as investigated in [145]. This is because the optimal reset maps are calculated offline and rely on models of the closed loop. The parameters of the filters, however, are adapted online and these issues remain to be solved in future work.

4

A Parameterization of Robustly Stabilizing Controllers for Robot Manipulators

In the previous chapter, the parameterization of stabilizing controllers was investigated in the context of adaptive control methods based on local modes, assuming that an accurate model of the controlled process is available. In this chapter, we will focus on the parameterization in the context of robotics. Uncertainty in the plant description, arising inherently from approximate inverse dynamics (AID) control approaches, will be considered using the dual parameterization of plants stabilized by a given controller. Our goal is to shape a set of controllers designed for robust control of robot manipulators, such that numerous advanced design methods are applicable for enhancement of the manipulator controller in the robust stability framework of the double-Youla parameterization.

First, a generalized plant description of the rigid-body robot manipulator under AID control is derived. An important aspect is that the sources of uncertainty are kept separated throughout. This is in contrast to the most well-known robust robot control approaches, which lump all neglected dynamics in a single coupled and nonlinear term. Next, the loop uncertainty occurring within the interplay of neglected manipulator dynamics and the gains of the outer-loop controller is quantified. This problem is approached by means of a dual Youla parameterization that has not yet been exploited in this context of robot manipulator control. The main result of the chapter is the combined primary and dual Youla parameterization for robust control of rigid-body manipulators, *i. e.*, a double-Youla parameterization for manipulator control. The result applies to a wide range of approximate inverse dynamics configurations, a two-degree-of-freedom (feedforward/feedback) controller design, and can be used in conjunction with numerous methods for the design of the Q-parameter. The derivation contains several novel theoretical results beyond the scope of robotics. In particular, an explicit state-space realization of the dual Youla parameter is derived for a non-dynamic uncertainty operator, given an arbitrary linear stabilizing controller and a strictly proper plant.

In order to make the method accessible to practitioners, the very common case is considered of applying a PD state feedback as baseline outer-loop controller. Thereby, the framework specializes to a double-Youla parameterization based on a static central controller. The uncertainty measurement of the dual parameterization is discussed by study of a standard planar elbow manipulator tracking control example. The entire rigorous robust stability framework proposed in this chapter is finally illustrated with a six DoF manipulator in a challenging varying payload situation.

This chapter was published mostly in [60] and the remainder is structured as follows. We first introduce the motivation in Sec. 4.1 before we review the most relevant robust AID control methods in more detail in Sec. 4.2. The precise problem settings tackled in this chapter are introduced in Sec. 4.3. We then reformulate the uncertainty bounds into a generalized plant description in Sec. 4.4, before deriving in Sec. 4.5 a dual Youla measure of closed-loop uncertainty in the robot manipulator control loop. This leads to the main result given in Sec. 4.6, a general double-Youla parameterization for robot manipulators under approximate inverse dynamics control; the specialization using a static nominal controller is then reported in Sec. 4.7. The design steps are summarized in Sec. 4.8. In Sec. 4.9, the results are exemplified and thoroughly discussed by means of a planar elbow manipulator and a PUMA P560 robot model with six degrees of freedom (DoF). After a discussion of current limitations of the proposed method in Sec. 4.10, the chapter concludes with Sec. 4.11. The mathematical proofs and some technical details are provided in Appendix B.

4.1 Introduction

4.1.1 Motivation

Operating conditions of robot manipulators change over time, impairing the control performance— for example, by wear, varying load, *etc.* In order to adapt the controller to such situations, a variety of data-driven methods has been proposed in the literature, including adaptive and learning control [215, 210, 219, 132]. In practice, however, very simple feedback controllers and model-based architectures are prevalent.

On the one hand, if the complete dynamic model is available, feedback linearization can be applied. On the other hand, simple, completely model free controllers are ubiquitous, most prominently the popular proportional-derivative (PD) controller. Both may be interpreted as two extremes on the spectrum of approximate inverse dynamics (AID) controllers: ideal linearization is theoretically achieved if the model is perfect; if, on the contrary there simply is no dynamical model in the controller, only the PD action of the outer loop remains. In this chapter, we consider the following question: given a robot manipulator under AID control, *how* and *how much* can the outer-loop controller be modified during operation without compromising stability? Answering this question ultimately allows to use for online control performance enhancement methods that are otherwise hard to employ in a strict robust stability framework, *e. g.*, black-box optimization or reinforcement learning.

It is intuitively clear that the worse the controller is informed about the dynamic characteristics of the controlled robot, the harder it will be to provide stability guarantees when the controller is modified during operation. This situation is depicted in Fig. 4.1 and motivates this chapter: we provide a method to *quantify* this trade-off. We then characterize a *set* of robustly stabilizing controllers $\mathcal{K}_\mathrm{R}$, *i. e.*, all controllers contained in $\mathcal{K}_\mathrm{R}$ stabilize the uncertain nonlinear inner loop resulting from a particular AID situation. Interestingly, not only the model accuracy but also the nominal outer-loop controller K influence the amount of apparent uncertainty in the loop. Only comparatively recently, Bascetta and Rocco [13]

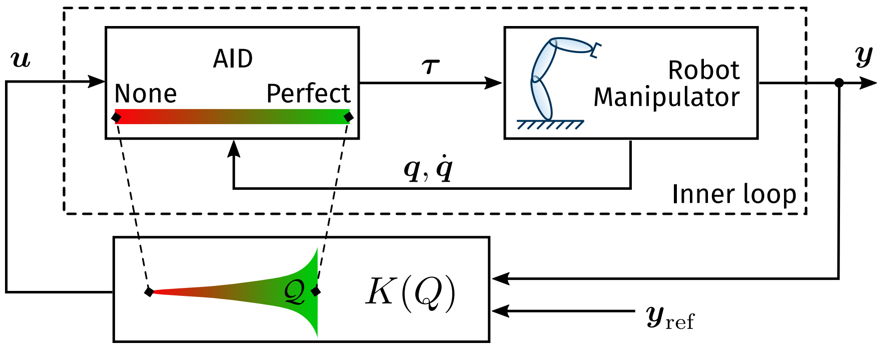


Figure 4.1: The goal of this chapter is to parameterize a set of robustly stabilizing feedback controllers for rigid-body robot manipulators. To this end, a general AID controller is considered in the inner loop that may range from a perfect inverse dynamics model to none at all. An outer-loop feedback controller K be given that stabilizes the loop but may not yet yield satisfactory performance. Therefore, in general, one can augment K by some stable parameter $Q \in \mathcal{Q} \subseteq \mathcal{RH}_\infty$. However, if the dynamic model of in the inner AID loop is imperfect, only a subset $\mathcal{Q} \subset \mathcal{RH}_\infty$ is admissible to preserve the stability of the overall loop. In this chapter, the set $\mathcal{Q}$ and the controller parameterization $K(Q)$ are characterized such that various advanced design methods can be used for robust controller performance enhancement.

reformulate the robust control of rigid manipulators to account for this issue, after it had not been considered by the most common robust robot control methods (see [215, 210, 219, 132]). The new framework proposed in this chapter provides a unified perspective and constitutes a tool to explore freedom in controller (re-)tuning, given a simplified model of the robot manipulator and a nominal outer-loop controller K.

Central to our method is a Q-parameterization of stabilizing controllers and the dual S-parameterization of plants stabilized by a controller as reviewed in Sec. 2.1 and in Sec. 2.2.1. The primary parameterization was already applied to the robust linear design of robot controllers around 30 years ago [218], known in robotics as the stable factorization approach [218, 246]. Our interest in extending this approach is due to the beneficial stability and robustness properties obtained by a Q-parameterization: such a parameterization is very helpful to leverage machine learning also in a feedback configuration, that is, in closed loop and on hardware. Inherently, the question is raised of how much feedback controller modification is admissible, given the effect of neglected uncertainties in the closed-loop system. The dual Youla parameter S is needed in order to answer this question and there is yet a gap in the literature how to utilize the dual parameterization [160] for the control of robot manipulators.

4.1.2 Control of Rigid-Body Robot Manipulators

Manipulator Dynamics and Tracking control. We consider a rigid-body robot manipulator with n links, described by the standard Euler-Lagrange model [219, 210, 215]

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{dist}}, \tag{4.1}$$

$$\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{f}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}), \tag{4.2}$$

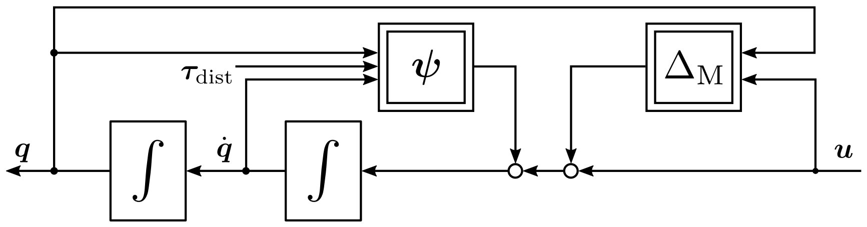


Figure 4.2: Perturbed double integrator structure resulting from approximate inverse dynamics control

where $\boldsymbol{q} \in \mathbb{R}^n$ is the vector of generalized coordinates (representing joint positions), $\boldsymbol{\tau} \in \mathbb{R}^n$ is the input vector of generalized force (torque), $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$, $\boldsymbol{M}(\boldsymbol{q}) \succ 0$ is the inertia matrix, and $\boldsymbol{n} \in \mathbb{R}^n$ is a vector that summarizes the vector of Coriolis and centrifugal terms $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathbb{R}^n$, the friction terms $\boldsymbol{f}(\dot{\boldsymbol{q}}) \in \mathbb{R}^n$, and the gravitational terms $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$. It is assumed that the input disturbance $\boldsymbol{\tau}_{\text{dist}}$ is a Lebesgue measurable function. The state of the system is denoted $\boldsymbol{x} = \text{col}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. Given a desired path in joint space $\boldsymbol{q}_{\text{d}} \in \mathcal{C}^2$, $\boldsymbol{q}_{\text{d}}, \dot{\boldsymbol{q}}_{\text{d}}, \ddot{\boldsymbol{q}}_{\text{d}} \in \mathcal{L}_\infty(\mathbb{R}_+, \mathbb{R}^n)$, the tracking error is formed as $\boldsymbol{e} = \text{col}(\boldsymbol{e}_1, \boldsymbol{e}_2) = \text{col}(\boldsymbol{q}_{\text{d}} - \boldsymbol{q}, \dot{\boldsymbol{q}}_{\text{d}} - \dot{\boldsymbol{q}})$. The measurement available for feedback is assumed as $\hat{\boldsymbol{q}} = \boldsymbol{q} + \boldsymbol{w}_1$, $\hat{\dot{\boldsymbol{q}}} = \dot{\boldsymbol{q}} + \boldsymbol{w}_2$, where $\boldsymbol{w}$ represents measurement uncertainty. The goal of tracking control is to find a controller generating $\boldsymbol{\tau}$ such that $\boldsymbol{e}$ vanishes.

Approximate Inverse Dynamics Control. The parameters of (4.1) are in practice not known exactly. The *approximate* or *realistic* inverse dynamics control [219, 210] therefore attempts to cancel the nonlinearities of (4.1) by feedback linearization based on the measured quantities and an available model of the the dynamic parameters, denoted by $\widehat{(\cdot)}$:

$$\boldsymbol{\tau} = \hat{\boldsymbol{M}}(\hat{\boldsymbol{q}})\boldsymbol{u} + \hat{\boldsymbol{n}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}), \qquad \text{where} \quad \hat{\boldsymbol{n}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}) = \hat{\boldsymbol{C}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}})\hat{\dot{\boldsymbol{q}}} + \hat{\boldsymbol{f}}(\hat{\dot{\boldsymbol{q}}}) + \hat{\boldsymbol{g}}(\hat{\boldsymbol{q}}). \tag{4.3}$$

The available model parameters may have been obtained by estimations (*e. g.*, due to unknown load) or approximations (unknown dynamic model, simplified models, *etc.*). The error quantities are denoted using $\widetilde{(\cdot)}$ as

$$\widetilde{\boldsymbol{M}}(\boldsymbol{q}, \hat{\boldsymbol{q}}) = \hat{\boldsymbol{M}}(\hat{\boldsymbol{q}}) - \boldsymbol{M}(\boldsymbol{q}), \tag{4.4a}$$
$$\tilde{\boldsymbol{n}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}, \boldsymbol{q}, \dot{\boldsymbol{q}}) = \hat{\boldsymbol{n}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}) - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{4.4b}$$

The dynamics given by application of the control law (4.3) to the system (4.1) is referred to as *inner loop*, and the vector $\boldsymbol{u} \in \mathbb{R}^n$ is the new control input that is to be determined by an outer loop. Clearly, if $\widetilde{\boldsymbol{M}} \equiv \boldsymbol{0}$, $\tilde{\boldsymbol{n}} \equiv \boldsymbol{0}$, (4.3) achieves a perfect feedback linearization and the minimum-phase nonlinear robot equations (4.1) are turned into a set of double integrators. Thus, a common choice to stabilize the resulting system is a linear state feedback with positive definite gain matrices $\boldsymbol{K}_{\text{P}}, \boldsymbol{K}_{\text{D}} \in \mathbb{R}^{n \times n}, \boldsymbol{K}_{\text{P}}, \boldsymbol{K}_{\text{D}} \succ 0$ in the outer-loop. The control law

$$\boldsymbol{u} = [\boldsymbol{K}_{\text{P}} \;\; \boldsymbol{K}_{\text{D}}]\, \boldsymbol{e} \tag{4.5}$$

then effectively acts as a PD controller.

In the realistic situation, however, the control law (4.3) inserted into (4.1) yields the following *perturbed double integrator*:

$$\ddot{\boldsymbol{q}} = (\boldsymbol{I} + \boldsymbol{\Delta}_{\mathrm{M}})\,\boldsymbol{u} + \boldsymbol{\psi}, \quad \text{where} \tag{4.6}$$

$$\boldsymbol{\Delta}_{\mathrm{M}} = \boldsymbol{M}(\boldsymbol{q})^{-1}\widetilde{\boldsymbol{M}}(\boldsymbol{q}, \hat{\boldsymbol{q}}) = \boldsymbol{M}(\boldsymbol{q})^{-1}\hat{\boldsymbol{M}}(\hat{\boldsymbol{q}}) - \boldsymbol{I}, \tag{4.7}$$

$$\boldsymbol{\psi} = \boldsymbol{M}(\boldsymbol{q})^{-1}\,\tilde{\boldsymbol{n}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}, \boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{M}(\boldsymbol{q})^{-1}\boldsymbol{\tau}_{\mathrm{dist}}. \tag{4.8}$$

As depicted in Fig. 4.2, the double integrator is perturbed by a multiplicative input uncertainty operator $\boldsymbol{\Delta}_{\mathrm{M}}$ caused by an inaccurate inertia matrix estimate $\hat{\boldsymbol{M}}$, and an inverse additive disturbance term $\boldsymbol{\psi}$. Robust approximate inverse dynamics controller design thus amounts to selecting $\boldsymbol{u}$ in the outer loop that rejects the inverse additive disturbance $\boldsymbol{\psi}$, subject to being robust w. r. t. the multiplicative input uncertainty $\boldsymbol{\Delta}_{\mathrm{M}}$ caused by the inaccurate inertia model. Hence, in what follows, *controller design* refers to the *outer loop* controller generating $\boldsymbol{u}$.

Manipulator Norm Bounds. The following properties are routinely assumed for the system (4.1)–(4.2), and the interested reader is referred to [132, 215, 210] for the details and in-depth interpretation of these assumptions.

Assumption 4.1 *(Manipulator dynamic bounds [132, 210]).* When the robot arm (4.1) is revolute, there exist known positive constants $M_{\mathrm{l}}, M_{\mathrm{u}}, C_{\mathrm{u}}, F_{\mathrm{u}}$, and g_{u} such that, for all $(t, \boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^t \times \mathbb{R}^n \times \mathbb{R}^n$, the matrices $\boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the vectors $\boldsymbol{f}(\dot{\boldsymbol{q}})$ and $\boldsymbol{g}(\boldsymbol{q})$ satisfy the following inequalities:

$$0 < M_{\mathrm{l}} \leq \|\boldsymbol{M}(\boldsymbol{q})^{-1}\| \leq M_{\mathrm{u}} < \infty, \tag{4.9a}$$

$$\|\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\| \leq C_{\mathrm{u}}\|\dot{\boldsymbol{q}}\|, \quad \|\boldsymbol{F}(\dot{\boldsymbol{q}})\| \leq F_{\mathrm{u}}\|\dot{\boldsymbol{q}}\|, \quad \|\boldsymbol{g}(\boldsymbol{q})\| \leq g_{\mathrm{u}}. \tag{4.9b}$$

The workspace of the manipulator is bounded $\|\boldsymbol{q}\|_{\mathcal{L}_\infty} \leq q_{\max}$; hence, the inertia matrix $\boldsymbol{M}(\boldsymbol{q})$ is invertible for all $\boldsymbol{q}$. Finally, $\|\dot{\boldsymbol{q}}\|_{\mathcal{L}_\infty} \leq v_{\max}$ holds due to the power limitation. ◇

As in [218], let the reference and disturbance signals fulfill the following.

Assumption 4.2 *(Exogenous signal bounds).*

1. The input disturbance is bounded from above by a known constant C_{dist} such that $\|\boldsymbol{\tau}_{\mathrm{dist}}\|_{\mathcal{L}_\infty} \leq C_{\mathrm{dist}} < \infty$.
2. The measurement noise fulfills $\|\boldsymbol{w}_i\|_{\mathcal{L}_\infty} \leq W_i$ for some known constants $W_i < \infty$.
3. The reference trajectory satisfies $\|\boldsymbol{q}_{\mathrm{d}}\|_{\mathcal{L}_\infty} \triangleq Q_{\mathrm{d}}$. ◇

Moreover, the accuracy of the dynamical model is assumed to fulfill the following properties.

Assumption 4.3 *(Model approximation).*

1. There exists a constant $\alpha > 0$ such that for all $\boldsymbol{q} \in \mathbb{R}^n$

$$\|\boldsymbol{\Delta}_{\mathrm{M}}\| = \|\boldsymbol{M}(\boldsymbol{q})^{-1}\hat{\boldsymbol{M}}(\hat{\boldsymbol{q}}) - \boldsymbol{I}\| \leq \alpha. \tag{4.10}$$

For our analysis, it is not necessarily required that $\alpha < 1$ as in [218, 13]. It will be shown, however, that robust performance enhancement is feasible in the framework only if $\alpha < 1$.

2. There exists a function $\Phi : \mathbb{R} \mapsto \mathbb{R}$ such that $\forall \boldsymbol{x}, \hat{\boldsymbol{x}} \in \mathbb{R}^{2n}: \; \|\tilde{\boldsymbol{n}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}, \boldsymbol{q}, \dot{\boldsymbol{q}})\| \leq \Phi\left(\|\boldsymbol{x}\|\right)$. Here, the bound is taken as

$$\Phi\left(\|\boldsymbol{x}\|\right) = \alpha_0 + \alpha_1 \|\boldsymbol{x}\|. \tag{4.11}$$

A detailed exposition how to obtain a suitable Φ and constants $\alpha_0, \alpha_1 \in \mathbb{R}$ is given in Appendix C.1. ◇

4.2 Review of Robust Approximate Inverse Dynamics Control

Before robust AID controllers are characterized via the double-Youla parameterization, a brief overview of the related work is given. The robust control of robot manipulators has been subject to research for decades and a wealth of methods has been developed [215, 210, 219, 132], based on a multitude of underlying approaches, *e. g.*, linear multivariable, passivity-based, sliding mode[1, 199]. Therefore, only selected work is reviewed paradigmatically by ascending dynamic model demand.

Static and model free. It has long been known that a high-gain PD controller applied directly to the nonlinear rigid-body system robustly yields uniform ultimate boundedness of the tracking error [44, 183]. Design methods for gain selection are nonetheless being investigated today [113] and a high-gain PD controller can be suitable even for fast dynamic manipulation tasks [203].

Based on crude approximation. Even if not based on inverse dynamics, robust manipulator control methods typically employ a nominal inertia model, *e. g.*, using a disturbance observer [200] or model-free time delay control [98]. That is, only the most dominant part of the manipulator dynamics is approximated and the rest treated as an uncertain disturbance input [211, 87, 116]. Taking only a diagonal estimate of the inertia matrix, the problem is handled as a linear decoupled system subject to the disturbances induced by neglected cross-coupling terms. This way, it is always possible to achieve $\alpha < 1$ in (4.10). In the authors' experience [57], this approach constitutes a viable trade-off to build a parameterization-based robustly stable learning control system that noticeably exploits domain knowledge without a detailed dynamical model of the robot manipulator at hand.

Nonlinear model-based. Taking a nonlinear dynamical model of the robot manipulator allows to accomplish a better approximate feedback linearization. Nonetheless, the model (4.6) is always imprecise to a certain extent, *i. e.*, (4.7) and (4.8) do not vanish. Classic robust manipulator control methods [215, 210, 219, 132] therefore design the outer-loop controller to ensure robust stability of the overall loop. Recent research also considers performance or optimality criteria, *e. g.*, $\mathcal{H}_\infty$ optimality [114], time-domain bounds [112], or orbital stabilization around the desired trajectory [175].

Most relevant for the developments in this thesis are the classic linear multi-variable design [218] and the Lyapunov-based robust manipulator control designs [13] reviewed in more detail next.

4.2.1 The Linear Multi-Variable Approach

A double-Youla parameterization allows to consider perturbations of both the controller and the plant models. The early robust manipulator control methods, in contrast, only exploit the characterization of all stabilizing compensators for the linear unperturbed model. Spong and Vidyasagar [218] were able to show that one can always find a linear control law stabilizing the nonlinear loop resulting from approximate inverse dynamics if the dynamic model of the robot satisfies Assumptions 4.3–4.1 and $\alpha < 1$ in (4.10). To this end, consider the control

$$\boldsymbol{u} = \ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\nu}, \tag{4.12}$$

where the additional term $\boldsymbol{\nu}$ is supposed to increase the robustness against the uncertainties due to the approximate controller (4.3). Inserting (4.12) and (4.3) into (4.1), the resulting error dynamics read

$$\dot{\boldsymbol{e}} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I} \end{bmatrix} (\boldsymbol{\eta} + \boldsymbol{\nu}), \tag{4.13}$$

where the vector $\boldsymbol{\eta}$ is induced by the uncertainty resulting from the approximate model in the closed loop:

$$\boldsymbol{\eta} = \boldsymbol{\Delta}_{\mathrm{M}} (\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\nu}) + \boldsymbol{M}^{-1}(\tilde{\boldsymbol{n}} + \boldsymbol{\tau}_{\mathrm{d}}). \tag{4.14}$$

Hence, $\boldsymbol{\eta}(\ddot{\boldsymbol{q}}_{\mathrm{d}}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\nu})$ is a nonlinear term that cannot simply be rejected as if it were an external disturbance. Instead, under the assumptions (4.10) and (4.11) with $\alpha < 1$, it is shown in [218] that for some constants b and δ the bound $\|\boldsymbol{\eta}\|_{\mathcal{L}_{\mathrm{T}\infty}} \leq (\delta\beta_1 + \alpha\beta_3)\|\boldsymbol{\eta}\|_{\mathcal{L}_{\mathrm{T}\infty}} + b$ holds, where $\|\cdot\|_{\mathcal{L}_{\mathrm{T}\infty}}$ denotes the truncated $\mathcal{L}_\infty$ norm [48]. Then if $(\delta\beta_1 + \alpha\beta_3) < 1$, the control $\boldsymbol{\nu}$, the tracking error $\boldsymbol{e}$, and the uncertainty $\boldsymbol{\eta}$ are bounded. A sequence of Youla parameters Q_k is thus designed such that $\beta_1 \to 0$ and $\beta_3 \to 1$ for $k \to \infty$. The resulting robust controller K_k is finally given from (2.6) and is generally a high-gain dynamic compensator.

4.2.2 Lyapunov-Based

Another popular approach to design a robust control input $\boldsymbol{u}$ for (4.3) is based on Lyapunov's second method [215, 210, 219, 132]. It is usually assumed that a PD controller with feedforward acceleration compensation and an additional input term $\boldsymbol{\nu}$ is employed in the outer loop,

$$\boldsymbol{u} = \ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{K}_{\mathrm{P}} (\boldsymbol{q}_{\mathrm{d}} - \boldsymbol{q}) + \boldsymbol{K}_{\mathrm{D}} (\dot{\boldsymbol{q}}_{\mathrm{d}} - \dot{\boldsymbol{q}}) + \boldsymbol{\nu}, \tag{4.15}$$

which results in the error system

$$\dot{\boldsymbol{e}} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\boldsymbol{K}_{\mathrm{P}} & -\boldsymbol{K}_{\mathrm{D}} \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I} \end{bmatrix} (\boldsymbol{\eta} + \boldsymbol{\nu}). \tag{4.16}$$

Similar to (4.13)–(4.14), the uncertainty in the closed loop is lumped in a single term $\boldsymbol{\eta}$ and one needs to select an appropriate control Lyapunov function to design $\boldsymbol{\nu}$ so as to suppress the destabilizing effects of $\boldsymbol{\eta}$. For the details and assumptions underlying the classic approach, the reader is referred to the literature [219], [210, Chap. 6.5.3],[132, Chap. 5.2].

4.2.3 Drawbacks of Established Approaches

The robot control methods based on the approaches of Sec. 4.2.1 and 4.2.2 have a number of drawbacks in common. First, all of the AID uncertainty is lumped in a single quantity $\boldsymbol{\eta}$. However, it is clear from (4.6) that an inaccurate inertia model results in multiplicative input uncertainty whereas the neglected manipulator nonlinearities induce the inverse additive disturbance $\boldsymbol{\psi}$. Therefore, the structure of (4.6) is dismissed when the analysis starts with a single term $\boldsymbol{\eta}$. Second, an outer-loop PD controller is commonly [132, 219, 215, 210] applied to stabilize (4.6) *prior to the robustness analysis.* Therefore, in all subsequent developments one has to work with an error system (4.16) where the gains $\boldsymbol{K}_{\mathrm{P}}, \boldsymbol{K}_{\mathrm{D}}$ of the controller occur in the dynamic matrix $\boldsymbol{A}$.

These aspects entail unfavorable consequences. The additional input $\boldsymbol{\nu}$ must be carefully constructed to suppress the internal nonlinear disturbance $\boldsymbol{\eta}$ and it remains prohibitive, in general, to adapt $\boldsymbol{\nu}$ on-the-fly by data-driven methods. Furthermore, in order to design $\boldsymbol{\nu}$, a scalar bound $\|\boldsymbol{\eta}\| \leq \rho(\|\boldsymbol{e}\|)$ is required, which in the classic robust manipulator control [132, 219, 215, 210] depends on the gains $\|[\boldsymbol{K}_{\mathrm{P}}, \boldsymbol{K}_{\mathrm{D}}]\|$ of the nominal PD controller. Thus, as pointed out in [1, 13], higher gains in the controller require stronger robustifying inputs $\boldsymbol{\nu}$ to correct for *seemingly* higher uncertainty. In contrast, the practitioner's approach of using a high-gain PD controller is widely known to work robustly in practice. For the Lyapunov-based design, this discrepancy has been resolved by Bascetta and Rocco in the revised design [13]. These works will be compared to the novel parameterization in Sec. 4.9.3.

4.3 Problem Formulation

In order to circumvent the drawbacks summarized in Sec. 4.2.3, we keep the uncertainty separated throughout instead of lumping the effects of (4.7)–(4.8) into a single additive term. As a sideline, we derive bounds on the perturbations to (4.6) *before* any outer-loop controller comes into place. In order to eventually keep the influence of the outer-loop controller transparent as well, we describe the uncertainty by means of the dual Youla parameter. Therefore, in the framework proposed next, by construction the uncertainty due to AID remains clearly distinguishable from the gains of the outer-loop controller.

It is unfortunately not obvious to find in the literature [132, 219, 215, 210, 127, 255] a suitable generalized plant [269, 217] description for AID, as most methods work with error dynamics (4.13) or (4.15). Therefore, the first problem we tackle is how the standard robotic bounds (4.10)–(4.11) translate into a generalized setup (Fig. 4.3) without lumping the effects of (4.7)–(4.8) into a single additive term.

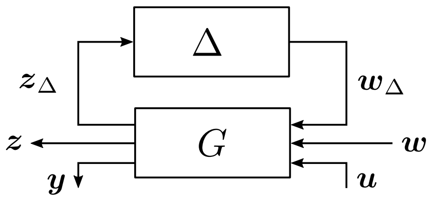


Figure 4.3: Generalized plant setup with uncertainty Δ in open loop

Problem 4.1 *(Generalized plant formulation of approximate inverse dynamics).* Given the system (4.1) and the control law (4.3) with Ass. 4.1–4.3 fulfilled, reformulate the approximate inverse dynamics (4.6) and the corresponding bounds as a system

$$G\colon \begin{cases} \boldsymbol{z}_\Delta = G_{z_\Delta w_\Delta}\boldsymbol{w}_\Delta + G_{z_\Delta w}\boldsymbol{w} + G_{z_\Delta u}\boldsymbol{u} \\ \boldsymbol{z} = G_{zw_\Delta}\boldsymbol{w}_\Delta + G_{zw}\boldsymbol{w} + G_{zu}\boldsymbol{u} \\ \boldsymbol{y} = G_{yw_\Delta}\boldsymbol{w}_\Delta + G_{yw}\boldsymbol{w}, \end{cases} \tag{4.17}$$

such that the uncertainty (4.7) and (4.8) acting on the nominal double integrator $\ddot{\boldsymbol{q}} = \boldsymbol{u}$ is described by a linear fractional transformation as shown in Fig. 4.3. That is, the uncertainty is captured by an unknown norm-bounded Δ operating on the signals $\boldsymbol{z}_\Delta$ to yield the input perturbation $\boldsymbol{w}_\Delta$,

$$\boldsymbol{w}_\Delta(t) = \Delta(\boldsymbol{x}(t), t)\, \boldsymbol{z}_\Delta(t), \quad \text{where } \exists\delta\, \forall t\colon \|\Delta(\boldsymbol{x}(t), t)\| \leq \delta. \tag{4.18}$$

◇

In the next step, we analyze how these bounds affect the closed-loop system when the nominal outer-loop controller is applied.

Problem 4.2 *(Dual Youla bound of uncertainty under approximate inverse dynamics).* Given the system (4.17)–(4.18) and a nominal outer-loop stabilizing controller K_0, characterize by means of a dual Youla bound $\|S\|_\infty \leq \gamma_{\mathrm{S}}$ the worst-case dynamic perturbation in closed-loop w. r. t. the nominal controlled plant $\ddot{\boldsymbol{q}} = \boldsymbol{u}$. ◇

Once Problem 4.2 is solved, the solution to the following problem is immediate.

Problem 4.3 *(Double-Youla for robust approximate inverse dynamics).* For a given robot under approximate inverse dynamics as in Probs. 4.1 and 4.2, characterize in terms of the parameter set $\mathcal{Q}$ a subset of robustly stabilizing controllers $\mathcal{K}_{\mathrm{R}} \subset K(Q)$. ◇

4.4 Generalized Plant Formulation of Uncertainty Bounds

4.4.1 Reformulation of AID Uncertainty to $G - \Delta$ Structure

We first tackle Prob. 4.1 to obtain an uncertainty formulation in the form of the generalized plant depicted in Fig. 4.3. For convenience, let us restate (4.6) in the form

$$\ddot{\boldsymbol{q}} = \boldsymbol{u} + \boldsymbol{w}_\Delta^{\mathrm{U}} + \boldsymbol{w}_{\mathrm{dist}}, \tag{4.19}$$

where external disturbances $\boldsymbol{w}_{\text{dist}} \triangleq \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{\tau}_{\text{dist}}$, and the vector of internal model uncertainty $\boldsymbol{w}_{\Delta}^{\text{U}} = \boldsymbol{w}_{\Delta}^{\text{M}} + \boldsymbol{w}_{\Delta}^{\psi}$ is summing up both inertia-induced uncertainty $\boldsymbol{w}_{\Delta}^{\text{M}}$ and signal uncertainty $\boldsymbol{w}_{\Delta}^{\psi}$:

$$\boldsymbol{w}_{\Delta}^{\text{M}} \triangleq \boldsymbol{\Delta}_{\text{M}}(\boldsymbol{q},\hat{\boldsymbol{q}})\,\boldsymbol{u}, \qquad\qquad \boldsymbol{w}_{\Delta}^{\psi} \triangleq \boldsymbol{M}^{-1}(\boldsymbol{q})\,\tilde{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}},\hat{\boldsymbol{q}},\hat{\dot{\boldsymbol{q}}}).$$

Subsequently, the dependency on joint positions and velocities is dropped for brevity in notation. With Ass. 4.2 the summand $\boldsymbol{w}_{\text{dist}}$ is bounded by $\|\boldsymbol{w}_{\text{dist}}\|_{\mathcal{L}_\infty} \leqslant M_{\text{u}}C_{\text{dist}}$ and depending neither on state nor on control; $\boldsymbol{w}_{\text{dist}}$ is therefore considered as external disturbance input. Note that by (4.10)–(4.11), both $\boldsymbol{w}_{\Delta}^{\text{M}}$ and also $\boldsymbol{w}_{\Delta}^{\psi}$ are non-dynamic, time-varying nonlinear uncertainties; it is therefore possible to use a norm-bounded uncertainty description (4.18), where the upper bound on the induced matrix norm $\|\Delta\| \triangleq \max_{\boldsymbol{z}_\Delta(t)\neq 0} \frac{\|\Delta \boldsymbol{z}_\Delta(t)\|}{\|\boldsymbol{z}_\Delta(t)\|} \leq \delta$ is satisfied for all times t [177, Sec. 2.3.1]. This corresponds to the maximum amplification over all input directions $\boldsymbol{z}_\Delta$ and is key to solving Problem 4.1.

Theorem 4.1 (Structured uncertainty under approximate inverse dynamics). Under the conditions of Problem 4.1, the perturbed double integrator system (4.6) can be conservatively reformulated as a G-Δ structure depicted in Fig. 4.4, where $\boldsymbol{w}_\Delta \triangleq \operatorname{col}(\boldsymbol{w}_{\Delta}^{\text{M}}, \boldsymbol{w}_{\Delta}^{\psi})$, $\boldsymbol{z}_\Delta \triangleq \operatorname{col}(\boldsymbol{u}, z_{\Delta}^{\text{f}}, \boldsymbol{q}, \dot{\boldsymbol{q}})$, $z_{\Delta}^{\text{f}} \equiv 1$ and $\Delta(t) \in \mathcal{D}_\Delta$ with the perturbation set

$$\mathcal{D}_\Delta = \left\{ \boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\Delta}_{\text{M}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Delta}_{\Psi} \end{bmatrix} : \boldsymbol{\Delta}_{\text{M}} \in \mathbb{R}^{n\times n}, \boldsymbol{\Delta}_{\Psi} \in \mathbb{R}^{n\times(2n+1)}, \|\boldsymbol{\Delta}_{\text{M}}\| \leq \alpha_{\text{M}}, \|\boldsymbol{\Delta}_{\Psi}\| \leq \alpha_{\Psi} \right\}. \tag{4.20}$$

The bounds α_{M} and α_Ψ are scalars given by (B.3).

Proof: The derivation is given in Appendix B.1. ■

By adopting the uncertainty structure (4.20), the separation of the two uncertainty sources is preserved because the summation of $\boldsymbol{u}$, $\boldsymbol{w}_{\Delta}^{\text{M}}$ and $\boldsymbol{w}_{\Delta}^{\psi}$ in (4.19) is captured by the signal interconnection in the generalized plant G as depicted in Fig. 4.4. Therefore, this structure is carried into the dual Youla operator in the sequel.

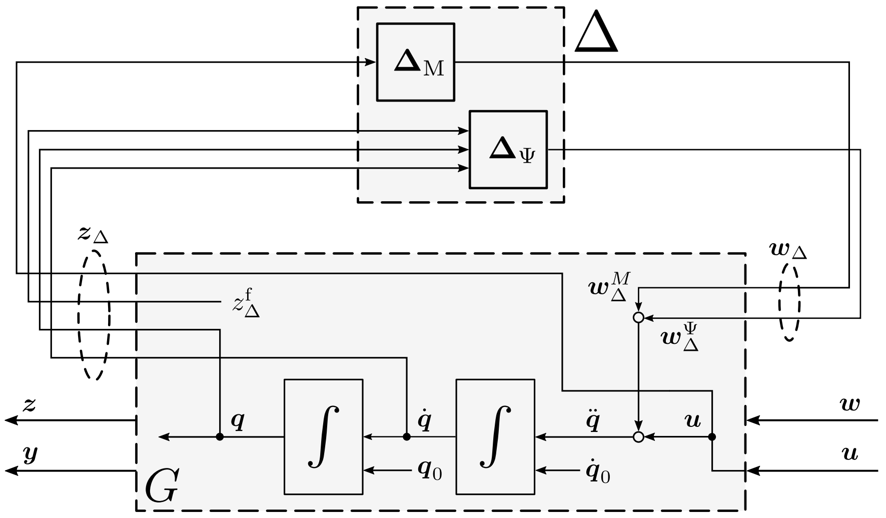
Remark 4.1 *(Uncertainty separation).* In order to obtain such a description, one could also work with $\boldsymbol{w}_{\Delta}^{\text{U}} = \Delta^{\text{U}}\boldsymbol{z}_\Delta$ instead of collecting the terms $\boldsymbol{w}_{\Delta}^{\text{M}}$, $\boldsymbol{w}_{\Delta}^{\psi}$ of (4.19) separately in the vector $\boldsymbol{w}_\Delta$. Then, however, the result is unstructured with an overly conservative bound on the uncertainty matrix $\boldsymbol{\Delta}$. ◁

4.4.2 Example: State-Space Realization of Generalized Plant Configuration for Tracking Control

This section only serves to demonstrate the specific form of the realization for a tracking control configuration and is included for completeness.

Let the state-space description of (4.17) be given as

$$G : \left[\begin{array}{c|ccc} \boldsymbol{A}_{11} & \boldsymbol{B}_{11} & \boldsymbol{B}_{12} & \boldsymbol{B}_{13} \\ \hline \boldsymbol{C}_{11} & \boldsymbol{D}_{11} & \boldsymbol{D}_{12} & \boldsymbol{D}_{13} \\ \boldsymbol{C}_{21} & \boldsymbol{D}_{21} & \boldsymbol{D}_{22} & \boldsymbol{D}_{23} \\ \boldsymbol{C}_{31} & \boldsymbol{D}_{31} & \boldsymbol{D}_{32} & \boldsymbol{0} \end{array}\right], \tag{4.21}$$



Δ
$\boldsymbol{\Delta}_{\mathrm{M}}$
$\boldsymbol{\Delta}_{\Psi}$
$\boldsymbol{z}_{\Delta}$
z_{Δ}^{f}
$\boldsymbol{w}_{\Delta}$
$\boldsymbol{w}_{\Delta}^{M}$
$\boldsymbol{w}_{\Delta}^{\Psi}$
$\boldsymbol{z}$
$\boldsymbol{y}$
$\boldsymbol{q}$
$\dot{\boldsymbol{q}}$
$\ddot{\boldsymbol{q}}$
$\boldsymbol{u}$
$\boldsymbol{w}$
$\boldsymbol{u}$
$\boldsymbol{q}_0$
$\dot{\boldsymbol{q}}_0$
G

Figure 4.4: Generalized plant interconnection of a robot manipulator under approximate inverse dynamics control with uncertainties "pulled out" [269]. This figure only depicts the perturbed double integrator system and the uncertainty, *i. e.*, the G-Δ structure suitable for subsequent analyses. The precise configuration of the control and performance channels $\boldsymbol{y}, \boldsymbol{z}$ depends on the manipulator control goal at hand, *e. g.*, tracking or impedance control. A tracking control example is given in Sec. 4.9 with the corresponding state-space realization in Sec. 4.4.2.

with $(\boldsymbol{A}_{11}, \boldsymbol{B}_{13})$ and $(\boldsymbol{A}_{11}, \boldsymbol{C}_{31})$ stabilizable and detectable pairs, respectively.

In this section, the entries of the matrix are derived for the case of a two-degree-of-freedom (feedforward/feedback) tracking control design (Fig. 4.4 specialized according to Remark 4.8 and Remark 4.9). In this case, (4.21) is determined from

$$\boldsymbol{A}_{11} = \begin{bmatrix} \mathbf{0}_n\, \boldsymbol{I}_n \\ \mathbf{0}_n\, \mathbf{0}_n \end{bmatrix},\ \boldsymbol{B}_{11} = \begin{bmatrix} \mathbf{0}_n\, \mathbf{0}_n \\ \boldsymbol{I}_n\, \boldsymbol{I}_n \end{bmatrix},\ \boldsymbol{B}_{12} = \begin{bmatrix} \mathbf{0}_{n\times(5n+1)}\, \mathbf{0}_n \\ \mathbf{0}_{n\times(5n+1)}\, \boldsymbol{I}_n \end{bmatrix},\ \boldsymbol{B}_{13} = \begin{bmatrix} \mathbf{0}_n \\ \boldsymbol{I}_n \end{bmatrix}, \tag{4.22}$$

$$\boldsymbol{C}_{11} = \begin{bmatrix} \mathbf{0}_{(n+1)\times 2n} \\ \mathbf{0}_n \quad \boldsymbol{I}_n \end{bmatrix},\ \boldsymbol{C}_{21} = \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0}_n \\ \multicolumn{2}{c}{\mathbf{0}_{n\times 2n}} \\ \boldsymbol{I}_n & \mathbf{0}_n \end{bmatrix}, \boldsymbol{C}_{31} = \begin{bmatrix} \boldsymbol{I}_{2n} \\ \mathbf{0}_{3n\times 2n} \end{bmatrix}, \tag{4.23}$$

$$\boldsymbol{D}_{11} = \mathbf{0}_{(2n+1)\times 2n},\ \boldsymbol{D}_{12} = \begin{bmatrix} \mathbf{0}_{n\times(6n+1)} \\ 1 \quad \mathbf{0}_{1\times 6n} \\ \mathbf{0}_{n\times(6n+1)} \end{bmatrix},\ \boldsymbol{D}_{13} = \begin{bmatrix} \boldsymbol{I}_n \\ \mathbf{0}_{(n+1)\times n} \end{bmatrix}, \tag{4.24}$$

$$\boldsymbol{D}_{21} = \mathbf{0}_{3n\times 2n},\ \boldsymbol{D}_{22} = \begin{bmatrix} \mathbf{0}_{n\times 1} & -\boldsymbol{I}_n & \mathbf{0}_{n\times 5n} \\ \multicolumn{3}{c}{\mathbf{0}_{2n\times(6n+1)}} \end{bmatrix},\ \boldsymbol{D}_{23} = \left[\mathbf{0}_n^\top, \boldsymbol{I}_n^\top, \mathbf{0}_n^\top\right]^\top, \tag{4.25}$$

$$\boldsymbol{D}_{31} = \mathbf{0}_{5n\times 2n},\ \boldsymbol{D}_{32} = \begin{bmatrix} \mathbf{0}_{2n\times 1} & \mathbf{0}_{2n\times 3n} & \boldsymbol{I}_{2n} & \mathbf{0}_{2n\times n} \\ \mathbf{0}_{3n\times 1} & \boldsymbol{I}_{3n} & \multicolumn{2}{c}{\mathbf{0}_{3n}} \end{bmatrix}. \tag{4.26}$$

Remark 4.2 *(Obtaining numeric values defining the plant interconnection).* In this section, for demonstration and reference, the system (4.21) is given explicitly by (4.22)–(4.26) for a tracking control example. In a practical implementation, however, it is inconvenient and error-prone to establish the matrices by hand. Instead, one may use a computer program to implement the interconnection visually and then obtain the matrices of (4.21), subsequently required to calculate the controller, numerically. For example, in MATLAB/Simulink, the matrices can be obtained by (pseudo-)linearizing the plant model using `linmod` and subsequent partitioning of the matrix according to the I/O dimensions. $\triangleleft$

4.5 Dual Youla Description of Closed-Loop Uncertainty

We now calculate a realization of the uncertain dual Youla parameter S when the nominal controller K_0 is applied to stabilize the uncertain plant $\mathcal{F}_u(G, \Delta)$. Recall that in (2.17)–(2.18) all systems stabilized by K_0 are parameterized as $\mathcal{G}$. Consequently, if $G_{yu}(\Delta) = \mathcal{F}_u(G, \Delta)$ is robustly stabilized by K_0, *i. e.*, $G_{yu}(\Delta) \in \mathcal{G}$ for all $\Delta \in \mathcal{D}_\Delta$, then equivalently there exists an uncertain $S \in \mathcal{S}_\Delta \subseteq \mathcal{S}$. Key to our approach is the following explicit connection of the open-loop uncertainty description in terms of Δ and in terms of the S obtained in closed-loop with K_0 applied [160, Th. 3.4], [161]:

$$G_{yu}(S) = G_{yu}(\Delta) \quad \Leftrightarrow \quad S(\Delta) = T_{\Delta,21}\Delta\left(I - T_{\Delta,11}\Delta\right)^{-1} T_{\Delta,12} = \mathcal{F}_u\left(T_\Delta, \Delta\right). \tag{4.27}$$

In (4.27), T_Δ refers to the mapping from the inputs $\mathrm{col}(\boldsymbol{w}_\Delta, \boldsymbol{s})$ to the outputs $\mathrm{col}(\boldsymbol{z}_\Delta, \boldsymbol{r})$. In operator notation, T_Δ is expressed as

$$T_\Delta = \begin{bmatrix} T_{\Delta,11} & T_{\Delta,12} \\ T_{\Delta,21} & T_{\Delta,22} \end{bmatrix} = \begin{bmatrix} G_{z_\Delta w_\Delta} + G_{z_\Delta u} U_0 \tilde{M}_0 G_{y w_\Delta} & G_{z_\Delta u} M_0 \\ \tilde{M}_0 G_{y w_\Delta} & 0 \end{bmatrix}. \tag{4.28}$$

Remark 4.3. As Niemann points out in [160], "[...] S depends only on the uncertain block Δ and the coprime factors." For the purpose of this thesis, however, we would like to emphasize the fact that (4.27) and (4.28) also depend on the interconnection used in the generalized plant (4.17). This is an important feature of our method: in contrast to the robust control approaches summarized in Sec. 4.2, by the derivation in Sec. 4.4.1, we keep the inertia-induced uncertainty $\boldsymbol{w}_\Delta^{\mathrm{M}}$ completely separated from the nonlinear inputs $\boldsymbol{w}_\Delta^{\psi}$, thus reducing the conservatism of the resulting S. $\triangleleft$

With the generalized plant of Fig. 4.4 and the bounded perturbation set $\mathcal{D}_\Delta$ from (4.20), we obtain the following result.

Theorem 4.2 (Realization of dual Youla uncertainty set). Consider the system (4.1) with the approximate inverse dynamics controller (4.3) such that Ass. 4.1–4.3 are fulfilled. According to Sec. 4.4.1, the resulting system dynamics is conservatively covered by the linear, stabilizable and detectable plant (4.17) with a state-space realization (4.21) and the uncertainty structure (4.20). Let a nominal linear controller

$$K_0 \triangleq \left[\begin{array}{c|c} \boldsymbol{A}_{\mathrm{K}} & \boldsymbol{B}_{\mathrm{K}} \\ \hline \boldsymbol{C}_{\mathrm{K}} & \boldsymbol{D}_{\mathrm{K}} \end{array}\right] \tag{4.29}$$

be applied to the outer loop, with the pairs $(\boldsymbol{A}_{\mathrm{K}}, \boldsymbol{B}_{\mathrm{K}})$ stabilizable and $(\boldsymbol{A}_{\mathrm{K}}, \boldsymbol{C}_{\mathrm{K}})$ detectable. Let the state feedback gains $\boldsymbol{F}_{\mathrm{G}}$ and $\boldsymbol{F}_{\mathrm{K}}$ be designed such that $\boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{F}_{\mathrm{G}}$ and $\boldsymbol{A}_{\mathrm{K}} + \boldsymbol{B}_{\mathrm{K}}\boldsymbol{F}_{\mathrm{K}}$

are stable. Then, the set of uncertain dual Youla operators is given by

$$\mathcal{S}_\Delta = \left\{ S(\Delta) : \left[\begin{array}{c|c} \boldsymbol{A}_\mathrm{S} & \boldsymbol{B}_\mathrm{S} \\ \hline \boldsymbol{C}_\mathrm{S} & \boldsymbol{D}_\mathrm{S} \end{array}\right], \forall \Delta \in \mathcal{D}_\Delta \right\}, \tag{4.30}$$

where

$$\begin{aligned}
\boldsymbol{A}_\mathrm{S} &= \begin{bmatrix} \boldsymbol{B}_{13}\boldsymbol{D}_\mathrm{K}\boldsymbol{C}_{31} + \boldsymbol{A}_{11} & \boldsymbol{B}_{13}\boldsymbol{C}_\mathrm{K} & \boldsymbol{A}_{\mathrm{S},13} & \boldsymbol{A}_{\mathrm{S},14} \\ \boldsymbol{B}_\mathrm{K}\boldsymbol{C}_{31} & \boldsymbol{A}_\mathrm{K} & \boldsymbol{B}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{C}} & \boldsymbol{A}_{\mathrm{S},24} \\ \boldsymbol{0} & \boldsymbol{0} & \bar{\boldsymbol{A}} & \bar{\boldsymbol{B}}\boldsymbol{C}_{11} + \bar{\boldsymbol{B}}\boldsymbol{D}_{13}\boldsymbol{F}_\mathrm{G} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_{13}\boldsymbol{F}_\mathrm{G} + \boldsymbol{A}_{11} \end{bmatrix}, \\
\boldsymbol{B}_\mathrm{S} &= \begin{bmatrix} \boldsymbol{B}_{\mathrm{S},11} \\ \boldsymbol{B}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13} \\ \bar{\boldsymbol{B}}\boldsymbol{D}_{13} \\ \boldsymbol{B}_{13} \end{bmatrix}, \\
\boldsymbol{C}_\mathrm{S} &= \begin{bmatrix} \boldsymbol{C}_{31} & -\boldsymbol{F}_\mathrm{K} & \boldsymbol{D}_{31}\Delta\bar{\boldsymbol{C}} & \boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{C}_{11} + \boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13}\boldsymbol{F}_\mathrm{G} \end{bmatrix}, \\
\boldsymbol{D}_\mathrm{S} &= \boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13}, \\
\boldsymbol{A}_{\mathrm{S},13} &= \boldsymbol{B}_{11}\Delta\bar{\boldsymbol{C}} + \boldsymbol{B}_{13}\boldsymbol{D}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{C}}, \\
\boldsymbol{A}_{\mathrm{S},14} &= \boldsymbol{B}_{11}\Delta\bar{\boldsymbol{D}}\boldsymbol{C}_{11} + \boldsymbol{B}_{11}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13}\boldsymbol{F}_\mathrm{G} \\
&\quad + \boldsymbol{B}_{13}\boldsymbol{D}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{C}_{11} + \boldsymbol{B}_{13}\boldsymbol{D}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13}\boldsymbol{F}_\mathrm{G}, \\
\boldsymbol{A}_{\mathrm{S},24} &= \boldsymbol{B}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{C}_{11} + \boldsymbol{B}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13}\boldsymbol{F}_\mathrm{G}, \\
\boldsymbol{B}_{\mathrm{S},11} &= \boldsymbol{B}_{11}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13} + \boldsymbol{B}_{13}\boldsymbol{D}_\mathrm{K}\boldsymbol{D}_{31}\Delta\bar{\boldsymbol{D}}\boldsymbol{D}_{13},
\end{aligned} \tag{4.31}$$

and the expressions for $\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}, \bar{\boldsymbol{D}}$ are given by the realization (B.7).

Proof: The proof is given in Appendix B.2. ■

From the realization (4.31), it becomes apparent that S is much more involved than the diagonal uncertainty structure of Δ. Hence, standard robustness tools [12] such as μ-analysis [266] or worst-case gain assessment [170] cannot be directly applied. In order to solve Prob. 4.2, it is nonetheless required to find an estimate $\hat{\gamma}_\mathrm{S}$ of the worst-case gain

$$\gamma_\mathrm{S} \triangleq \sup_{S \in \mathcal{S}_\Delta} \|S\|_\infty. \tag{4.32}$$

To this end, we propose to adopt a randomized approach [236] allowing for the following probabilistic worst-case assessment.

Proposition 4.1 (Probabilistic dual Youla uncertainty bound). Assign with $p \in (0,1)$, $\delta \in (0,1)$ the desired probability level such that $P(\|S(\Delta)\|_\infty \leq \hat{\gamma}_\mathrm{S}) \geq p$, $\forall \Delta \in \mathcal{D}_\Delta$ holds with probability $1-\delta$. The corresponding gain bound $\hat{\gamma}_\mathrm{S}$ is obtained by the empirical maximum $\hat{\gamma}_\mathrm{S} = \max\limits_{i=1,\ldots,n_\mathrm{S}} \|S_i\|_\infty$, using $n_\mathrm{S} \geq \left\lceil \frac{\ln 1/\delta}{\ln 1/p} \right\rceil$ iid samples from $\mathcal{D}_\Delta$.

Proof: Direct from the definition of probabilistic worst-case performance assessment using a randomized algorithm [236]. ■

A procedure to obtain $\hat{\gamma}_\mathrm{S}$ correspondingly is summarized in Algorithm 4.1.

Remark 4.4 *(Realization of dual Youla parameter for norm-bounded memoryless uncertainty).* The state-space realization for the dual Youla parameter S according to (4.31) is a fairly general result by itself: it covers, under the usual assumptions for double-Youla parameterizations, the linear generalized plant interconnection of the form (4.21) subject to non-dynamic (memoryless) norm-bounded uncertainty Δ, while allowing for any linear central controller K_0 in the Q-parameterization. Thus, the result extends [160] in that, due to the symbolic derivation, the realization of S given in (4.31) is explicit in the state-space matrices and hence suitable for numeric computation of the $\mathcal{H}_\infty$-norm. By contrast, in our simulation study reported in Sec. 4.9, a direct computation by (4.27) quickly becomes inaccurate: the numeric inversion of systems easily yields badly conditioned matrices, and a straightforward implementation of S by formula (4.27) using MATLAB state-space objects `ss` yields $\|S\|_\infty = \infty$ although S is stable. $\lhd$

4.6 Characterization of Robustly Stabilizing Controllers: Double-Youla Parameterization

With S according to (4.27) and (4.31), respectively, the control loop can now be described as shown in Fig. 4.5. The loop $T(S)$ is stabilized but uncertain and the controller is determined by some $Q \in \mathcal{Q}$. Writing out the matrix transfer operator $T_{zw}(S, Q) = \mathcal{F}_\ell\left(T(S), Q\right)$ from references $\boldsymbol{w}$ to outputs $\boldsymbol{z}$, one obtains

$$T_{zw}(S, Q) = T_{11}(S) + T_{12}(S)Q\left(I - T_{22}(S)Q\right)^{-1} T_{21}(S).$$

The following sufficient condition is established to ensure the overall stability of the scheme.

Algorithm 4.1 Calculation of worst-case gain estimate $\hat{\gamma}_\mathsf{S}$

```
Input: $p, \delta$ – desired probability levels
       $\alpha_\mathsf{M}, \alpha_\Psi$ – norm bounds (B.3)
       $G$ – realization (4.21) of $G$, e.g., according to Sec. 4.4.2
       $K_0$ – realization of nominal controller
       $\boldsymbol{F}_\mathsf{G}$ – stabilizing state feedback for plant
       $\boldsymbol{F}_\mathsf{K}$ – stabilizing state feedback for controller
1  $n_\mathsf{S} \leftarrow \left\lceil \left(\ln \frac{1}{\delta} / \ln \frac{1}{p}\right) \right\rceil$
2  for $i = 1, 2, 3, \ldots, n_\mathsf{S}$ do
3      $\boldsymbol{\Delta}_i^\mathsf{M} \leftarrow$ draw sample uniformly over $\mathcal{B}_2^{n\times n}(\alpha_\mathsf{M})$
4      $\boldsymbol{\Delta}_i^\Psi \leftarrow$ draw sample uniformly over $\mathcal{B}_2^{n\times(2n+1)}(\alpha_\Psi)$
5      $S_i \leftarrow S\left(\mathrm{blkdiag}\left(\boldsymbol{\Delta}_i^\mathsf{M}, \boldsymbol{\Delta}_i^\Psi\right)\right)$ by (4.31)
6  end for
7  $\hat{\gamma}_\mathsf{S} \leftarrow \max_{i=1,\ldots,n_\mathsf{S}} \|S_i\|_\infty$
Output: $\hat{\gamma}_\mathsf{S}$
```

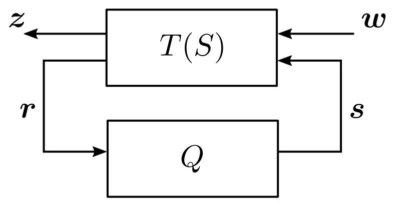


Figure 4.5: Double-Youla parameterization under external inputs with the pre-stabilized uncertain closed loop $T(S)$ and controller parameterization via Q

Lemma 4.1 (Double Youla under exogenous disturbance and plant uncertainty). Under the previous assumptions, for all $\Delta \in \mathcal{D}_\Delta$ there exists correspondingly an $S \in \mathcal{S}_\Delta$. Denote by $\mathcal{Q}$ the set of finite-gain stable, admissible parameters $Q \in \mathcal{Q}$. Referring to Fig. 4.5, $T_{\mathrm{zw}}(S, Q)$ is robustly stable if both

(i) K_0 robustly stabilizes the uncertain plant $\mathcal{F}_u(G, \Delta)$, *i. e.*,

$$\forall \Delta \in \mathcal{D}_\Delta \colon \begin{bmatrix} I & -\begin{bmatrix} 0 & 0 \\ 0 & K_0 \end{bmatrix} \\ -\mathcal{F}_u(G, \Delta) & I \end{bmatrix}^{-1} \in \mathcal{RH}_\infty, \tag{4.33}$$

(ii) the loop (Q, S) is stable and $\forall Q \in \mathcal{Q}, \forall S \in \mathcal{S}_\Delta \colon \quad \|Q\|_\infty \|S\|_\infty < 1$.

Proof: The proof is provided in Appendix B.3. ■

By the condition (ii), it is immediate to characterize a set of controller parameters $\mathcal{Q}$ that preserves robust stability and is suitable for a wide range of performance enhancement methods.

Theorem 4.3 (Set of robust approximate inverse dynamics controllers). Under the assumptions of Thm. 4.2, let the worst-case gain (4.32) be estimated by $\hat{\gamma}_{\mathrm{S}} \overset{\geq}{\to} \gamma_{\mathrm{S}}$. If $\hat{\gamma}_{\mathrm{S}}$ is finite, K_0 is robustly stabilizing. A subset of all robustly stabilizing controllers is then given by

$$\mathcal{K}_{\mathrm{R}} = \left\{ K(Q) \colon (2.5)\text{–}(2.6) \text{ and } \|Q\|_\infty < 1/\hat{\gamma}_{\mathrm{S}} \right\} \tag{4.34}$$

and every controller $K \in \mathcal{K}_{\mathrm{R}}$ stabilizes the nonlinear system (4.6)–(4.8).

Proof: According to Thm. 4.1, the uncertain nonlinear dynamics (4.6)–(4.8) are conservatively covered by the uncertainty structure $\mathcal{D}_\Delta$ from (4.20). As by Thm. 4.2, $\mathcal{D}_\Delta$ corresponds to $\mathcal{S}_\Delta$ when the nominal controller K_0 is applied. With Prop. 4.1, $\hat{\gamma}_{\mathrm{S}}$ is the worst-case gain over all $\mathcal{S}_\Delta$ by letting $p \to 1, \delta \to 0$. The result is then a direct consequence of Lemma 4.1. ■

On the one hand, the first condition (i) of Lemma 4.1, *i. e.*, robust stability of the uncertain loop with only the nominal controller applied, may seem quite restrictive. On the other hand, the stability of the (Q, S) loop is ensured by a small-gain argument; thus, it also guarantees internal stability when Q is a nonlinear or time-varying stable operator with an appropriately defined stability notion [48]. Hence, due to the double-Youla parameterization, a variety of advanced methods can be used for manipulator control design in the proposed rigorous robust stability framework, as motivated in Fig. 4.1.

Remark 4.5 *(Model complexity trade-off).* Thm. 4.3 provides a way to *quantify* the trade-off between the accuracy of the available manipulator model and the amount of controller enhancement permissible without sacrificing robust stability of the system: a very imprecise robot model will yield large γ_S and consequently $\mathcal{K}_\mathrm{R} \to \{K_0\}$ as $\mathcal{Q} \to \emptyset$. As for the other extreme, a perfect model allows for a true feedback linearization; hence $S = 0$, and by $\hat{\gamma}_\mathrm{S} \to 0$ Thm. 4.3 recovers the set of all controllers that stabilize a double integrator. $\triangleleft$

Remark 4.6 *(Robust stability under nominal control).* If the nominal controller K_0 does not stabilize the system under all perturbations $\Delta \in \mathcal{D}_\Delta$, then $\gamma_\mathrm{S} = \infty$, *i. e.*, there is some unstable $S(\Delta)$. In such cases, it may via Q still be possible to adaptively stabilize the (Q, S) loop of the double-Youla parameterization [234]. We do not pursue such approaches further but restrict attention to a more robust setting: here, stabilization under all perturbations $\Delta \in \mathcal{D}_\Delta$ by means of all controllers (4.34) is the asset allowing for straightforward online performance enhancement. $\triangleleft$

4.7 Special Case: Static Nominal Control

In this section, we specialize the main result to the very commonly used controller of the PD-type (4.5) or (4.15) as initial controller K_0. To this end, under the previous assumptions, a Q-parameterization can be constructed as follows.

Proposition 4.2 (Central J for static nominal control). A realization of the central system J in order to build a Q-parameterization around a static controller for the system (4.22) is

$$J\colon \left[\begin{array}{c|cc} \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{F}_\mathrm{G} & \boldsymbol{0} & \boldsymbol{B}_{13} \\ \hline \boldsymbol{F}_\mathrm{G} - \boldsymbol{D}_0\boldsymbol{C}_{31} & \boldsymbol{D}_0 & \boldsymbol{I} \\ -\boldsymbol{C}_{31} & \boldsymbol{I} & \boldsymbol{0} \end{array}\right], \ \boldsymbol{x}_\mathrm{J}(0) = \boldsymbol{0}. \tag{4.35}$$

Proof. The system (4.35) is a straightforward specialization of Prop. 2.5 to the case of a strictly proper plant. $\square$

A block diagram of (4.35) is depicted in Fig. 4.6.

Remark 4.7 *(Components of control input).* Comparing to Sec. 4.2, the overall control signal corresponds exactly to the standard ansatz for the outer loop (4.15) but the additional signal $\boldsymbol{\delta u}$ is generated differently from the "robust-control term" [13] $\boldsymbol{\nu}$; precisely, model-based in the Q-parameterization generated from (4.35). $\triangleleft$

Note that the central system J inherits with (4.35) a separation structure that is very useful for implementation on hardware: the two-port system (4.35) is given directly by the static nominal controller $\boldsymbol{D}_0$, and some dynamic augmentation to generate the signal $\boldsymbol{r}$.

Next, the influence of the stabilizing gain $\boldsymbol{F}_\mathrm{G}$ is discussed. Recall that this matrix is a design parameter, introduced to construct a coprime factorization (A.21). Referring to the realization of S in Corollary 4.1, from the block-triangular structure of $\boldsymbol{A}_\mathrm{S}$ it is clear that the stability of S is determined by the block matrices on the diagonal. The block $\boldsymbol{A}_{\mathrm{S},11} = \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{C}_{31}$ is the nominal closed-loop transfer function matrix, consisting of the (unperturbed) double integrator system and the PD controller $\boldsymbol{D}_0$. The second block $\bar{\boldsymbol{A}}$

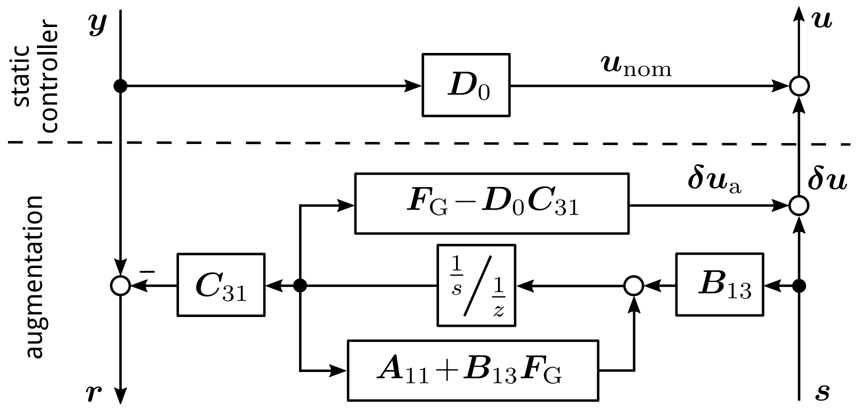


Figure 4.6: Central system J to generate a Q-parameterization, based on a static nominal controller $K_0 \triangleq \boldsymbol{D}_0$. Note that the augmentation simplifies further by choosing $\boldsymbol{F}_\mathrm{G} = \boldsymbol{D}_0 \boldsymbol{C}_{31}$. In this case, $\boldsymbol{\delta u}_\mathrm{a} = 0$ and consequently $\boldsymbol{u} = \boldsymbol{u}_\mathrm{nom} + \boldsymbol{s}$.

depends on the perturbation $\boldsymbol{\Delta}$ and the controller $\boldsymbol{D}_0$, but not on $\boldsymbol{F}_\mathrm{G}$. Hence, the importance of using a robust controller in the central system cannot be relaxed by design of $\boldsymbol{F}_\mathrm{G}$. Finally $\boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{F}_\mathrm{G}$ is the dynamic matrix of the coprime factor M_0, which is a stability matrix by requirement (see Thm. 4.2). It is known that S can be interpreted as a frequency-shaped difference between the actual plant $G(\Delta)$ and the nominal plant G_0 in the bandwidth of the nominal operating frequencies [234, Sec. 2.7, Rem. 6]. Therefore, we advocate to choose

$$\boldsymbol{F}_\mathrm{G} = \boldsymbol{D}_0 \boldsymbol{C}_{31} \tag{4.36}$$

in order to obtain $\boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{F}_\mathrm{G} = \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{C}_{31}$. Referring to Fig. 4.6, note that by this choice the central system J is further simplified: in this case, the filtered output $\boldsymbol{s}$ is directly passed to the controlled system, as

$$\boldsymbol{u} = \boldsymbol{u}_\mathrm{nom} + \boldsymbol{\delta u}_\mathrm{a} + \boldsymbol{s} \overset{(4.35)\text{ with }\boldsymbol{F}_\mathrm{G}=\boldsymbol{D}_0\boldsymbol{C}_{31}}{=} \boldsymbol{u}_\mathrm{nom} + \boldsymbol{s}.$$

In this case, only the filtered output $\boldsymbol{s}$ is added to the control input and the proposed framework becomes particularly simple to implement on a robotic platform: opposed to the standard Youla parameterization [269], no observer-based central controller is required. Instead, *robot manipulators that are already driven by a PD controller can be augmented to generate the parameterization.* The uncertainty quantification in the closed loop becomes nonetheless feasible as expressed by $\hat{\gamma}_\mathrm{S}$.

In order to quantify the uncertainty in the closed loop by $\hat{\gamma}_\mathrm{S}$, an expression for the corresponding dual Youla parameter is derived as well.

Corollary 4.1. The realization of the uncertain dual Youla parameter $S(\Delta)$ describing the approximate inverse dynamics effect (4.17),(4.20) controlled by (4.35) is given by

$$S : \left[\begin{array}{ccc|c} \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{C}_{31} & \boldsymbol{A}_{\mathrm{S},12} & \boldsymbol{A}_{\mathrm{S},13} & \boldsymbol{B}_{\mathrm{S},11} \\ \boldsymbol{0} & \bar{\boldsymbol{A}} & \bar{\boldsymbol{B}}\boldsymbol{C}_{11} + \bar{\boldsymbol{B}}\boldsymbol{D}_{13}\boldsymbol{F}_{\mathrm{G}} & \bar{\boldsymbol{B}}\boldsymbol{D}_{13} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{F}_{\mathrm{G}} & \boldsymbol{B}_{13} \\ \hline \boldsymbol{C}_{31} & \boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{C}} & \boldsymbol{C}_{\mathrm{S},13} & \boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{D}_{13} \end{array}\right], \tag{4.37}$$

where

$$\begin{aligned} \boldsymbol{A}_{\mathrm{S},12} &= \boldsymbol{B}_{11}\boldsymbol{\Delta}\bar{\boldsymbol{C}} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{C}}, \\ \boldsymbol{A}_{\mathrm{S},13} &= \boldsymbol{B}_{11}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{C}_{11} + \boldsymbol{B}_{11}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{D}_{13}\boldsymbol{F}_{\mathrm{G}} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{C}_{11} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{D}_{13}\boldsymbol{F}_{\mathrm{G}}, \\ \boldsymbol{B}_{\mathrm{S},11} &= \boldsymbol{B}_{11}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{D}_{13} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{D}_{13}, \\ \boldsymbol{C}_{\mathrm{S},13} &= \boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{C}_{11} + \boldsymbol{D}_{31}\boldsymbol{\Delta}\bar{\boldsymbol{D}}\boldsymbol{D}_{13}\boldsymbol{F}_{\mathrm{G}}, \\ \bar{\boldsymbol{A}} &= \boldsymbol{A}_{11} + \boldsymbol{B}_{11}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{C}_{11} + \boldsymbol{B}_{11}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{D}_0\boldsymbol{C}_{31} + \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{C}_{31} + \tilde{\boldsymbol{B}}\boldsymbol{C}_{11} + \tilde{\boldsymbol{B}}\boldsymbol{D}_{13}\boldsymbol{D}_0\boldsymbol{C}_{31}, \\ \bar{\boldsymbol{B}} &= -\boldsymbol{B}_{11}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1} - \tilde{\boldsymbol{B}}, \qquad \bar{\boldsymbol{C}} = -\tilde{\boldsymbol{D}}^{-1}\boldsymbol{C}_{11} - \tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{D}_0\boldsymbol{C}_{31}, \qquad \bar{\boldsymbol{D}} = \tilde{\boldsymbol{D}}^{-1}, \\ \tilde{\boldsymbol{B}} &= \boldsymbol{B}_{13}\boldsymbol{D}_0\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}, \qquad \tilde{\boldsymbol{D}} = \boldsymbol{I} - \boldsymbol{D}_{11}\boldsymbol{\Delta} - \boldsymbol{D}_{13}\boldsymbol{D}_0\boldsymbol{D}_{31}\boldsymbol{\Delta}. \end{aligned}$$

□

Proof: The derivation is analogous to the full case of Theorem 4.2, replacing the coprime factors with those for a static controller from (2.16). Alternatively, the given realization of S can be obtained from (4.31) by appropriately removing obsolete rows and columns for undefined $\boldsymbol{A}_{\mathrm{K}}$, $\boldsymbol{B}_{\mathrm{K}}$ and $\boldsymbol{C}_{\mathrm{K}}$, and taking $\boldsymbol{D}_{\mathrm{K}} = \boldsymbol{D}_0$. ■

4.8 Summary of the Design Steps

The steps to employ the proposed robust robot manipulator control framework in practice are summarized as follows.

① Design a static nominal controller K_0 with the goal of ensuring robust stability to the uncertain system, given an underlying approximate inverse dynamics controller.

② Determine, *e. g.*, by simulation or experiment, an estimate of the norm bounds of Ass. 4.1 and 4.2 for the robot manipulator at hand and the bounds (4.10) and (4.11) that characterize the accuracy of the inverse dynamics controller.

③ Calculate by (B.3) the bounds α_{M} and α_{Ψ}, corresponding to inertia respectively nonlinear signal uncertainty.

④ Obtain by Alg. 4.1 an estimate $\hat{\gamma}_{\mathrm{S}}$ of the worst-case dual Youla operator norm.

⑤ Add the dynamic augmentation to build the central system (4.35) shown in Fig. 4.6. Then, improve the performance by adding Q, designed by any suitable method, subject to $\|Q\|_\infty < 1/\hat{\gamma}_{\mathrm{S}}$.

Steps ① and ② are standard, whereas steps ③–⑤ exploit the novel parameterization based on the double-Youla approach.

4.9 Illustrative Numeric Study

First, let us discuss the implications of the main result by the familiar planar elbow manipulator example. Next, the practical utilization of the proposed framework is illustrated by means of a robotic manipulator with 6 rotational degrees of freedom under varying payload. We will finally compare the novel robust stability framework to the existing ones and outline how a variety of control design methods can be used in the control approach put forward in this article.

4.9.1 Discussion of Worst-Case Dual Youla Uncertainty Measure

The dual Youla characterization according to Prop. 4.1 of our main result provides a new perspective to uncertainty quantification in AID manipulator control.

Example setup. As in the reference book [132], we use a planar elbow manipulator with two rotational degrees of freedom (DoF) to illustrate the control scheme and discuss our results. The dynamical model is given in Appendix C.2. A preliminary analysis yields the following numeric values of the bounds (4.9) for this manipulator:

$$M_\mathrm{u} = 3.765, \quad F_\mathrm{u} = 3, \quad C_\mathrm{u} = 1.289, \quad g_\mathrm{u} = 13.44. \tag{4.38}$$

In order to analyze the interplay of the available model knowledge and the outer-loop controller w. r. t. the uncertainty $S(\Delta)$, the most commonly used controllers of type (4.3) are summarized in Tab. 4.1; the abbreviations PFL, AID, SID, DS, GC, and NID correspondingly refer to the inner-loop controllers in the following. The parameters used in the simulation study and the achievable uncertainty bounds are given in more detail in Tab. C.1.

Remark 4.8 *(Two-degree-of-freedom control).* We employ a two-degree-of-freedom control scheme in the sequel, *i. e.*, the feedforward and feedback gains can be tuned independently. By selecting $\boldsymbol{y} = \mathrm{col}(\hat{\boldsymbol{q}}, \dot{\hat{\boldsymbol{q}}}, \boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d})$ instead of $\boldsymbol{y} = \mathrm{col}(\boldsymbol{e}, \dot{\boldsymbol{e}}, \ddot{\boldsymbol{q}}_\mathrm{d})$, the following developments hold for the general class of two-degree-of-freedom controllers. We use the state-space realization

Table 4.1: Common cases of control laws employed in the inner approximate inverse dynamics loop (4.3), with the values of the uncertainty bounds according to Thm. 4.1. This table is an overview excerpt from Tab. C.1; the dynamic models are provided in Appendix C.2.

Inner Controller		**Bounds**	
Type	**Abbreviation**	α_M	α_Ψ
Perfect feedback linearization	PFL	0	0
Approximate inverse dynamics	AID	0.2523	9.092
Simplified inverse dynamics	SID	0.6457	59.37
Diagonal scaling	DS	0.6457	77.99
Gravity compensation	GC	2.765	59.37
No inverse dynamics	NID	2.765	77.99

as given by (4.22)–(4.26) in Sec. 4.4.2, corresponding to a tracking control configuration of Fig. 4.4. $\triangleleft$

In order to keep the following discussion simple, we restrict most of our attention to the specialization of the main result from Sec. 4.7. The central controller is based on static $\boldsymbol{u}_0 = K_0\boldsymbol{y}$ with

$$K_0 : \boldsymbol{D}_0 = \left[-\boldsymbol{K}_\mathrm{p}, -\boldsymbol{K}_\mathrm{d}, \boldsymbol{K}_\mathrm{p}, \boldsymbol{K}_\mathrm{d}, \boldsymbol{K}_\mathrm{ff}\right], \tag{4.39}$$

and the Q-parameterization is generated from (4.35). Consequently, S is calculated according to Corollary 4.1 unless stated otherwise. The stabilizing gains $\boldsymbol{F}_\mathrm{G}, \boldsymbol{F}_\mathrm{K}$ are design parameters in the construction of the stabilizing factorization (A.21) of the plant and controllers and affect γ_S. In this section, no $\boldsymbol{F}_\mathrm{K}$ is needed because of the static nominal controller, and unless stated differently, $\boldsymbol{F}_\mathrm{G}$ is calculated by (4.36). Note that, by construction, S is a dynamical operator mapping $\boldsymbol{s} \in \mathbb{R}^{n_\mathrm{u}}$ to $\boldsymbol{r} \in \mathbb{R}^{n_\mathrm{y}}$; hence, in this example with two joints and $n_\mathrm{y} = 10$ measurements according to Remark 4.8, S is a 10×2 uncertain system. With γ_S from (4.32) describing the worst-case uncertainty in the closed-loop system given a specific outer loop controller, the following qualitative properties should be captured:

- Larger model uncertainty (*i. e.*, larger values of α_M, α_Ψ) should result in larger γ_S.
- Robust and high-gain nominal controllers should lead to lower γ_S.
- Feedforward control action should *not* increase uncertainty γ_S of the feedback loop.

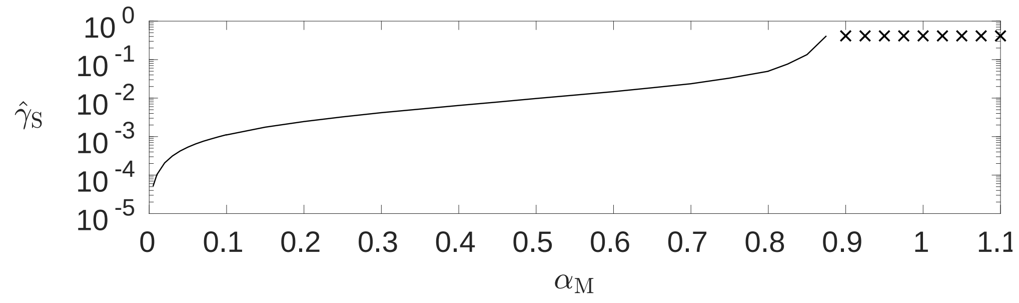
By inspection of (4.31), all these aspects influence the expression for $S(\Delta)$, therefore each will be discussed separately. In order to obtain $\hat{\gamma}_\mathrm{S}$, the procedure from Tab. 4.1 is used with the confidence level set to $p = 99.99\%$ and $\delta = 10^{-4}$, corresponding to $n_\mathrm{S} \geq 92099$ samples over the uncertainty set $\mathcal{D}_\Delta$ for each calculation of $\hat{\gamma}_\mathrm{S}$. Subsequently, in order to uniformly sample over norm-bounded real matrices, the toolbox [240] is used, implementing suitable methods described in detail in [236]. A single evaluation of $\hat{\gamma}_\mathrm{S}$ with 10^5 samples takes approximately 20 min on a current desktop computer (3.9 GHz, 32 GB RAM) using MATLAB R2017a.

Remark 4.9 *(Signal uncertainty for revolute manipulators).* The example manipulator consists only of revolute joints, thus a constant bound on the gravity error can be assumed. In other words, it is known *exactly* that $\|\tilde{\boldsymbol{n}}\|$ does not depend on joint positions and consequently, the uncertainty on nonlinearities can be taken as a matrix $\boldsymbol{\Delta}_\Psi \in \mathbb{R}^{n\times(n+1)}$. $\triangleleft$

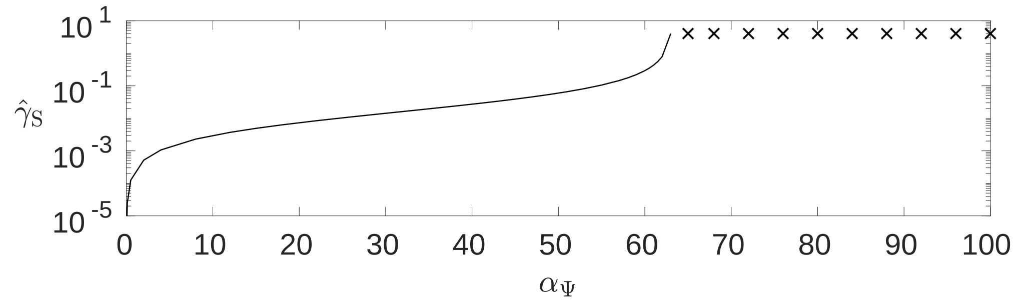
Influence of AID parameters. We first investigate how the accuracy of the approximate model influences the worst-case gain γ_S. To this end, for now assume that the outer-loop controller is a PD feedback controller for critical damping in the nominal loop, *i. e.*, it is defined by (4.39) with

$$\boldsymbol{K}_\mathrm{p} = \mathrm{diag}(K_\mathrm{p}, K_\mathrm{p}), \boldsymbol{K}_\mathrm{d} = \mathrm{diag}(K_\mathrm{d}, K_\mathrm{d}),\ \text{where } K_\mathrm{p} = 10^3, K_\mathrm{d} = 63.3, \quad \text{and } \boldsymbol{K}_\mathrm{ff} = \boldsymbol{0}. \tag{4.40}$$

First, consider the PFL case of Tab. 4.1, *i. e.*, $\alpha_\mathrm{M} = \alpha_\Psi = 0$. Clearly, in this case there is no uncertainty as the underlying feedback linearization uses a perfect model of the manipulator dynamics. The uncertainty set consequently degenerates to $\mathcal{D}_\Delta = \{\boldsymbol{0}\}$ and from (4.31) immediately $S = 0$ follows. As expected from Remark 4.5, the nominal decoupled double integrators describe the resulting loop perfectly.



(a) Evaluation of $\hat{\gamma}_\mathsf{S}$ using a fine discretization over the range of $0 < \alpha_\mathsf{M} \leq 1.1$ and $\alpha_\Psi = 0$. A logarithmic scaling is used on the y-axis in order to visualize the change of $\hat{\gamma}_\mathsf{S}$ in the order of magnitudes for $\alpha_\mathsf{M} < 0.85$, up to instability of S occuring rapidly for $\alpha_\mathsf{M} > 0.85$.



(b) Evaluation of the influence of α_Ψ with $\alpha_\mathsf{M} = 0$. A logarithmic scaling is used on the y-axis for enhanced visibility.

Figure 4.7: Influence of accuracy of inertia and nonlinearities' bounds on $\hat{\gamma}_\mathsf{S}$. Unstable S are marked by x in the graphs only for visibility; by definition, $\gamma_\mathsf{S} = \infty$ if S is not stable.

In the robust control approaches reviewed in Sec. 4.2, the accuracy of the inertia model plays a crucial role as measured by the value of α from (4.10): both [218] and [13] need $\alpha < 1$ as a prerequisite. We therefore begin by investigating the influence of α_M on the worst-case gain γ_S while $\alpha_\Psi = 0$. The calculations were carried out over a fine grid on $0 < \alpha_\mathrm{M} \leq 1.1$. The results are shown in Fig. 4.7a. The uncertainty characterized by the worst-case $\|S(\Delta)\|_\infty$ is very low for $0 < \alpha_\mathrm{M} < 0.7$. We can then observe a rapid increase for $0.7 < \alpha_\mathrm{M} < 0.85$ and $\hat{\gamma}_\mathrm{S} = \infty$, *i. e.*, unstable S, for values greater than $\alpha_\mathrm{M} \approx 0.85$. In other words, robust stability of the inner AID loop using the PD controller with the gains (4.40) is lost for inertia uncertainty greater in norm than 0.85, given that $\tilde{\boldsymbol{n}} = 0$.

Similarly, the influence of the norm of the neglected nonlinearities $\tilde{\boldsymbol{n}}$ can be evaluated. To this end, we have calculated $\hat{\gamma}_\mathrm{S}$ over a grid of $0 < \alpha_\Psi \leq 110$ with $\alpha_M = 0$. As depicted in Fig. 4.7b, for low values of $0 \leq \alpha_\Psi < 50$, $\hat{\gamma}_\mathrm{S}$ stays small with a sudden increase up to instability of S for $\alpha_\Psi > 63$. Comparing to Tab. 4.1, we find that for the gains (4.40), robust stability cannot be concluded for the inner-loop controller types NID, GC, DS and SID, while viable robustness is obtained under the AID law.

We also performed a parameter sweep calculation over a grid of both α_M and α_Ψ as shown in Fig. 4.8. It can be observed that there is a relatively clear boundary between model accuracy that allows for robust controller enhancement, and too inaccurate modeling which results in

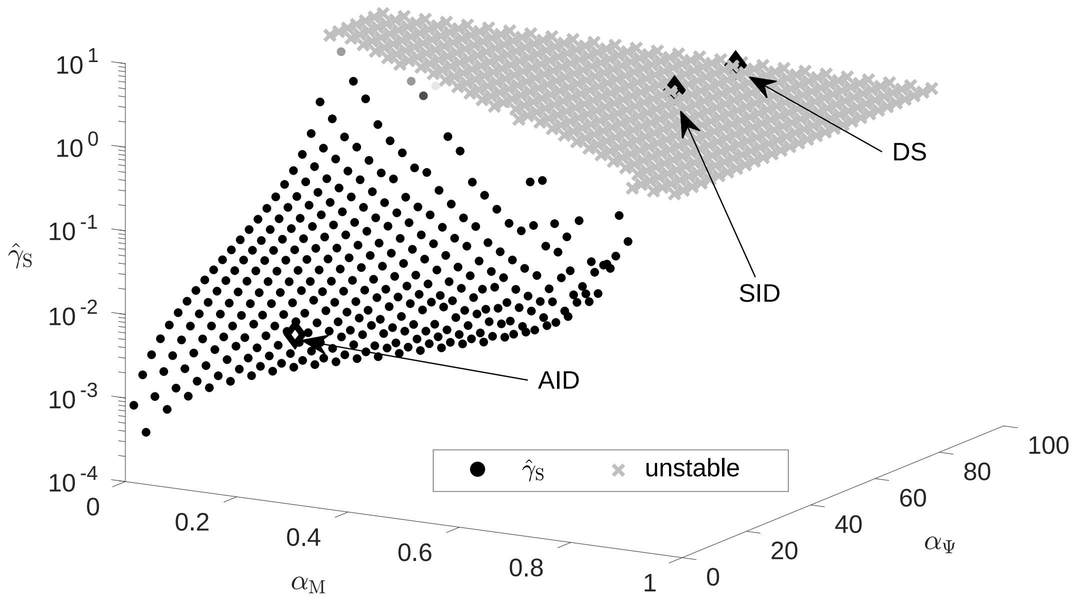
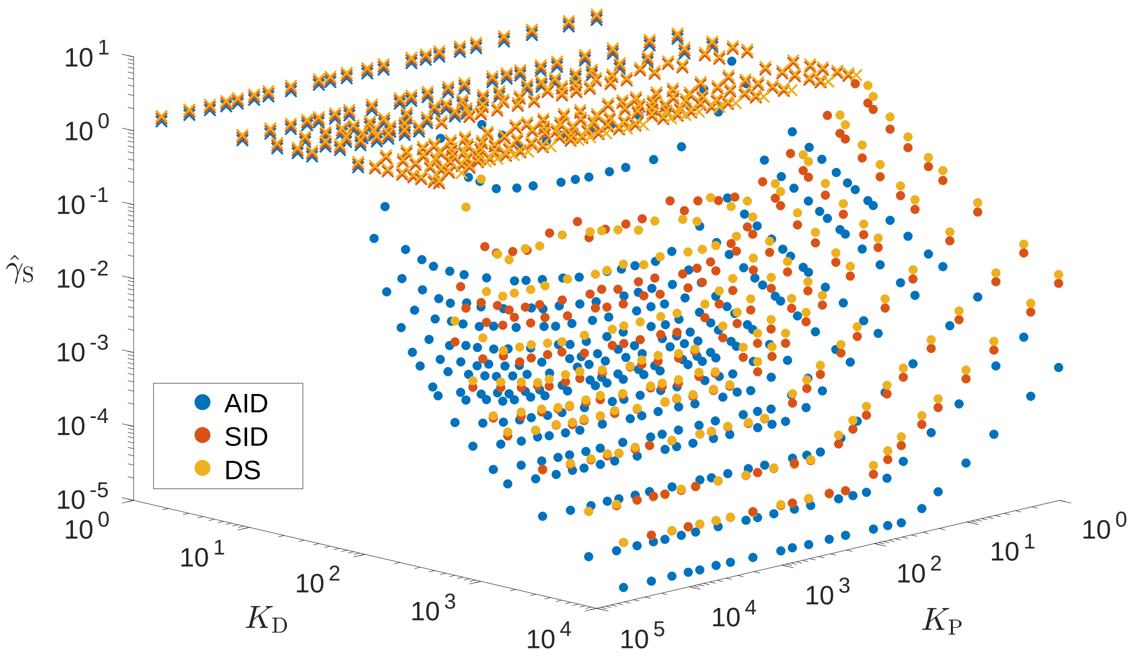


Figure 4.8: Evaluation of worst-case $\|S\|_{\infty}$ over a grid of both α_{Ψ} and α_{M}, where unstable S are marked by x. There is a relatively clear boundary between overall acceptable worst-case model uncertainty and a non-robust nominal control loop with the gains (4.40) in the outer-loop. The depicted inner-loop controllers AID, SID, and DS are according to Tab. C.1. Here, only the AID allows for robustly stable controller enhancement by some parameter Q.

$\hat{\gamma}_{\mathrm{S}} = \infty$. Note that these values are depending on the controller gains and were obtained using the controller (4.40). Figs. 4.7 – 4.8 demonstrate how the main result allows to quantify the worst-case uncertainty set under a given approximate inverse dynamics situation.

Influence of nominal outer-loop controller K_0— Central feedback gains. A high-gain PD controller is working robustly in practice to control the rigid manipulator [183]. It is also clear from (4.31) that the P- and D-gains of the nominal controller influence the uncertainty expressed by the dual Youla parameter. Hence, we calculated the values of $\hat{\gamma}_{\mathrm{S}}$ over a grid of P/D combinations for the central gain (4.39). As shown in Fig. 4.9, the evaluation was performed over a dense grid with values of $0 < K_{\mathrm{p}} < 5 \cdot 10^{4}$ and $0 < K_{\mathrm{d}} < 10^{4}$. Three different parameter sweeps were performed, with the uncertainty levels α_{M} and α_{Ψ} set corresponding to the inner-loop controllers AID, SID and DS from Tab. 4.1. In accordance to intuition, one can observe that the uncertainty in the closed loop decreases with increasing gains and increasing model accuracy. Further, a minimum D-gain of approximately $K_{\mathrm{d}} > 10$ is needed in order to conclude robust stability although the model employed in the AID controller is relatively good. Note that even if the model had been perfect, some $K_{\mathrm{d}} > 0$ would have been required as lead compensation for the double integrator plant. In accordance to the literature [183], large PD gains are required for robustification in case of limited model knowledge.

Influence of nominal outer-loop controller K_0— Feedforward term. The previous analysis only considered a controller feeding back position and velocity errors. Here, the effect of adding feedforward terms is examined, *i. e.*, $\boldsymbol{K}_{\mathrm{ff}} \neq 0$ in (4.39). In the robust robot controllers



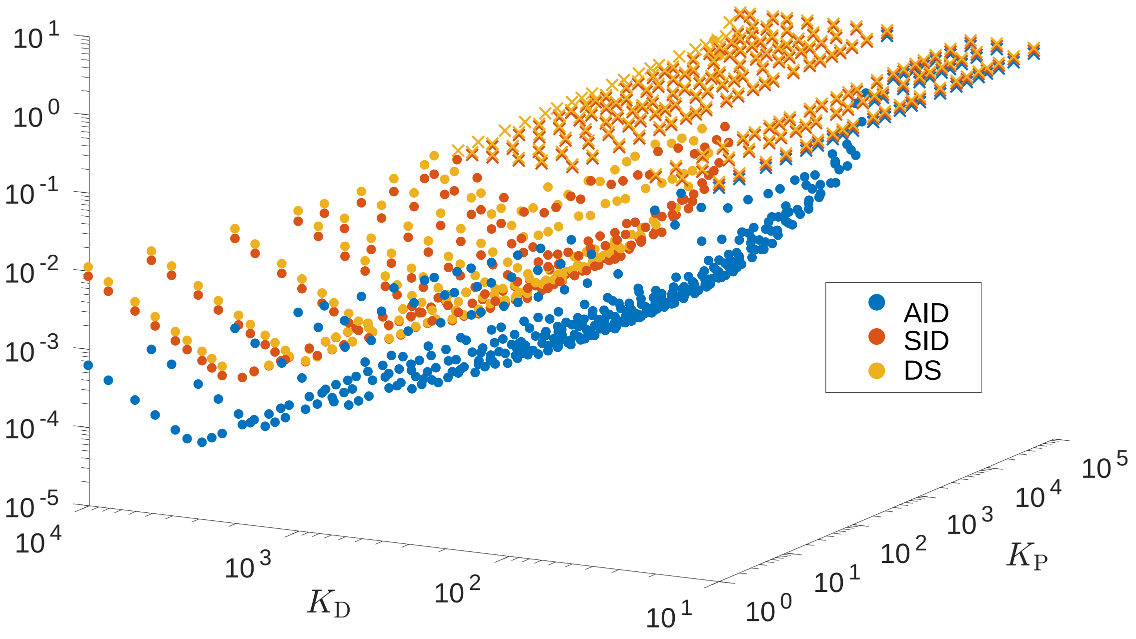


Figure 4.9: Evaluation of worst-case $\|S\|_\infty$ over a grid of P- and D-gains in the outer loop, given the three inner-loop controllers AID, SID, and DS according to Tab. C.1. Configurations where $\hat{\gamma}_{\mathsf{S}}$ is not finite are marked by x for visibility. The GC and NID controllers yield unstable S everywhere on the depicted grid and are hence excluded from the graph. The PFL case is also not depicted as $\hat{\gamma}_{\mathsf{S}} = 0, \forall K_{\mathsf{d}}, K_{\mathsf{p}} > 0$.

of [218, 13, 210], the outer-loop controller (4.12) and (4.15) includes a summand $\ddot{\boldsymbol{q}}_\mathrm{d}$. In the ideal case then, $\boldsymbol{q} = \iint \ddot{\boldsymbol{q}}_\mathrm{d}\,\mathrm{d}t^2 = \boldsymbol{q}_\mathrm{d}$ even without feedback. In the methods reviewed in Sec. 4.2, the robust control term $\boldsymbol{\nu}$ also depends on the maximum norm $\|\ddot{\boldsymbol{q}}_\mathrm{d}\|_{\mathcal{L}_\infty}$ of the desired trajectory acceleration. In the framework put forward in this article, on the contrary, the uncertainty measure S does not change when the feedforward gains are included in (4.39) by some $\boldsymbol{K}_\mathrm{ff} \neq \boldsymbol{0}$. This may seem counter-intuitive, but is a simple consequence of the clear structure obtained by the generalized plant: due to the Q-parameterization, subsequent controller design is based on an internal model of the plant; hence, the structure is carried into all controllers parameterized by Q, *cf.* Remark 4.3. More formally, a two-degree-of-freedom controller can be thought of as a single controller $K = [K_\mathrm{ff}, K_\mathrm{fb}]$ in feedback connection with an augmented plant $\left[0,\ G^\top\right]^\top$ [231, p. 32f]. Hence, all two-degree-of-freedom controllers stabilizing G are obtained [231, p. 49f, p. 53] as all one-degree-of-freedom controllers for $\left[0,\ G^\top\right]^\top$. Consequently, the plant in feedforward control channels is known *exactly*: it is a zero operator. The control $\boldsymbol{u}$ cannot affect feedforward signals. Naturally, the associated uncertainty with this plant is zero, as are the corresponding entries in $S = \begin{bmatrix} S_\mathrm{fb} \\ S_\mathrm{ff} \end{bmatrix} = \begin{bmatrix} S_\mathrm{fb} \\ 0 \end{bmatrix}$. Indeed, in our example system with the measurements according to Remark 4.8, calculating the uncertain operator $S(\Delta)$ confirms that its realization always has the form

$$S : \left[\begin{array}{c|c} \boldsymbol{A}_\mathrm{S}(\Delta) & \boldsymbol{B}_\mathrm{S}(\Delta) \\ \hline \boldsymbol{I}_{4\times 4}\ \boldsymbol{0}_{4\times 6} & \multirow{2}{*}{$\boldsymbol{0}_{10\times 2}$} \\ \boldsymbol{0}_{6\times 10} & \end{array}\right].$$

There is accordingly no uncertainty associated with the last six components of $\boldsymbol{y}$, *i. e.*, $\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d}$. It is a clear advantage of the proposed parameterization that the loop uncertainty measure γ_S can be made independent from quantities such as $\boldsymbol{q}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d}$ by the two-degree-of-freedom design according to Remark 4.8. Note that Fig. 4.9 is obtained irrespectively of $\boldsymbol{K}_\mathrm{ff}$ because $\|S\|_\infty = \|S_\mathrm{fb}\|_\infty$.

4.9.2 Illustrative Example: Double-Youla Parameterization Control of a 6 DoF Robot Model with Varying Payload

With the previous example, only the novel dual Youla perspective on AID uncertainty was discussed. In this section, the utilization of the complete double-Youla parameterization is illustrated by means of a multi-DoF robot system under varying payload. To this end, we consider a PUMA P560 manipulator with 6 DoF, the dynamic model being publicly available from [42]. The task is to track a fast reference trajectory for each joint while the payload m_p applied in 10 cm distance to the end-effector is uncertain within $m_\mathrm{p} \in [0.5, 1.5]$ kg. The trajectory is chosen to cross areas of the state-space where nonlinearities and inertial interactions are strong, precisely we use $\boldsymbol{q}_{\mathrm{d},i}(t) = q_{0,i} + a_i \frac{\pi}{180} \sin(2\pi f_i\, t)$, where $a_1 = 90, a_2 = 45, a_3 = 22.5, a_4 = 55, a_5 = 50, a_6 = 133$ and $f_1 = 0.2, f_2 = 0.4, f_3 = 1, f_4 = 0.5, f_5 = 0.25, f_6 = 0.2$. Let us walk through steps ①–⑤ of our approach as summarized in Sec. 4.7.

① For the nominal control design, an inverse dynamics controller is used in the inner loop based on the robot model for $m_\mathrm{nom} = 1.0$ kg. Given the varying payload, this controller can only achieve an approximate feedback linearization. Therefore, an outer-loop controller K_0 is designed to robustify the loop. We use the proportional gains $K_\mathrm{p} = 900$ and derivative gains $K_\mathrm{d} = 2\sqrt{K_\mathrm{p}} = 60$ in each joint.

② Next, one needs to obtain numeric values for the bounds (4.10), (4.11) and (4.9a). By a simulation as in [191], we find

$$\alpha = 0.3438 \text{ (worst-case with } m_\mathrm{p} = 0.5), \quad \alpha_0 = 5.4028, \quad \alpha_1 = 4.3689, \quad M_\mathrm{u} = 5.7032.$$

③ While the previous bounds are only due to the manipulator and uncertain feedback linearization, we now construct the novel generalized plant setup proposed in Thm. 4.1. Just as in the previous example, we use the two-degree-of-freedom design structure given in Sec. 4.4.2. Now, the uncertainties are conceptually pulled out as shown in Fig. 4.4. To keep the conservatism reasonable, some characteristics of the manipulator are considered. In particular:

- All 6 DoF of the P560 are rotational (*cf.* Remark 4.9).
- The Coriolis and centripetal effects of the P560 are to a very large extent determined from $\dot{q}_1, \dot{q}_2, \dot{q}_3$.
- Friction is independent of payload, hence compensated by the inverse dynamics controller.

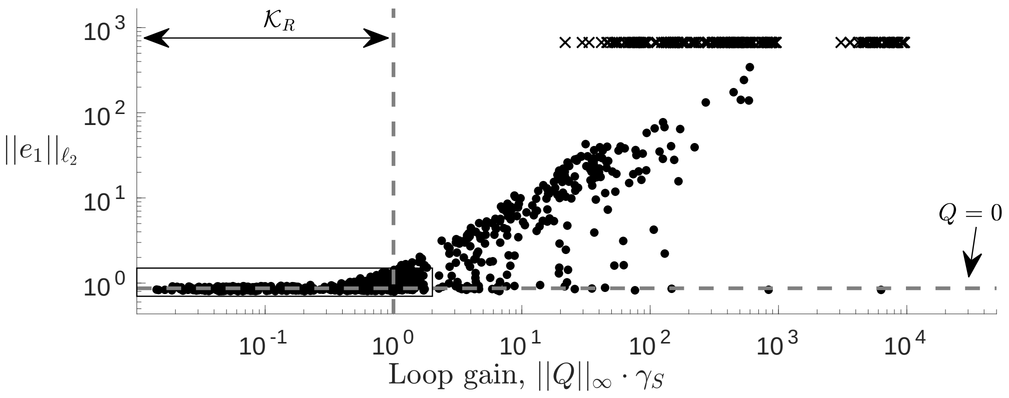
Accordingly, the uncertainty structure (4.20) is described by matrices $\boldsymbol{\Delta}_\mathrm{M} \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{\Delta}_\Psi \in \mathbb{R}^{3\times(3+1)}$. From the values obtained in step ②, we have as of (B.3) the associated norm bounds $\alpha_\mathrm{M} = 0.3438$ and $\alpha_\Psi = 39.627$.

④ Next, the realization of the uncertain dual Youla operator (4.37) is calculated. We then assign the probability levels $p = 99.9\%$ and $\delta = 10^{-4}$ to employ Algorithm 1 from Tab. 4.1, using 10^4 samples. The worst estimate is $\hat{\gamma}_\mathrm{S} = 0.1137$.

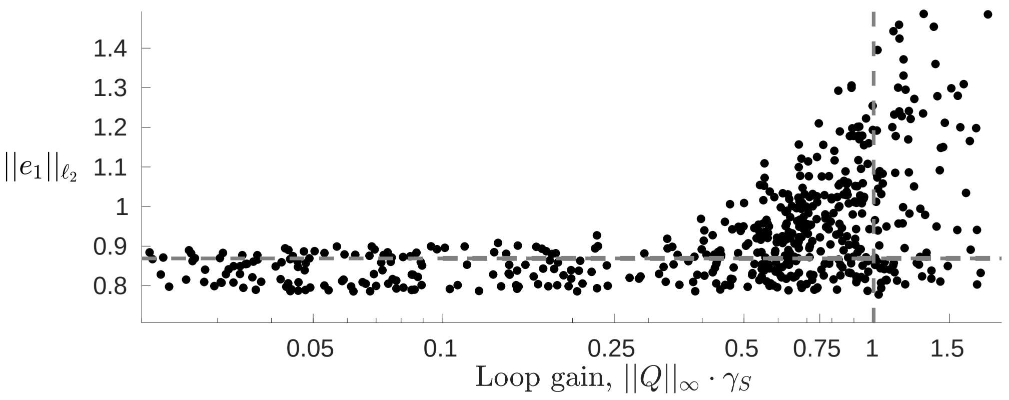
⑤ The detailed uncertainty quantification of the dual Youla parameter S by ①-④ is worthwhile once the Q-parameterization is used to enhance performance. That is, the nominal PD controller is firstly augmented as shown in Fig. 4.6. According to Theorem 4.3, the parameterization now allows to search over robustly stabilizing controllers simply by choosing a stable (possibly time-varying) finite-gain $\mathcal{L}_2$ stable parameter system Q. By the value of $\hat{\gamma}_\mathrm{S} = 0.1137$, robust stability is assured if $\|Q\|_\infty < 1/\hat{\gamma}_\mathrm{S} \approx 8.8$.

As discussed in Chap. 2, there is a multitude of design methods for the parameter Q and the reader is referred to the literature, *e. g.*, [269, 51, 28, 23, 80, 59, 231]. A specific learning approach will be considered in Chap. 8 of this thesis. In general, the less restricted $\|Q\|_\infty$ needs to be, the more design freedom there is for any such method.

Let us briefly illustrate this trade-off. To this end, some controllers Q are "designed" by randomly sampling stable dynamic systems of maximum order 10. We run 20 times 50 nonlinear simulations of the closed loop over $T = 10\,\mathrm{s}$, each time increasing the allowed threshold of the loop norm $\|Q\|_\infty \cdot \hat{\gamma}_\mathrm{S}$ in a logarithmic range of $0.05, 0.1, 0.25, 0.75, \ldots, 500, 10^3, 10^4$. In order to account for the uncertain load, a random value $0.5 \leq m_\mathrm{p} \leq 1.5$ is assigned to the payload in each simulation run. The initial state is $\boldsymbol{q}_0 = [0, \pi/4, -\pi/2, 0, 0, 0, \mathbf{0}_{1\times 6}]^\top$. A discrete-time formulation of the controller is used for implementation. The sampling time is constant with $t_\mathrm{s} = 5\,\mathrm{ms}$ and the solutions are obtained by a 3rd order Runge-Kutta method (Bogacki-Shampine). To keep the comparison irrespective of feedforward control, only a feed-



(a) Joint tracking error norms obtained by 1000 simulation runs with randomly sampled controllers. The gray vertical dashed line denotes the border of the set $\mathcal{K}_\mathrm{R}$, *i. e.*, $K(Q) \in \mathcal{K}_\mathrm{R}$ if $\|Q\|_\infty \hat{\gamma}_\mathrm{S} < 1$. The horizontal dashed line depicts the performance of the central PD controller $K(Q = 0)$. Unstable simulations are marked by x. The area in the rectangle is magnified in Fig. 4.10b.



(b) Performance of 552 randomly sampled controller augmentations, of which 452 fulfill $Q : K(Q) \in \mathcal{K}_\mathrm{R}$ (left side of the vertical dashed line). The best such controller improves the tracking performance by 11% w. r. t. the nominal design ($Q = 0$) indicated by the horizontal dashed line.

Figure 4.10: Tracking errors of a P560 with uncertain payload and 1000 randomly sampled filters Q. The simulation confirms that all controllers $K(Q) \in \mathcal{K}_\mathrm{R}$ robustly stabilize the nonlinear loop. Conservatism is discernible in that some $K(Q) \notin \mathcal{K}_\mathrm{R}$ also yield improved performance.

back controller augmentation is used in this example, *i. e.*, $Q_\mathrm{ff} = \mathbf{0}_{6\times 18}$ in all simulations.

The result of these simulations is summarized in Fig. 4.10, depicting the ℓ_2-norm of the 6-dimensional error signals. Controllers on the left side of the vertical dashed line in Fig. 4.10a are within the set $\mathcal{K}_\mathrm{R}$ from (4.34). It can be observed that none of the controllers in $\mathcal{K}_\mathrm{R}$ leads to a significant increase in the error norm. Leaving the set of admissible robust controllers, the error norms rapidly increase up to practically useless control behavior and instability of the simulated loops starting from approximately $\|Q\|_\infty > 20/\hat{\gamma}_\mathrm{S}$. Compared to $Q = 0$, 46 of the 548 controllers $K \notin \mathcal{K}_\mathrm{R}$ yield better performance; however, 211 controllers $K \notin \mathcal{K}_\mathrm{R}$ lead to instability. This study shows how the parameterization of Theorem 4.3 is indeed useful:

it allows to search exclusively over robustly stabilizing controllers. Robust performance can subsequently be obtained by suitable design of Q subject to $\|Q\|_\infty < 1/\hat{\gamma}_S$. Approximately half of the LTI systems Q actually improve the performance of the overall control system in this study, albeit being only randomly sampled. This is depicted in Fig. 4.10b, which shows a magnification of the area marked by the rectangle in the lower left corner of Fig. 4.10a.

4.9.3 Comparison to Related Work

Single primary vs. double-Youla parameterization. The robust control of Sec. 4.2.1 derived in the classic frequency domain Youla parameterization always yields a high-gain dynamic compensator [218]. Here, the parameterization is in a state-space description and it can be based on both dynamic and static central K_0. Moreover, the conventional approach is lumping all uncertainties in a single term (4.14) of internal feedback disturbances $\boldsymbol{\eta}$ that must be suppressed. In contrast, the structure (4.20) proposed in this chapter more accurately reflects that of AID uncertainty (4.6)–(4.8).

It may be less evident that the purpose of the parameterizations is quite different. In the method of [218], the central controller does not ensure robust stability; robustness is obtained by careful design of Q (“The choice of [Q] is not easy to see” [218]). As the primary Youla parameterization only yields stabilizing controllers for the unperturbed plant, *i. e.*, the nominal double integrator, in [218] the design of the filter system Q must ensure that the internal disturbance $\boldsymbol{\eta}$ is suppressed and not destabilizing. In this chapter, we do not report another robustification tool but rather characterize a whole set of robustly stabilizing AID controllers (4.34) such that the control performance can be enhanced online (*cf.* Fig. 4.1 and Remark 4.6). Such enhancement is possible using time-varying or switching schemes, including adaptive [234, 126], learning [57], model predictive [237], hybrid [80], and gain scheduling [221, 23] approaches. Most of these references only focus on the design aspect of the system Q, *i. e.*, design for a nominal model within the set of stabilizing controllers. By the contributions in this chapter, such a design can be systematically tightened to the subset of robustly stabilizing controllers for robot manipulators. Finally, the traditional approach dictates the design of the system Q and is therefore limited to robust trajectory tracking control. The double-Youla framework reported here, in general, does not make this restriction because all derivations refer to the generalized plant (4.17).

Comparison to Lyapunov-based designs. By the revised robust control design, Bascetta and Rocco [13] essentially undo the step in the Lyapunov-based designs (Sec. 4.2.2) of lumping the closed-loop uncertainty into a single term $\boldsymbol{\eta}$ in (4.16). Their method consists of two steps: first, the nominal PD controller is designed to ensure global asymptotic stability of the error system under perturbation with any admissible matrix $\|\boldsymbol{\Delta}_\mathrm{M}\| \leq \alpha$. Second, the asymptotic performance under the influence of the neglected terms $\tilde{\boldsymbol{n}}$ is recovered by an additional feedback term $\boldsymbol{\nu} = f(\boldsymbol{e})$ obtained via a quadratic Lyapunov function. Our requirement of (4.33) that robust stability is ensured by the nominal controller is therefore similar to the first step of [13]. While the methodological approach is quite different, the additional input $\delta \boldsymbol{u}$ in our method corresponds to the term $\boldsymbol{\nu}$ in [13] as emphasized in Remark 4.7. The parameterization of Thm. 4.3 thus constitutes a viable alternative to the

Lyapunov-based methods. The general drawback of a factorization approach is that only uniform ultimate boundedness is ensured as long as the linear bound (4.11) holds. In turn, the double-Youla parameterization brings some advantages. First, the Lyapunov approaches to robust manipulator control typically result in high frequency additional control signals $\boldsymbol{\nu}$. Here, in contrast, the characteristics of $\delta\boldsymbol{u}$ depend exclusively on the design of Q. For example, the controllers of Sec. 4.9.2 generate smooth signals because Q is LTI. Second, [13] only deals with robust tracking control. The novel double-Youla parameterization, however, is derived for a generalized plant structure (4.17) and an arbitrary LTI controller K_0. It therefore complements the Lyapunov-based approaches in terms of versatility, as (4.17) allows to represent via $\boldsymbol{z}$ many more control objectives in the design step of Q.

Recent approaches. Let us finally recapitulate the distinctive features of the presented double-Youla approach compared to recent literature on robust manipulator control. The conservatism is determined by the accuracy of the model bounds and nominal outer-loop compensator gains. Helwa *et al.* [76] consequently propose a learning-based adjustment of the uncertainty bounds in the Lyapunov-based robust design. Nonetheless, the robustification term is a switching signal of potentially high frequency and the overall design is tailored towards tracking control; one could then also work with a robust-adaptive scheme, *e. g.*, [73]. Also the work of Kim *et al.* [112] is related, in that the L_1 robustness bounds therein are as well derived by consideration of a generalized plant interconnection. However, [112] starts from the error system (4.16) and the disturbance signal is only split into exogenous and internal components. The uncertainty structure used in [112] hence does not distinguish between inertia and signal uncertainty, resulting in conservative design, *cf.* Remark 4.1. In all these works, the analysis is conducted regardless of the nominal outer-loop PD controller that has considerable influence on robustness. The dual Youla measure of uncertainty proposed in this chapter, in turn, provides a general approach to systematically quantify this influence.

4.10 Limitations

In this section, the current limitations of the framework presented in this chapter are discussed.

Non-robust initial controller. In order to obtain $\hat{\gamma}_{\mathrm{S}} < \infty$, by condition (i) of Lemma 4.1 the nominal controller K_0 needs to stabilize the system under all perturbations $\Delta \in \mathcal{D}_\Delta$. As revealed by the study of the example system above, with a static initial controller, the uncertain model cannot be robustly stabilized when low gains are employed or when the dynamic model is too inaccurate. This is in line with known results on robust manipulator control theory. On the other hand, it may well be possible to robustify a control loop in the double-Youla parameterization by the addition of Q, even adaptively [234]. With such a design approach, in turn, the resulting (dynamic) controller $K(Q)$ could be used as the initial controller, just as in [218]. The corresponding worst-case uncertainty assessment is then feasible by the general solution of Thm. 4.2 for a dynamic central controller. In this case, however, there does not seem to be a clear advantage over starting, for example, with an $\mathcal{H}_\infty$-method for the design of the nominal controller.

Precautions in practical implementations. When applied to real systems, further precautionary measures should be taken in order to avoid damage to the robot or the environment. With the control system being robustly internally stable, no signal grows unbounded in the closed-loop system (uniform ultimate boundedness of the tracking error). This should neither be confused with Lyapunov stability nor asymptotic performance. Taking the linear error bound (4.11) further restricts us to semi-global stability, *i. e.*, it is only valid as long as the quadratic error norm is over-bounded. While theoretically not necessary, in practical implementations the feedforward gain of Q_{ff} should also be carefully bounded to ensure that the robot is never driven out of its admissible workspace.

Conservatism of the method. Robust control introduces conservatism by definition. In our method, several assumptions underpinning our approach introduce such conservatism: the over-bounding of the nonlinear parts by means of a norm-bounded matrix, the linear bound employed in the function Φ, and finally the small-gain theorem to establish stability under controller perturbations. In the simulation study above, this conservatism is visible in Fig. 4.10a. The simulation of the actual nonlinear loop may still result in acceptable error norms even for controllers $K(Q) \notin \mathcal{K}_{\mathrm{R}}$.

4.11 Conclusion

In this chapter, a double-Youla parameterization for robust control of rigid body manipulators has been proposed, with the dual parameterization being the key tool to quantify the uncertainty of the control loop. The overall contribution is the construction of a subset of robustly stabilizing controllers for approximate inverse dynamics manipulator control problems. Using a static nominal controller, the theory specializes to a handy structure suitable for practical implementations. The proposed methodology constitutes a control strategy that allows to apply numerous advanced design methods for enhancement of inverse dynamics based feedback controllers in a strict robust stability framework. While the design of the system $Q \in \mathcal{Q}$ for performance improvement is not within the scope of this chapter, learning will be explored in Chap. 6 as one such way to enhance the performance. By the proposed framework, however, the admissible set $\mathcal{Q}$ can be systematically tightened to contain only robustly stabilizing controllers for robot manipulators.

The main limitation of the proposed approach is the conservatism introduced by over-approximating the nonlinear error dynamics using an uncertain matrix w. r. t. the linear nominal double integrator model. In principle, one could adapt certain measures to reduce conservativeness during the later controller design stage. For example, it is common to use some pre- and post- filtration of Q such that the worst case loop gain is approximately constant over frequency, aiming to construct a less conservative controller, as for example in [237]. One might even aim to build less restrictive controllers by allowing for temporary (unstable) finite growth as long as it is guaranteed that the loop will again be a strong enough contraction at a later time instant [243]. However, such ideas are clearly not in the spirit of guaranteed robust stability put forward in this thesis, aiming for applicability to real-world robots.

As emphasized earlier, the parameterization generally entails controllers based on the internal model principle. Clearly, also the disturbance observer compensation designs are a specific implementation of the internal model principle [111]. Just as the linear robust internal-loop compensator structure from [111] has been generalized to the nonlinear setting [114], a nonlinear flavor of our method might be developed in future work: with suitable extensions [4], one could take advantage of the available manipulator model directly in terms of a parameterization. It may also be feasible to develop a dual Youla uncertainty characterization for the robust manipulator control with a disturbance observer based inner loop compensation [200]. Another open research direction is to include the analysis of flexible robots [140] in the framework.

Part II

Model-Free Learning Control from the Robotics and Controller Parameterization Viewpoints

5

Automated Continuous Online Least-Squares Policy Iteration for Robot Control

In the previous part of this thesis, the parameterization of stabilizing controllers was examined as a tool to establish robust adaptive algorithms for robotics. By design, these parameterizations are model-based. However, for many robotic tasks, detailed mathematical modeling is hard or time-consuming, which makes reinforcement learning (RL) an attractive alternative to model-based control design. In this second part, model-free RL algorithms are examined from the perspective of robotics (Chap. 5) and w. r. t. the interplay with the previously shown parameterizations (Chap. 6). Generally in parts II and III of the thesis, the methods are developed in the discrete-time setting throughout.

In this chapter, novel RL algorithms are reported that belong to the class of least-squares policy iteration (LSPI) algorithms. The development of these algorithms was initially motivated by taking the designated deployment in the framework of stabilizing parameterizations into account. Moreover, the specific needs arising from the robotics problem domain background are considered. These new results therefore contribute towards the important goal of powerful online learning robot control. Parts of this chapter have been published in [61].

The remainder of this chapter is organized as follows. First we recall the basics of RL in Sec. 5.1 before outlining in detail the need for an online, continuous and automatic LSPI algorithm in Sec. 5.2. The general related work is reviewed in Sec. 5.3, leading to a summary of the contributions of this chapter in Sec. 5.4, before the main ideas of specifically LSPI and its kernel variant LSPI are recalled in Sec. 5.5. On the way to the main result, in Sec. 5.6 a novel kernel-based online LSPI algorithm is proposed. The main contribution is then given in Sec. 5.7, an online LSPI algorithm with automatic tuning capability that is applicable to continuous action space domains. Our simulation studies reported in Sec. 5.8 evaluate the new methods, highlighting their advantages over the baseline algorithms and discussing their performance for a wide range of algorithmic parameters. The chapter is concluded in Sec. 5.9.

5.1 Introduction to Reinforcement Learning

In this section, the basic concepts of RL are introduced as required for the developments later in this thesis. Interacting with the environment in trial-and-error fashion is the core idea of RL methods [228], allowing to infer desired behavior. While RL constitutes a general framework to learn sophisticated behaviors in a multitude of disciplines, robotic tasks are often closely related to optimal or adaptive control problems. In this context, some RL methods can be conceived of as direct adaptive optimal control [227]. Some contributions in the field of adaptive dynamic programming are also relevant, particularly if it is important to keep a continuous-time formulation, see for example [249] and the references therein. Given the large body of literature, the interested reader is referred to the standard reference [228] for the perspective of AI, to [249] for a more control-oriented point of view, and to [34] for an elaborate discussion on RL in continuous state- and action domains as typical for many control systems. For robot control, iterative discrete-time RL algorithms are more frequently used, see [115] for a comprehensive overview. Selected examples include tracking performance improvement of robot manipulators [174, 57], aerial transportation with drones [172], control of autonomous vehicles [250, 245], marine vehicle navigation [242], pendulum systems [259, 45], human-robot cooperative manipulation [171], apprenticeship learning [154] and information exchange in cooperative multi-agent systems [238].

Overview. In this thesis, we adopt an artificial intelligence convention to RL because it is the approach predominantly taken in robotics. This section is to only provide a very brief overview of the central concepts, starting from the theory of multi-stage decision processes over a MDP [21, 182]. We adopt the following definition.

Definition 5.1 *(Markov Decision Process [34, Ch. 2.2]).* A Markov decision process (MDP) is defined by the 5-tuple $(\mathscr{S}, \mathscr{A}, P_s^a, \mathsf{r}, \gamma)$ with the state space $\mathscr{S}$, the action space $\mathscr{A}$, the transition function $P_s^a : \mathscr{S} \times \mathscr{A} \mapsto \mathscr{S}$, the reward function $\mathsf{r} : \mathscr{S} \times \mathscr{A} \mapsto \mathbb{R}$ returning a cost index R_t that describes the immediate benefit of applying action $\mathsf{a}_t \in \mathscr{A}$ in state $\mathbf{s}_t \in \mathscr{S}$, and the scalar discount factor $0 \leq \gamma \leq 1$. ◇

The goal of solving an MDP refers to choosing actions such that the future accumulated discounted reward is optimized:

$$\text{maximize } \mathsf{G}_t = \mathsf{R}_{t+1} + \gamma \mathsf{R}_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i \mathsf{R}_{t+i+1}. \tag{5.1}$$

If the dynamics of an MDP are known, *i. e.*, the transition functions P_s^a and the reward functions are known, the optimal policy can be found via planning algorithms, most prominently dynamic programming (DP) [15, 182, 21]. The goal of maximizing the return for every possible state leads to the central idea of *value- (or critic-)based* methods: constructing a ranking of all possible states $\mathbf{s} \in \mathscr{S}$ of the MDP with the purpose of finding the optimal action $\mathsf{a}_t^\star$ in each step t that is expected to lead to the highest ranked successor state $\mathbf{s}'$. To this end, the *state value function* $\mathcal{V}_\pi : \mathscr{S} \mapsto \mathbb{R}$,

$$\mathcal{V}_\pi(\mathbf{s}_t) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathsf{R}_{t+1} \,\middle|\, \mathbf{s}_t \right], \tag{5.2}$$

describes the expected return when starting in state $\mathbf{s}_t$ and navigating through the MDP according to a *policy* $\pi : \mathscr{S} \mapsto \mathscr{A}$. It is important to note that such a representation can only be created with respect to a policy that determines the state transitions; hence, the subscript π. Solving an MDP refers to finding an optimal policy $\pi^\star$ that maximizes the expected return in all states, $\pi^\star = \arg\max_\pi \mathcal{V}_\pi(\mathbf{s})$, $\forall \mathbf{s} \in \mathcal{S}$; such $\pi^\star$ always exists [182].

Usage of the state value function $\mathcal{V}_\pi(\mathbf{s})$, however, requires knowledge about the transition probabilities of the MDP to evaluate possible successor states. Contrary to that, Reinforcement learning operates on a trial-and-error basis and does not rely on information about the MDP dynamics. The reward function r is usually designed to represent the control/decision objective. However, depending on the context, this function may also be unknown in RL. A typical example for this case is when a human teacher provides in each step with R_t a subjective ranking of performance to a robotic system.

If action a_t is chosen, let $P_s^a(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t, \mathsf{a}_t)$ be the (unknown) probability that the system transitions to state $\mathbf{s}_{t+1}$, *i. e.*, the transition function corresponds to the system dynamics. The value function can be extended to account as well for the value of taking a specific action a_t, defining the *state-action value function* $\mathcal{Q}_\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ as

$$\mathcal{Q}_\pi(\mathbf{s}_t, \mathsf{a}_t) = \mathbb{E}_\pi[\mathsf{G}_t \,|\, \mathbf{s}_t, \mathsf{a}_t]. \tag{5.3}$$

The $\mathcal{Q}$-function assigns each state-action pair the expected sum of rewards when starting from state $\mathbf{s}_t$, taking action a_t, and henceforth following the policy π.

As the policy π determines both $\mathcal{V}_\pi$ and $\mathcal{Q}_\pi$, the *optimal* value functions $\mathcal{V}^\star(\mathbf{s})$ and $\mathcal{Q}^\star(\mathbf{s}, \mathsf{a})$ are

$$\mathcal{V}^\star(\mathbf{s}) = \max_\pi \mathcal{V}_\pi(\mathbf{s}), \qquad \mathcal{Q}^\star(\mathbf{s}, \mathsf{a}) = \max_\pi \mathcal{Q}_\pi(\mathbf{s}, \mathsf{a}). \tag{5.4}$$

The state-action space is henceforth denoted $\mathcal{SA} \triangleq \mathcal{S} \times \mathcal{A}$, and a state-action value function $Q : \mathcal{SA} \mapsto \mathbb{R}$ entails a (greedy) policy via the *policy improvement*

$$\pi(\mathbf{s}) = \arg\max_{\mathsf{a} \in \mathcal{A}} \mathcal{Q}(\mathbf{s}, \mathsf{a}). \tag{5.5}$$

5.2 Motivation

Two main classes of RL algorithms are the value-based approaches and the value function free methods, *e. g.*, policy search. On the one hand, policy based RL is predominant in robotic applications due to several factors [46]: a policy search algorithm works with an explicitly pre-structured parametric policy and iteratively improves the policy by locally optimizing directly in the space of parameters. Therefore, suitable policy representations allow to reduce the learning problem from the potentially high-dimensional state-action space to a lower-dimensional optimization problem in parameter space, greatly simplifying the learning problem in practice [225]. Moreover, the demand for continuous and possibly multidimensional action spaces is more naturally covered in policy based algorithms. On the other hand, a value function based method constructs a ranking over the state and action sets w. r. t. the expected long-term reward, thereby implicitly encoding a globally optimal policy. This approach, however, entails properties that become particularly problematic for robot control [46]. Function approximators [65] must be employed to represent the value of a given state/action combination in the oftentimes large state-action space of robotic systems. Accordingly, the computational complexity easily becomes intractable due to the curse of dimensionality. A particularly recurring research question is therefore how the action space in continuous domains can be smoothly approximated, *e. g.*, by discretization and subsequent symbolic post-processing [3] or heuristically by expert knowledge and fuzzy representations [86].

Despite their drawbacks, value function based algorithms are preferred in some robotic applications in order to avoid the limitations of policy search, see [115, Tab. 1]. In particular, one needs to construct suitable policy parameterizations and find good initial policy parameters for local optimization in policy search. A class of popular value function algorithms is based on least-squares policy iteration (LSPI) [128]. Extensions to approximation-based LSPI are studied in detail in [34], and an online least-squares policy iteration (OLSPI) algorithm is derived in [33]. These algorithms iteratively evaluate and improve the control policy, are sample-efficient, and have comparatively good convergence properties due to the least-squares techniques for policy evaluation. For example, [172, 245, 171, 238, 250, 242] all employ some form of LSPI.

It is currently, however, rather tedious to apply LSPI algorithms to practical robotic problems. First of all, there often is a demand not only for a continuous state but also a continuous action space representation. Therefore, it is necessary to employ a value function approximation (VFA) method and the achievable performance depends considerably on an appropriate representation for the system at hand. Next, operating online means that data cannot be collected in advance but has to be obtained incrementally, requiring fast enough processing cycle times and manageable memory complexity. Finally, it is crucial to employ well-tuned algorithmic parameters in order to obtain a performant learning system. For example, Anderlini *et al.* [5] report unexpected behavior of LSPI in the control of a wave energy converter model, presumably due the radial basis function approximation. In robotics, this issue can become even more tedious, particularly when tuning the algorithmic parameters is costly in experimental setups where merely collecting suitable data can be hard, *e. g.*, in closed-loop feedback systems.

Summary. To leverage the potential of LSPI in robotics, algorithms are needed that operate *online*, over *continuous state and action spaces*, and *automatically* handle the VFA. Such an algorithm is proposed in this chapter.

5.3 Related Work

Given the wealth of literature on general RL, attention is given specifically to LSPI class algorithms employing function approximators to represent the value function. A more extensive treatment of approximation-based RL can be found in [34]. If for example deterministic dynamics can be exploited, fuzzy techniques [34, Ch. 4] offer a viable alternative to encode prior expert domain knowledge in the value function. An introduction to RL with linear function approximators in particular is provided in [65]. In general, however, feature or basis function (BF) selection and correspondingly "a memory management scheme for LSPI's data [...] is non-trivial" [65, Ch. 4.5, p. 437]. Adaptively growing kernel representations [204] offer a promising way to deal with this problem: the very same issue of BF selection with memory management arises in kernel adaptive filtering [139], and a multitude of sparsification schemes have recently been developed in the signal processing community. The general VFA problem is pervasive in high-dimensional RL; hence, we focus on reviewing kernel-based RL methods. For a broader perspective, the interested reader is referred to the discussion in [228, Ch. 8], [34, Ch. 3.6], [65, Ch. 3], and the references therein.

Kernel methods [204] have in common that a sparsified set of features is used to represent a high-dimensional, implicit feature space only by means of the raw data transformed by the kernel. With the versatility of Gaussian processes [186], kernel methods are also becoming successful more and more in the fields of RL and in robotics. Several methods exploit such a representation to model the dynamics, *e. g.*, [45, 179, 247]. These approaches are not reviewed in more detail as they pursue an indirect, *i. e.*, model-based, approach. Several value-based model-free RL methods with non-parametric value function modeling have been developed, as reviewed next. The paper [169] by Ormoneit and Sen is an early contribution showing that the distribution of the estimate may be conceived of as a Gaussian process. Jung and Polani [100] further develop kernel least-squares policy evaluation (KLSPE), a kernelized online policy evaluation scheme and demonstrate their results on a high-dimensional benchmark system; however, a discrete set of pre-defined actions is used. Xu *et al.* [260] later develop kernel-based least-squares policy iteration (KLSPI), a flavor of LSPI where data is selected according to an approximate linear dependency (ALD) criterion and the value function is represented by means of a kernel expansion. Closely related papers are [97] and [262], which employ direct recursive versions of KLSTD respectively KLSPI. These algorithms, however, are not optimized for online usage and are only applicable to discrete state sets. Recently, Cui *et al.* [43] demonstrate that so-called kernel dynamic policy programming (KDPP) is applicable to high-dimensional robotic systems and the authors also compare to the KLSPI

Table 5.1: Overview of model-free value-based RL algorithms with kernel VFA capability, with LSPI and OLSPI included for comparison. The symbols ✓, (✓), and ✗ correspond to "yes", "partially", and "no".

Approach		Online	Continuous		Feature selection		Admissible system dimensionality	Kernel trick	Initial policy required
Reference	Algorithm		State	Action	Automatic	Sparsification			
[128]	LSPI	✗	✓	✗	✗	–	low	✗	✗
[33]	OLSPI	✓	✓	✗	✗	–	low	✗	✗
[34]	OLSPI with cont. action	✓	✓	✓	✗	–	low	✗	✗
[260]	KLSPI	✗	✓	✗	✓	ALD	mid	✓	✓
[262]	Online KBLSPI	(✓)	(✓)	✗	✓	ALD	mid	✓	✗
[97]	KRLSTD	✗	(✓)	✗	✓	Laplacian	low	✓	✓
[43]	KDPP	✗	✓	✗	✓	ALD	high	✓	✗
[61]	OKLSPI	✓	✓	✗	✓	Coherence	mid	✓	✗
[61]	AOLSPI	✓	✓	✓	✓	Coherence	mid	(✓)	✗

algorithm; nonetheless, [43] uses ALD for the dictionary sparsification step as well and also KDPP is only applicable with a discrete action set. These approaches have the common advantage that the features are generated in data-driven fashion but the VFA is still in linear form. A comparison of these value-based model-free algorithms is summarized in Tab. 5.1. As can be seen, the current kernel-trick based approaches lack the capability of continuous action space representation.

A unifying view of kernel-based RL w. r. t. other regularization schemes is given by Taylor and Parr [235]. Another related algorithm is called kernel-based dual heuristic programming (KDHP) [258], whose applicability to hardware was shown in [259] using inverted pendulum systems. Its *online* mechanism, however, is to run RL over simulated data and then use the final policy on the robotic system, which contradicts our requirements outlined above. Xu *et al.* [261] compare a batch KLSPI algorithm for unmanned ground vehicle control with an online actor-critic based on KDHP. Along the same lines is the more recent [88], using a kernelized RL algorithm for longitudinal control of autonomous land vehicles, operating with batch samples and ALD sparsification as well. Wang *et al.* [250] in turn approach the problem of cruise control of an autonomous vehicle by tuning the parameters of a proportional-integral controller online according to a policy learned with KLSPI. In their approach, the data samples are also collected in advance and the policy is obtained by running the batch algorithm offline.

Pioneering work to analyze the convergence of KLSPI type algorithms for large-scale or continuous state-space MDPs is reported by Ma and Powell in [144]. A rigorous analysis on solving MDPs more generally by policy iteration with kernel representations is now provided by Farahmand *et al.* [52].

5.4 Contribution

The main contribution of this chapter is to show how the OLSPI algorithm with a polynomial basis for continuous action representation [34] can be endowed with a kernel-inspired automatic feature selection method of low computational complexity. Hence, we obtain an automatic OLSPI (AOLSPI) algorithm that preserves the analizability properties of the LSPI class, yet can be applied in fashion similar to direct adaptive optimal control. Implementing our algorithm requires only a relatively small amount of modifications starting from OLSPI; nonetheless, some critical tuning parameters of the VFA are removed. Hence, practitioners will benefit by easier deployment to actual systems.

In deriving the novel algorithm, we have several side contributions. (1) We start by adding capabilities to the KLSPI from [260] to work online in the above sense, *i. e.*, under incremental data collection and reduced processing burden. Opposed to [97, 262], we discuss the role of the sparsification scheme to save computational time, based on advances in the field of kernel adaptive filtering. We then (2) obtain a modification of OLSPI's standard temporal difference (TD) update rule, which also allows for a kernel-inspired approach to distribute basis functions for the continuous state and action VFA, without actually applying the kernel trick to OLSPI. To benefit from enhanced information processing nonetheless, (3) the similarity measure of the sparsification process is used to extrapolate learned information to new dictionary elements. Hence, (4) the convergence of the novel algorithm is shown to be eventually similar to that of a well-tuned OLSPI with a fixed set of BFs.

5.5 Review of State-of-the-Art LSPI

5.5.1 Least-Squares Policy Iteration

LSPI and temporal difference (TD) learning. Policy iteration (PI) is one particular method to learn $\mathcal{Q}^\star$. PI tackles the learning problem by starting with some randomly chosen policy and improving it iteratively until convergence to the optimal one. To this end, two steps are alternating. The first is *policy evaluation*, which refers to computing the state-action value function $\mathcal{Q}_\pi(\mathsf{s}, \mathsf{a})$ of the current policy. This estimate is then used in the second step, the policy improvement done via (5.5). The policy evaluation step requires to solve the Bellman equation [15] of the MDP

$$\mathcal{Q}_\pi(\mathsf{s}_t, \mathsf{a}_t) = \mathbb{E}_\pi[\mathsf{R}_{t+1} + \gamma \mathcal{Q}_\pi(\mathsf{s}_{t+1}, \mathsf{a}_{t+1}) \,|\, \mathsf{s}_t, \mathsf{a}_t]. \tag{5.6}$$

In continuous spaces $\mathscr{S}$ or $\mathscr{A}$ that typically occur in physical systems, it is in general not possible to solve the policy evaluation step exactly. In this case, the state-action value function $\mathcal{Q}(\mathsf{s}, \mathsf{a})$ is commonly approximated as $\hat{\mathcal{Q}}(\mathsf{s}, \mathsf{a})$ by means of a linear approximation architecture [34, 65]. To this end, a set $\mathcal{F} = \left\{\phi_i(\mathsf{s}, \mathsf{a}) : \mathscr{S}\mathscr{A} \mapsto \mathbb{R}, \forall i = 1, \ldots, N_\phi\right\}$ of features is selected, which consists of N_ϕ state and action dependent BFs $\phi(\cdot, \cdot)$. The approximated value $\hat{\mathcal{Q}}$ for a given state-action tuple (s, a) is then computed as a weighted sum of the BFs

$$\hat{\mathcal{Q}}(\mathsf{s}, \mathsf{a}) = \sum_{i=1}^{N_\phi} \phi_i(\mathsf{s}, \mathsf{a})\theta_i = \boldsymbol{\phi}(\mathsf{s}, \mathsf{a})^\top \boldsymbol{\theta}. \tag{5.7}$$

Solving (5.6) approximately by minimizing the approximation error in a least squares sense results in the LSPI algorithm. In its original form [128], this algorithm is offline, *i. e.*, it requires a batch of transition data samples of interactions with the environment. Busoniu *et al.* [34] present a variant that processes interactions with the environment on the fly, therefore called OLSPI. Both algorithms build a matrix $\boldsymbol{\Lambda}$ and a vector $\boldsymbol{\lambda}$ from subsequent interactions in order to solve the projected Bellman equation by TD learning according to

$$\boldsymbol{\Lambda} \leftarrow \boldsymbol{\Lambda} + \boldsymbol{\phi}(\mathsf{s}, \mathsf{a}) \left(\boldsymbol{\phi}(\mathsf{s}, \mathsf{a}) - \gamma \boldsymbol{\phi}(\mathsf{s}', \pi(\mathsf{s}'))\right)^\top, \tag{5.8}$$

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \boldsymbol{\phi}(\mathsf{s}, \mathsf{a})\mathsf{r}. \tag{5.9}$$

LSPI rebuilds these matrices in every iteration from scratch, whereas OLSPI continues to update $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ as long as it interacts with the environment.

Continuous action domains. In order to use OLSPI over scalar continuous action domains, orthogonal polynomials such as Chebyshev polynomials of the first kind $\Psi_j : [-1, 1] \mapsto [-1, 1]$ of degree j, $0 \leq j \leq M$, are used to construct an extended feature vector $\boldsymbol{\phi}(\mathsf{s}, \mathsf{a}) \in \mathbb{R}^{(M+1)N_\phi}$

as

$$\boldsymbol{\phi}(\mathsf{s},\mathsf{a}) = \begin{bmatrix} \boldsymbol{\phi}_{\mathrm{S}}(\mathsf{s})\Psi_0(\bar{a}) \\ \vdots \\ \boldsymbol{\phi}_{\mathrm{S}}(\mathsf{s})\Psi_M(\bar{a}) \end{bmatrix}, \text{ where } \boldsymbol{\phi}_{\mathrm{S}}(\mathsf{s}) = \begin{bmatrix} \phi_1(\mathsf{s}) \\ \vdots \\ \phi_{N_\phi}(\mathsf{s}) \end{bmatrix}. \tag{5.10}$$

The benefit of working with the extended feature vector (5.10) is that the approximation over the action space $\mathscr{A}$ is kept separated from that over the state space $\mathscr{S}$. In (5.10), without loss of generality, the action space $\mathscr{A}$ is scaled to exploit the orthogonality of the Chebyshev polynomials over the set $\bar{\mathscr{A}} \triangleq [-1,1]$, with the elements denoted $\bar{a} \in \bar{\mathscr{A}}$. Thus the policy improvement step (5.5) becomes tractable: computing (5.7) for the current state s results in a polynomial expression over $\bar{\mathsf{a}}$

$$\Psi(\bar{\mathsf{a}}) \; = c_M\bar{\mathsf{a}}^M + \dots + c_1\bar{\mathsf{a}} + c_0 \tag{5.11}$$

which is exactly representable by the coefficients c_j and it remains to compute $\arg\max_{\bar{\mathsf{a}}\in\bar{\mathscr{A}}} \Psi(\bar{\mathsf{a}})$ to find the greedy step (5.5) efficiently. Further details on OLSPI with Chebyshev polynomial approximation are skipped for brevity and the reader is referred to the literature [34, Ch. 5.3, p. 170ff, and Ch. 5.5, p.177ff].

Remark 5.1 *(Scalar action space).* A widespread restriction when applying LSPI algorithms is currently that the action space $\mathscr{A} \subset \mathbb{R}$ is scalar in order to keep (5.5) tractable. If the problem at hand requires a vector-valued continuous action space, one can run several instances of OLSPI in parallel. ◁

5.5.2 Kernel-Based Policy Iteration

A version of LSPI which exploits the *kernel trick* [204] to approximate the state-action value function $\mathcal{Q}(\mathsf{s},\mathsf{a})$ is presented in [260]. Similar to the linear approximation architecture, the Q function is approximated via a weighted sum of kernel functions, *i. e.*,

$$\begin{aligned} \hat{\mathcal{Q}}(\mathsf{s},\mathsf{a}) &= \sum_{i=1}^{N_\mathrm{K}} k_i(\mathsf{s},\mathsf{a})\theta_i = \boldsymbol{k}(\mathsf{s},\mathsf{a})^\top\boldsymbol{\theta} \\ \text{with} \quad k_i(\mathsf{s},\mathsf{a}) &\triangleq \kappa\left((\mathsf{s},\mathsf{a}),(\mathsf{s}_i,\mathsf{a}_i)\right), \\ \boldsymbol{k}(\mathsf{s},\mathsf{a}) &= \left[k_1(\mathsf{s},\mathsf{a}),\dots,k_{N_\mathrm{K}}(\mathsf{s},\mathsf{a})\right]^\top. \end{aligned} \tag{5.12}$$

The function $\kappa(\mathsf{x},\mathsf{x}') : \mathscr{S}\mathscr{A} \times \mathscr{S}\mathscr{A} \mapsto \mathbb{R}$ denotes the positive definite symmetric kernel function inducing a reproducing kernel Hilbert space (RKHS), *i. e.*, the feature space $\mathcal{H}$ with inner product $\langle\cdot,\cdot\rangle_\mathcal{H}$ such that

$$\kappa\left(\mathsf{x},\mathsf{x}'\right) = \langle\, \boldsymbol{\Phi}(\mathsf{x}), \boldsymbol{\Phi}(\mathsf{x}')\,\rangle_\mathcal{H}. \tag{5.13}$$

Denote by $\mathsf{x} = (\mathsf{s},\mathsf{a})$ a vector collecting state-action tuples. The mapping $\boldsymbol{\Phi} : \mathscr{S}\mathscr{A} \mapsto \mathcal{H}$ is the feature map which is implicitly defined by the kernel. The set $\mathcal{D} = \{(\mathsf{s}_i,\mathsf{a}_i) \in \mathscr{S}\mathscr{A}, i = 1,\dots,N_\mathrm{K}\}$ is a dictionary of $N_\mathrm{K} \triangleq |\mathcal{D}|$ collected state-action tuples. Roughly speaking, this set contains a finite number of points representative for the space spanned by $\mathscr{S}\times\mathscr{A}$. The main steps of the KLSPI algorithm are briefly summarized: based on the dictionary, the training data is iterated over in order to recursively solve the projected Bellman equation, leading to an improved policy. Then, the learning agent interacts greedily with its environment and produces new data samples. New samples are added to the dictionary only on per-need basis and the whole process is repeated until some convergence criterion is fulfilled. The advantage of the KLSPI algorithm is two-fold: first, the approximation of the $\mathcal{Q}$ function is computed in the RKHS; second, the set $\mathcal{D}$ of representative samples is created in automated fashion. In [260], this is done via ALD analysis applied to the dictionary state-action tuples (s, a): if a new tuple can be reasonably well represented by a linear combination of the N_K tuples already contained in the dictionary, its addition to the dictionary is not considered justified. Formally, the approximation error is calculated by

$$\delta = k_{**} - \boldsymbol{k}_*^\top \boldsymbol{K}^{-1} \boldsymbol{k}_*, \tag{5.14}$$

with $\boldsymbol{K} \in \mathbb{R}^{N_\mathrm{K} \times N_\mathrm{K}}$, $\boldsymbol{k}_* \in \mathbb{R}^{N_\mathrm{K}}$, and $k_{**} \in \mathbb{R}$ defined by the Mercer kernel κ, the training data $\mathcal{D}$, and the query input $\mathbf{x}' = (\mathbf{s}', \mathsf{a}')$ as

$$K_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D} \tag{5.15}$$

$$k_{*i} = \kappa(\mathbf{x}_i, \mathbf{x}'), \quad \forall \mathbf{x}_i \in \mathcal{D} \tag{5.16}$$

$$k_{**} = \kappa(\mathbf{x}', \mathbf{x}'). \tag{5.17}$$

Given a threshold δ_0, the ALD criterion states that $\mathbf{x}'$ is already sufficiently well represented by the dictionary if $\delta \leq \delta_0$. Accordingly, $\mathbf{x}'$ is added to $\mathcal{D}$ if $\delta > \delta_0$. For learning, a TD-like update similar to (5.8) and (5.9) is used, employing the vector of kernels $\boldsymbol{k}(\mathbf{s}, \mathsf{a})$ in place of the feature vector $\boldsymbol{\phi}(\mathbf{s}, \mathsf{a})$:

$$\boldsymbol{\Lambda} \leftarrow \boldsymbol{\Lambda} + \boldsymbol{k}(\mathbf{s}, \mathsf{a}) \left(\boldsymbol{k}(\mathbf{s}, \mathsf{a}) - \gamma \boldsymbol{k}(\mathbf{s}', \pi(\mathbf{s}'))\right)^\top, \tag{5.18}$$

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \boldsymbol{k}(\mathbf{s}, \mathsf{a})\mathsf{r}. \tag{5.19}$$

5.5.3 Problem Statement

In the notation above, it is clear that the core learning mechanism is quite similar in the LSPI, OLSPI, and KLSPI algorithms. With these well-established algorithms in mind, we are now in position to emphasize which parts of the algorithms allow for modifications in order to deploy LSPI more easily to actual robotic systems.

Xu *et al.* [260] state that their KLSPI can be used to optimize an existing policy online. This policy, however, is required to feature some level of performance. Due to this initial performance guarantee, the need for additional exploration is avoided. In spite of these assets, the KLSPI algorithms alternates between two main steps: data collection, *i. e.*, greedy interaction with the environment, and subsequent policy improvement. Data is thus processed in batches. Moreover, it is difficult to identify the required performance level of the initial policy. Note that this notion of *online* mechanism contrasts the requirements that typically occur in robotics outlined above.

Problem 5.1 *(KLSPI for online learning).* Development of an online version of KLSPI, *i. e.*, data should be processed once it becomes available while the per-iteration time must not increase significantly during run-time. ◇

The OLSPI algorithm from [34], in turn, is capable of online processing and continuous action space representations. Yet it should be clear that the choice of features ϕ_i is crucial to obtain good performance in any LSPI algorithm; as pointed out in [65, Ch. 4.5, p. 436], "[. . .] the choice of the representation can often play a much more significant role in the final performance of the solver than the choice of the algorithm." From a practitioner's point of view, this issue is ubiquitous when having to select basis functions in order to apply approximation-based RL algorithms to robotics, a tuning process that can be tedious. This process should therefore be automated.

Problem 5.2 *(OLSPI with automatic VFA).* Derivation of an OLSPI algorithm that is applicable to continuous state-action spaces and automatically selects suitable features in order to reduce hand-tuning of the VFA, or to obtain a good starting point for subsequent fine-tuning of OLSPI. ◇

5.6 Online Kernel Least-Squares Policy Iteration

The kernel-based RL approaches reviewed in Sec. 5.5 select data points based on ALD analysis. A first recursive version of KLSPI is presented in [262], however, considering only a discrete state space, using expensive ALD sparsification as well, and it lacks a convergence analysis. A first step is therefore to adopt a more efficient sparsification rule.

5.6.1 Sparsification Rule

A direct implementation of the ALD criterion (5.14) requires the inversion of a Gram matrix $\boldsymbol{K} \in \mathbb{R}^{N_\mathrm{K} \times N_\mathrm{K}}$, which results in a basic complexity of $\mathcal{O}(N_\mathrm{K}^3)$ [186]. Clearly, the per-iteration time will increase significantly with the growing dictionary; hence, the matrix inversion should be avoided. One alternative approach is to directly propagate the inverse matrix by recursive updates, similar as done in [100, 262]. However, the complexity is still $\mathcal{O}(N_\mathrm{K}^2)$ in this case; moreover, learning the inverse results in increased sensitivity w. r. t. the numeric initialization parameters. Recently, other sparsification methods are becoming more mature and well-understood, see *e. g.* [85]. We therefore propose to adopt another sparsification procedure that inherently is of only linear complexity: the *coherence* criterion introduced in [189].

The coherence μ of a dictionary $\mathcal{D}$ is defined as

$$\mu = \max_{\substack{i \neq j \\ 1 \leq i,j \leq |\mathcal{D}|}} \frac{|\kappa(\mathbf{x}_i, \mathbf{x}_j)|}{\sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i)\kappa(\mathbf{x}_j, \mathbf{x}_j)}},$$

therefore μ is large if the dictionary contains points $\mathbf{x}_i$ and $\mathbf{x}_j$ that are very similar in $\mathcal{H}$ as measured by (5.13). The decision rule whether to include a new sample $\mathbf{x}'$ into the dictionary or not is to restrict the coherence of the dictionary below a threshold $0 \leq \mu_0 \leq 1$, *i. e.*, if

$$\max_{\mathbf{x}_i \in \mathcal{D}} \frac{|\kappa(\mathbf{x}_i, \mathbf{x}')|}{\sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i)\kappa(\mathbf{x}', \mathbf{x}')}} < \mu_0, \tag{5.20}$$

then $\mathbf{x}'$ can be added to $\mathcal{D}$. In the following, it is assumed that a unit-norm kernel function is employed, *i. e.*, a kernel that fulfills $\|\kappa(\mathbf{x}, \cdot)\|_{\mathcal{H}} = 1, \forall \mathbf{x} \in \mathscr{S}\mathscr{A}$. The most well-known kernel with this property is the Gaussian kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{5.21}$$

and in this case (5.20) reduces to

$$\max_{\mathbf{x}_i \in \mathcal{D}} |\kappa(\mathbf{x}_i, \mathbf{x})| < \mu_0. \tag{5.22}$$

Hence, the complexity of the sparsification rule is reduced to $\mathcal{O}(N_\mathrm{K})$ evaluations of the kernel function and a simple element-wise comparison.

Remark 5.2 *(Babel criterion).* Instead of the maximum similarity of the datapoints (*i. e.*, the coherence) as a decision criterion, the cumulative coherence (*Babel* criterion) is sometimes considered for sparsification, see [85] for a comparison. In this case, a new data point is included in the dictionary if

$$\sum_{\mathbf{x}_i \in \mathcal{D}} |\kappa(\mathbf{x}_i, \mathbf{x})| \leq \tilde{\mu}_0.$$

Although of linear complexity as well, for the purpose of *online* RL, this sparsification is not as suitable as the maximum coherence-based diversity measure. The rationale behind will be clarified by means of the simulation study reported in Sec. 5.8. ◁

5.6.2 Online Dictionary Expansion

Rebuilding the matrices $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ in the TD update (5.18) and (5.19) from scratch after each interaction is the second shortcoming of KLSPI w. r. t. efficient online data processing. This problem can be circumvented as follows: recall that $\boldsymbol{\Lambda}$ is a sum of outer products of the two vectors $\boldsymbol{u} \triangleq \boldsymbol{k}(\mathsf{s}, \mathsf{a})$ and $\boldsymbol{v} \triangleq (\boldsymbol{k}(\mathsf{s}, \mathsf{a}) - \gamma \boldsymbol{k}(\mathsf{s}', \pi(\mathsf{s}')))$. Adding a new feature $\mathbf{x}$ to the dictionary $\mathcal{D}$ means to add one dimension u_{n+1} to $\boldsymbol{u}$ and v_{n+1} to $\boldsymbol{v}$. The resulting outer product $\tilde{\boldsymbol{u}}\tilde{\boldsymbol{v}}^\top$ becomes

$$\tilde{\boldsymbol{u}}\tilde{\boldsymbol{v}}^\top = \begin{bmatrix} \boldsymbol{u} \\ u_{n+1} \end{bmatrix} [\boldsymbol{v}^\top, v_{n+1}] = \begin{bmatrix} \boldsymbol{u}\boldsymbol{v}^\top & \boldsymbol{u}v_{n+1} \\ u_{n+1}\boldsymbol{v}^\top & u_{n+1}v_{n+1} \end{bmatrix}. \tag{5.23}$$

Thus, only one row and column is added while the other entries remain unaffected. This observation is key to retain the previous values of $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ during the subsequent rank-1 update. To this end, $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ need to be enlarged, *e. g.*, by adding an extra diagonal entry

$\Lambda_{\text{new}} \in \mathbb{R}$ to $\boldsymbol{\Lambda}$ and an extra entry $\lambda_{\text{new}} \in \mathbb{R}$ to $\boldsymbol{\lambda}$ as

$$\boldsymbol{\Lambda} \leftarrow \text{blkdiag}(\boldsymbol{\Lambda}, \Lambda_{\text{new}}) \triangleq \begin{bmatrix} \boldsymbol{\Lambda} & 0 \\ 0 & \Lambda_{\text{new}} \end{bmatrix}, \quad \boldsymbol{\lambda} \leftarrow [\boldsymbol{\lambda}^\top \quad \lambda_{\text{new}}]^\top. \tag{5.24}$$

From (5.18) we then have

$$\boldsymbol{\Lambda} \leftarrow \text{blkdiag}(\boldsymbol{\Lambda}, \Lambda_{\text{new}}) + \begin{bmatrix} \boldsymbol{u}\boldsymbol{v}^\top & \boldsymbol{u}v_{n+1} \\ u_{n+1}\boldsymbol{v}^\top & u_{n+1}v_{n+1} \end{bmatrix} = \tag{5.25}$$

$$= \underbrace{\begin{bmatrix} \boldsymbol{\Lambda} & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{u}\boldsymbol{v}^\top & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{bmatrix}}_{(\star)} + \underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \Lambda_{\text{new}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} & \boldsymbol{u}v_{n+1} \\ u_{n+1}\boldsymbol{v}^\top & u_{n+1}v_{n+1} \end{bmatrix}}_{(\star\star)}. \tag{5.26}$$

Conceptually, the resulting TD update (5.25) can be conceived of as the decomposition (5.26): it corresponds to a TD step ($\star$) as if the dictionary had not been modified, and the additional part ($\star\star$) is a TD step for the new point starting from Λ_{new}. Obviously, the values of $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ computed during prior iterations remain unchanged and therefore can be re-used directly after the dictionary is expanded. Further, it is always possible to choose $\Lambda_{\text{new}} = 0$ and $\lambda_{\text{new}} = 0$; however, a better method to obtain Λ_{new} and λ_{new} is proposed later in Sec. 5.7.1.

With these measures, we obtain the OKLSPI algorithm in Tab. 5.2. The algorithm contains basic building blocks of both the KLSPI and OKLSPI algorithms. Lines 1–4 initialize the algorithm and the control loop is set up in line 5. In line 6, either a random exploratory or the exploitative action is chosen via the standard ϵ-greedy mechanism. Line 7 describes the interaction with the environment, *i. e.*, the application of action a_t and the measurement of the successor state $\mathbf{s}_{t+1}$ and corresponding reward r_{t+1}. The lines 8–9 constitute the coherence sparsification criterion, and if needed the dictionary expansion is done in lines 10–11. The remaining lines 13–19 constitute the standard kernelized TD update. For the practitioner, it is emphasized that the policy improvement step in line 17 is of conceptual nature only: it suffices to perform the calculation in line 6 when choosing an exploitative action.

5.7 Automated Online Least-Squares Policy Iteration

Albeit online capability, the proposed OKLSPI algorithm only works with discrete action sets, a shortcoming of major concern for application on robotic devices. Recall from (5.10) that the OLSPI algorithm handles continuous action spaces by incorporating Chebyshev polynomials of the first kind into an extended feature vector $\boldsymbol{\phi}(\mathbf{s}, \mathsf{a})$. However, an analogous extension of the kernel-based LSPI algorithm is not yet known because the similarity of the features in the RKHS is computed implicitly using the kernel trick. In principle, one could analogously construct a kernel for continuous $\mathcal{SA}$ by composition with a suitable orthogonal polynomial kernel [173]. Nonetheless, the policy improvement step (5.5) could not be solved exactly anymore by means of a polynomial (5.11) because this would require to explicitly consider the feature map $\boldsymbol{\Phi}$ of (5.13). This is, however, contrary to the key idea of kernel methods that one does not need to know an explicit form of the feature map but only implicitly define it via (5.13). Therefore, we propose to rather combine the automated feature selection of the kernel-based approach with the OLSPI algorithm, which allows to

Table 5.2: Online kernel least-squares policy iteration with coherence sparsification and efficient dictionary expansion.

Algorithm 5.1 Online KLSPI (OKLSPI)

```
Input: $\kappa(\cdot,\cdot)$ — unit-norm (Mercer) kernel function
       $0 \leq \gamma < 1$ — discount factor
       $0 < \mu_0 \leq 1$ — coherence threshold
       $\{0 < \varepsilon_t < 1\}_{t=0}^{\infty}$ — exploration schedule
       $K_\theta \in \mathbb{N}$ — policy improvement interval
       $\mathbf{x}_0 = (\mathbf{s}_0, \mathsf{a}_0)$ — initial state/action tuple
$\mathcal{D} \leftarrow (\mathbf{s}_0, \mathsf{a}_0)$
$\mathbf{\Lambda}, \boldsymbol{\lambda}, \boldsymbol{\theta}, l \leftarrow 0$
$\mathbf{s}_t \leftarrow$ initial state $\mathbf{s}_0$
for $t = 0, 1, 2, 3, \dots$ do
    $\mathsf{a}_t \leftarrow \begin{cases} \text{uniform random action in } \mathscr{A} & \textbf{if } \mathrm{rand}([0,1]) \leq \varepsilon_t \\ \pi(\mathbf{s}_t) & \textbf{else} \end{cases}$
    apply $\mathsf{a}_t$, measure $\mathbf{s}_{t+1}, \mathsf{r}_{t+1}$
    $\mu = \max\left(\{|\kappa(\mathbf{x}_i, (\mathbf{s}_t, \mathsf{a}_t))| : \mathbf{x}_i \in \mathcal{D}\}\right)$
    if $\mu < \mu_0$ then
        $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{s}_t, \mathsf{a}_t)$
        $\mathbf{\Lambda} \leftarrow \mathrm{blkdiag}(\mathbf{\Lambda}, \Lambda_{\mathsf{new}}), \quad \boldsymbol{\lambda} \leftarrow [\boldsymbol{\lambda}^\top, \lambda_{\mathsf{new}}]^\top$ with $\Lambda_{\mathsf{new}}$ and $\lambda_{\mathsf{new}}$ according to (5.34)
    end if
    $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \boldsymbol{k}(\mathbf{s}_t, \mathsf{a}_t)\left(\boldsymbol{k}(\mathbf{s}_t, \mathsf{a}_t) - \gamma \boldsymbol{k}(\mathbf{s}_{t+1}, \pi(\mathbf{s}_{t+1}))\right)^\top$
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \boldsymbol{k}(\mathbf{s}_t, \mathsf{a}_t)\mathsf{r}_{t+1}$
    if $t = (l+1)K_\theta$ then
        $\boldsymbol{\theta} \leftarrow \mathbf{\Lambda}^{-1}\boldsymbol{\lambda}$
        $\pi = \arg\max_{\mathsf{a} \in \mathscr{A}} \sum_{i=1}^{|\mathcal{D}|} \theta_i \kappa((\mathbf{s}_i, \mathsf{a}_i), (\mathbf{s}, \mathsf{a}))$
        $l \leftarrow l + 1$
    end if
end for
```

use continuous space approximations. To this end, we automate the approximation over the state space by means of kernels but continue to construct the action space approximation using orthogonal polynomials. The resulting algorithm is termed *automated online least-squares policy iteration (AOLSPI)* and provides a solution to Prob. 5.2.

First, we need to build a dictionary over the state space $\mathcal{S}$ only, with an appropriate sparsification rule. To this end, we may simply adopt the previous approach, *i. e.*, a dictionary $\mathcal{D}_\mathrm{S}$ with sparsification criterion (5.22). We can now replace the basis function vector $\boldsymbol{\phi}_\mathrm{S}(s)$ in the extended feature construction (5.10) by a vector $\boldsymbol{k}_\mathrm{S}(\mathsf{s})$

$$\boldsymbol{k}_\mathrm{S}(\mathsf{s}) = [k_1(\mathsf{s}), \ldots, k_{N_\mathrm{S}}(\mathsf{s})]^\top,\ k_i(\mathsf{s}) = \kappa(\mathsf{s}, \mathsf{s}_i),\ \mathsf{s}_i \in \mathcal{D}_\mathrm{S}\ \subset \mathcal{S}, \tag{5.27}$$

with a unit norm kernel function $\kappa(\cdot,\cdot)$ and the number of dictionary elements $N_\mathrm{S} = |\mathcal{D}_\mathrm{S}|$. The corresponding feature vector $\hat{\boldsymbol{\phi}}$ is now given by

$$\hat{\boldsymbol{\phi}}(\mathsf{s}, \mathsf{a}) = \left[\boldsymbol{k}_\mathrm{S}^\top(\mathsf{s})\Psi_0(\mathsf{a}),\ \ldots,\ \boldsymbol{k}_\mathrm{S}^\top(\mathsf{s})\Psi_M(\mathsf{a})\right]^\top. \tag{5.28}$$

Next, the key question is how the growing dictionary can be handled in OLSPI. As evident from (5.28), the feature vector $\hat{\boldsymbol{\phi}}(\mathsf{s}, \mathsf{a})$ now consists of stacked state-dependent vectors of BFs $\boldsymbol{k}_\mathrm{S}(\mathsf{s})$, which are multiplied with Chebyshev polynomials of increasing, but maximum order M. Consequently, a new element in the dictionary $\mathcal{D}_\mathrm{S}$ leads to an increase of the feature vector size by $M + 1$ elements. Therefore, the adjustment of matrix $\boldsymbol{\Lambda}$ and vector $\boldsymbol{\lambda}$ after a dictionary update needs to be carried out differently than in the case of OKLSPI.

Consider how the corresponding TD update $\boldsymbol{\Delta}(\boldsymbol{\Lambda})$ of matrix $\boldsymbol{\Lambda}$ is now calculated using (5.8):

$$\begin{aligned}\boldsymbol{\Delta}(\boldsymbol{\Lambda}) &= \hat{\boldsymbol{\phi}}(\mathsf{s}, \mathsf{a})\left(\hat{\boldsymbol{\phi}}(\mathsf{s}, \mathsf{a}) - \gamma\hat{\boldsymbol{\phi}}(\mathsf{s}', \pi(\mathsf{s}'))\right)^\top \\ &= \begin{bmatrix}\boldsymbol{k}_\mathrm{S}(s)\Psi_0(a) \\ \vdots \\ \boldsymbol{k}_\mathrm{S}(s)\Psi_M(a)\end{bmatrix}\begin{bmatrix}\boldsymbol{k}_\mathrm{S}(s)\Psi_0(a) - \gamma\boldsymbol{k}_\mathrm{S}(s')\Psi_0(\pi(s')) \\ \vdots \\ \boldsymbol{k}_\mathrm{S}(s)\Psi_M(a) - \gamma\boldsymbol{k}_\mathrm{S}(s')\Psi_M(\pi(s'))\end{bmatrix}^\top.\end{aligned}$$

By examining the element of the first row and first column of $\boldsymbol{\Delta}(\boldsymbol{\Lambda})$ exemplarily, it can be observed that $\boldsymbol{\Delta}(\boldsymbol{\Lambda})$ consists of blocks, each containing a sum of outer products of the state dependent BFs vector. For example, the first block yields

$$\boldsymbol{\Delta}(\boldsymbol{\Lambda})_{(1,1)} = \boldsymbol{k}_\mathrm{S}(\mathsf{s})\boldsymbol{k}_\mathrm{S}^\top(\mathsf{s})\Psi_0(\mathsf{a})^2 - \gamma\boldsymbol{k}_\mathrm{S}(\mathsf{s})\boldsymbol{k}_\mathrm{S}^\top(\mathsf{s}')\Psi_0(\mathsf{a})\Psi_0(\pi(\mathsf{s}')).$$

Similarly, the other blocks differ only by the values of the Chebyshev polynomials that are multiplied to the two outer products $\boldsymbol{k}_\mathrm{S}(\mathsf{s})\boldsymbol{k}_\mathrm{S}^\top(\mathsf{s})$ and $\boldsymbol{k}_\mathrm{S}(\mathsf{s})\boldsymbol{k}_\mathrm{S}^\top(\mathsf{s}')$. At this point, the reasoning about outer products of growing vectors (5.23) applies, *i. e.*, the resulting matrix of the outer product of the vector of state-dependent BFs needs to be expanded by an extra row and an extra column. Note that this applies to all of the blocks in $\boldsymbol{\Delta}(\boldsymbol{\Lambda})$. By analogous derivation for the TD update of $\boldsymbol{\lambda}$ it is immediate that adding an element at every $(N_\mathrm{S} + 1)^{\mathrm{th}}$ index is required. Formally, we obtain the expansion

$$\boldsymbol{\Lambda}_{t+1} = \left[\begin{array}{c|c|c} \boldsymbol{\Lambda}_{t+1\,(1,1)} & \cdots & \boldsymbol{\Lambda}_{t+1\,(1,M+1)} \\ \hline \vdots & \ddots & \vdots \\ \hline \boldsymbol{\Lambda}_{t+1\,(M+1,1)} & \cdots & \boldsymbol{\Lambda}_{t+1\,(M+1,M+1)} \end{array}\right], \tag{5.29}$$

where each block is enlarged as

$$\begin{aligned} \boldsymbol{\Lambda}_{t+1\,(i,j)} &= \operatorname{blkdiag}\Big(\boldsymbol{\Lambda}_{t\,(i,j)},\ \Lambda_{t,\text{new}\,(i,j)}\Big), \\ \boldsymbol{\Lambda}_{t\,(i,j)} &= \boldsymbol{k}_{\mathrm{S}}(\mathsf{s})\boldsymbol{k}_{\mathrm{S}}^{\top}(\mathsf{s})\,\Psi_{i-1}(\mathsf{a})\Psi_{j-1}(\mathsf{a}) - \gamma\boldsymbol{k}_{\mathrm{S}}(\mathsf{s})\boldsymbol{k}_{\mathrm{S}}^{\top}(\mathsf{s}')\,\Psi_{i-1}(\mathsf{a})\Psi_{j-1}(\pi(\mathsf{s}')), \end{aligned} \tag{5.30}$$

and the block-partitioned vector update

$$\boldsymbol{\lambda}_{t+1} = \left[\begin{array}{c} \boldsymbol{\lambda}_{t+1\,(1)} \\ \hline \vdots \\ \hline \boldsymbol{\lambda}_{t+1\,(M+1)} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{\lambda}_{t\,(1)} \\ \lambda_{t,\text{new}\,(1)} \\ \hline \vdots \\ \hline \boldsymbol{\lambda}_{t\,(M+1)} \\ \lambda_{t,\text{new}\,(M+1)} \end{array}\right] \tag{5.31}$$

with

$$\boldsymbol{\lambda}_{t\,(i)} = \boldsymbol{k}_{\mathrm{S}}(\mathsf{s})\,\Psi_i(\mathsf{a}). \tag{5.32}$$

Again, $\Lambda_{t,\text{new}\,(i,j)} = 0$ and $\lambda_{t,\text{new}\,(i)} = 0$ are always possible choices and a preferred way to initialize the new entries is given in the next section.

The resulting automated online least-squares policy iteration (AOLSPI) algorithm is summarized in Tab. 5.3. Compared to the OLSPI algorithm reviewed in Sec. 5.5.1, only the lines 8–12 have to be added. It is therefore straightforward to enhance existing OLSPI implementations in order to realize the automatic VFA capability. Note that, as opposed to OKLSPI, the kernel activation in lines 8–10 only depends on the system state s, whereas the dependency of the extended feature vector $\hat{\boldsymbol{\phi}}$ on the action a is captured via the Chebyshev basis as in OLSPI. Therefore, the implementation of policy improvement remains tractable by means of the polynomial (5.11).

5.7.1 Similarity-Based Information Extrapolation in TD Update

Next, we examine how the online algorithms presented above process information after the dictionary expansion step. In a single TD update step, the algorithms in this chapter spread information over multiple elements of $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$, based on the similarity of the dictionary points w. r. t. the current and successor states, see (5.18) and (5.19) with (5.12), respectively (5.30) and (5.32) with (5.27). This mechanism is essential for learning, but partly disabled in the case of AOLSPI and OKLSPI: a new BF that is added to the dictionary some time after the learning process had started lacks the information that had been spread in the previous

Table 5.3: OLSPI with automatic basis function selection (AOLSPI). Bold line numbers indicate key differences w. r. t. OLSPI from [34].

Algorithm 5.2 Automated OLSPI (AOLSPI)

```
1   Input: $\kappa(\cdot,\cdot)$ — unit-norm kernel function
           $M \in \mathbb{N}$ — small integer number, highest degree of Chebyshev polynomials
           $0 \leq \gamma < 1$ — discount factor
           $0 < \mu_0 \leq 1$ — coherence threshold
           $\{0 < \varepsilon_t < 1\}_{t=0}^{\infty}$ — exploration schedule
           $K_\theta \in \mathbb{N}$ — policy improvement interval
           $\mathbf{x}_0 = (\mathbf{s}_0, \mathsf{a}_0)$ — initial state/action tuple
2   $\mathcal{D}_\mathsf{S} \leftarrow \mathbf{s}_0$
3   $\mathbf{\Lambda}, \boldsymbol{\lambda}, \boldsymbol{\theta}, l \leftarrow 0$
4   $\mathbf{s}_t \leftarrow$ initial state $\mathbf{s}_0$
5   for $t = 0, 1, 2, 3, \ldots$ do
6       $\mathsf{a}_t \leftarrow \begin{cases} \text{uniform random action in } \mathscr{A} & \textbf{if } \mathrm{rand}([0,1]) \leq \varepsilon_t \\ \pi(\mathbf{s}_t) & \textbf{else} \end{cases}$
7       apply $\mathsf{a}_t$, measure $\mathbf{s}_{t+1}, \mathsf{r}_{t+1}$
8       $\mu = \max\left(\{|\kappa(\mathbf{s}_i, \mathbf{s}_t)| : \mathbf{s}_i \in \mathcal{D}_\mathsf{S}\}\right)$
9       if $\mu < \mu_0$ then
10          $\mathcal{D}_\mathsf{S} \leftarrow \mathcal{D}_\mathsf{S} \cup (\mathbf{s}_t)$
11          $\mathbf{\Lambda}, \boldsymbol{\lambda} \leftarrow$ expansion according to $(5.29)-(5.32)$
12      end if
13      $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \hat{\boldsymbol{\phi}}(\mathbf{s}_t, \mathsf{a}_t)\left(\hat{\boldsymbol{\phi}}(\mathbf{s}_t, \mathsf{a}_t) - \gamma\hat{\boldsymbol{\phi}}(\mathbf{s}_{t+1}, \pi(\mathbf{s}_{t+1}))\right)^\top$
14      $\boldsymbol{\lambda} = \boldsymbol{\lambda} + \hat{\boldsymbol{\phi}}(\mathbf{s}_t, \mathsf{a}_t)\mathsf{r}_{t+1}$
15      if $t = (l+1)K_\theta$ then
16          $\boldsymbol{\theta} \leftarrow \mathbf{\Lambda}^{-1}\boldsymbol{\lambda}$
17          $\pi = \arg\max_{\mathsf{a} \in \mathscr{A}} \hat{\boldsymbol{\phi}}^\top(\mathbf{s}_t, \mathsf{a}_t)\boldsymbol{\theta}$
18          $l \leftarrow l + 1$
19      end if
20  end for
```

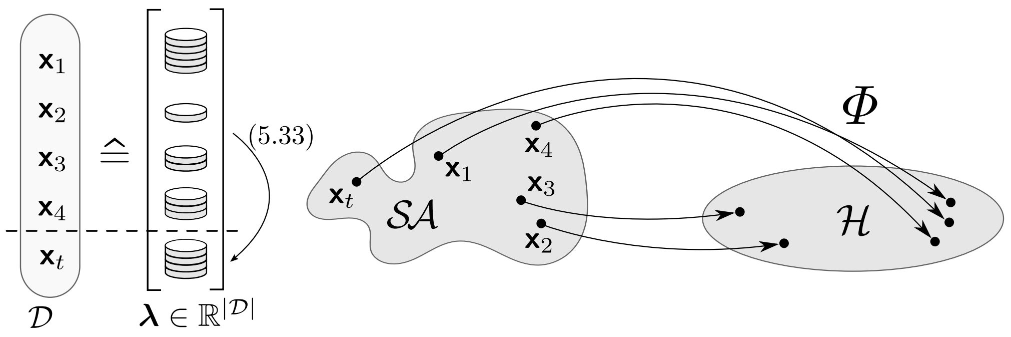


Figure 5.1: Visualization of the proposed similarity-based information extrapolation (5.33) for the TD update of $\boldsymbol{\lambda}$: according to (5.19), in each iteration, every entry of the vector $\boldsymbol{\lambda}$ receives a certain amount of the reward r determined by the kernel activation. Therefore, $\boldsymbol{\lambda}$ accumulates the rewards corresponding to each element $\mathbf{x} \in \mathcal{D} \subset \mathcal{SA}$. When the dictionary is expanded by a new element $\mathbf{x}_t$, λ_{new} can in consequence approximately be initialized with a weighted average of the collected rewards of the most similar dictionary points. Note that similarity is considered in the feature space $\mathcal{H}$: in the depicted example, $\mathbf{x}_1$ and $\mathbf{x}_4$ contribute most.

interactions with the environment. Taking $\Lambda_{\text{new}} = 0$ and $\lambda_{\text{new}} = 0$ assumes that there is not yet any information about the corresponding part of the state space— after all, it is a new point in the dictionary. By the subsequent interactions of the system with its environment, the information gap of the new BF will be closed asymptotically.

The dependency of the TD step on the similarity of the current and next states w. r. t. the dictionary elements implies, however, that regions of matrix $\boldsymbol{\Lambda}$ and vector $\boldsymbol{\lambda}$ which correspond to similar BFs should also have similar values in $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$. Hence, the similarity to the existing grid points as measured by the kernel function can be used to extrapolate entries of $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ to a new dictionary element. This idea is visualized in Fig. 5.1. While in this section, the formulas are derived to perform an approximative initialization, the numerical example in a later section will demonstrate its utility. Since the structure of $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ is dependent on the algorithm, the corresponding extrapolation rules are different and the OKLSPI-specific extrapolation is introduced first and then ported to AOLSPI.

OKLSPI. For the derivation of the basic extrapolation rule, let us revisit the TD update rule of $\boldsymbol{\lambda}$ given in (5.19), which is repeated here for the reader's convenience:

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \boldsymbol{k}(\mathsf{s}, \mathsf{a})\mathsf{r}.$$

Observe that the elements of $\boldsymbol{\lambda}$ are updated by a fraction of the received reward r as determined by the similarity of the current sample (s, a) with the elements of the dictionary. Grid points similar to each other will thus feature approximately the same values λ_i. Thus, we can safely assume that the true value of λ_{new} of a new BF should be of same magnitude as the values of $\boldsymbol{\lambda}$ corresponding to dictionary points the new BF. The value of the new element λ_{new} can therefore be obtained by extrapolation of the existing elements of $\boldsymbol{\lambda}$ weighted by the corresponding similarity, *i. e.*,

$$\lambda_{\text{new}} = \frac{\sum_{l=1}^{|\mathcal{D}|} k_l(\mathsf{s},\mathsf{a})\lambda_l}{\sum_{l=1}^{|\mathcal{D}|} k_l(\mathsf{s},\mathsf{a})}. \tag{5.33}$$

Extrapolating new elements of $\mathbf{\Lambda}$ is not as straightforward. Let us write out the TD update rule of $\mathbf{\Lambda}$ from (5.18) in expanded form:

$$\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \boldsymbol{k}(\mathsf{s},\mathsf{a})\boldsymbol{k}(\mathsf{s},\mathsf{a})^\top - \gamma\boldsymbol{k}(\mathsf{s},\mathsf{a})\boldsymbol{k}(\mathsf{s}',\pi(\mathsf{s}'))^\top.$$

The TD update of $\mathbf{\Lambda}$ consists of a subtraction of two outer products $\boldsymbol{k}(\mathsf{s},\mathsf{a})\boldsymbol{k}(\mathsf{s},\mathsf{a})^\top$ and $\boldsymbol{k}(\mathsf{s},\mathsf{a})\boldsymbol{k}(\mathsf{s}',\pi(\mathsf{s}'))^\top$. Recall that the coherence-based sparsification rule entails that the elements of the dictionary are dissimilar to a certain extent. Consequently, the first outer product mainly updates elements on the diagonal of $\mathbf{\Lambda}$. If the samples (s,a) and $(\mathsf{s}',\pi(\mathsf{s}'))$ differ, the second outer product mainly affects off-diagonal elements. To extrapolate these elements, knowledge about the policy's previous evolution would be required. In summary, we can assume that the update of the on-diagonal elements still mainly depends on the kernel vector $\boldsymbol{k}(\mathsf{s},\mathsf{a})$. Hence, an initialization for the new diagonal element Λ_{new} of the expanded matrix is obtained by a weighted average over the other diagonal elements as

$$\Lambda_{\text{new}} = \frac{\sum_{l=1}^{|\mathcal{D}|} k_l(\mathsf{s},\mathsf{a})\Lambda_{ll}}{\sum_{l=1}^{|\mathcal{D}|} k_l(\mathsf{s},\mathsf{a})}.$$

The strength of the extrapolation can be varied by actively restricting the number of considered grid points to a set $\tilde{\mathcal{D}} \subseteq \mathcal{D}$, yielding

$$\Lambda_{\text{new}} = \frac{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s},\mathsf{a})\Lambda_{ll}}{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s},\mathsf{a})}, \quad \lambda_{\text{new}} = \frac{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s},\mathsf{a})\lambda_l}{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s},\mathsf{a})}. \tag{5.34}$$

The set $\tilde{\mathcal{D}}$ can be taken, for example, by ranking the similarity to the new BF and selecting only a percentage $p_{\text{e}} \leq 1$ of most similar points. We call this approach *trust radius* in the following. The complete dictionary $\tilde{\mathcal{D}} = \mathcal{D}$ is used for $p_{\text{e}} = 1$; for $\tilde{\mathcal{D}} = \emptyset$ in turn, the conservative initialization of the new elements with zero $\Lambda_{\text{new}} = 0$ and $\lambda_{\text{new}} = 0$ is recovered.

AOLSPI. For the AOLSPI algorithm of Tab. 5.3, we adopt the extrapolation method of OKLSPI. It is essentially the same mechanism, yet applied separately to the segments of $\mathbf{\Lambda}$ and $\boldsymbol{\lambda}$. When enlarging the vector $\boldsymbol{\lambda}$ as (5.31), the newly added entry $\lambda_{t+1,\text{new}\,(i)}$ in every segment $\boldsymbol{\lambda}_{t+1\,(i)}, i = 1 \dots M+1$ is an average of the other elements of the i^{th} block segment of $\boldsymbol{\lambda}$, weighted by the similarity of the corresponding BF grid point to the grid point of the new BF, *i. e.*,

$$\lambda_{t+1,\text{new}\,(i)} = \frac{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s}) b_{t\,(i)_{(l)}}}{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s})}. \tag{5.35}$$

The values of $\mathbf{\Lambda}$ are extrapolated again in a more conservative way by considering only the block elements on the diagonal. Within these blocks $\mathbf{\Lambda}_{t+1\,(i,i)}$, the Chebyshev polynomials are equal. Hence, the two outer products are scaled by the same value and (5.30) simplifies to

$$\boldsymbol{\Lambda}_{t\,(i,i)} = \boldsymbol{k}_{\mathrm{S}}(\mathsf{s})\boldsymbol{k}_{\mathrm{S}}^{\top}(\mathsf{s})\Psi_i^2(\mathsf{a}) - \gamma_{\mathrm{S}}\boldsymbol{k}_{\mathrm{S}}(\mathsf{s})\boldsymbol{k}_{\mathrm{S}}^{\top}(\mathsf{s}')\Psi_i(\pi(\mathsf{s}'))\Psi_i(\mathsf{a}).$$

Now as in the case of OKLSPI, within the corresponding block, the first outer product $\boldsymbol{k}_{\mathrm{S}}(\mathsf{s})\boldsymbol{k}_{\mathrm{S}}^{\top}(\mathsf{s})$ updates mainly on-diagonal elements. The other outer product $\boldsymbol{k}_{\mathrm{S}}(\mathsf{s})\boldsymbol{k}_{\mathrm{S}}^{\top}(\mathsf{s}')$ further updates on-diagonal elements if s and s' are similar; otherwise, off-diagonal elements are updated depending on the policy π. The interpolation is therefore again restricted to the diagonal elements of the related block and the initialization of the new element is correspondingly

$$\Lambda_{t,\mathrm{new}\,(i,j)} = \frac{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s})\Lambda_{t\,(i,j)_{(l,l)}}}{\sum_{l=1}^{|\tilde{\mathcal{D}}|} k_l(\mathsf{s})}. \tag{5.36}$$

The number of used grid points can be selected according to a trust radius approach as in (5.34).

5.8 Discussion and Simulation Studies

Due to the limitations of value-based RL algorithms discussed in Sec. 5.2, policy search algorithms may be a more suitable choice for example in high-dimensional robotic learning control problems. If, however, an LSPI approach is appropriate for the control problem at hand, the algorithms proposed in this chapter constitute an online value-based approach capable of efficient, automatic VFA. Therefore, the task of having to explicitly distribute basis functions in a multidimensional space is avoided. While it is not expected that the presented online algorithms generally outperform their hand-tuned counterparts, a similar level of performance should be attained as by OLSPI in a well-tuned setting. In order to exemplify the two novel algorithms and evaluate their performance, we consider two standard LSPI benchmark scenarios and compare the results to those obtained with the established LSPI algorithms using well-tuned parameters.

5.8.1 Convergence Analysis

In this section, we briefly comment on the convergence of the novel algorithms. Recall that AOLSPI automates the process of selecting basis functions for OLSPI; further it is clear that the VFA plays a crucial role in the performance of OLSPI.

Remark 5.3 *(Performance guarantees of online LSPI).* Unfortunately, even the asymptotic properties of OLSPI with a fixed set of BFs are not yet completely understood, *cf.* [36, Ch. 3.6.1, p. 97]. The basic reason behind is that the policy improvement step in OLSPI is taken according to only an approximation of the value function. In consequence, the policy evaluation error may become large and the performance assertions of the basic LSPI [128] do not necessarily carry over to the online case [33]. ◁

Concerning the approximation architecture, however, Ma and Powell are able to show in [143] and [180] that under certain conditions, approximate policy iteration with Chebyshev polynomials converges in the mean. Thus, our effort is to show that the modifications introduced in this chapter do at least preserve the convergence properties of the prior algorithms. First, observe that, as proven by [189, Prop. 2], the size of the feature vector $\boldsymbol{\phi}$ converges to a fixed size at some time T, namely when the state space is completely covered with BFs as governed by the sparsification procedure and the fixed threshold μ_0. Henceforth, in all subsequent samples $t > T$, AOLSPI reduces to OLSPI as will be shown next. In the first place, the samples collected during $0 \leq t \leq T$ only contributed partly to the TD update (5.8) and (5.9) of $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$. This is because the associated BFs had not been part of the dictionary yet, hence the corresponding entries could not be updated. However, after convergence of the dictionary, *i. e.*, considering $t > T$, the feature vector basis is now fixed. We may hence think of the incomplete updates during $0 \leq T$ as some corrupted feature vectors $\boldsymbol{\phi}_{\mathrm{c}}$ affecting $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$. In the limit, the learning mechanism described by (5.8) and (5.9) becomes

$$\begin{aligned}
\boldsymbol{\Lambda} = & \underbrace{\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{T} \boldsymbol{\phi}_{\mathrm{c}}(\mathbf{s}_i, \mathsf{a}_i) \left(\boldsymbol{\phi}_{\mathrm{c}}(\mathbf{s}_i, \mathsf{a}_i) - \gamma \boldsymbol{\phi}_{\mathrm{c}}(\mathbf{s}'_i, \pi(\mathbf{s}'_i))\right)^{\top}}_{\to \mathbf{0}} \\
& + \lim_{N\to\infty} \frac{1}{N} \sum_{i=T+1}^{N} \boldsymbol{\phi}(\mathbf{s}_i, \mathsf{a}_i) \left(\boldsymbol{\phi}(\mathbf{s}_i, \mathsf{a}_i) - \gamma \boldsymbol{\phi}(\mathbf{s}'_i, \pi(\mathbf{s}'_i))\right)^{\top}, \\
\boldsymbol{\lambda} = & \underbrace{\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{T} \boldsymbol{\phi}(\mathbf{s}_i, \mathsf{a}_i) \mathsf{r}_i}_{\to \mathbf{0}} + \lim_{N\to\infty} \frac{1}{N} \sum_{i=T+1}^{N} \boldsymbol{\phi}(\mathbf{s}_i, \mathsf{a}_i) \mathsf{r}_i.
\end{aligned}$$

The limit in the first summand in both expressions exists and approaches zero as $N \to \infty$ because the sum of bounded matrices is bounded. By substitution of $i = T+1$ with $i = 0$ and reformulation, the remaining solution in the limit approaches that of the OLSPI algorithm

$$\begin{aligned}
\boldsymbol{\Lambda}^{\star} &= \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N} \boldsymbol{\phi}(\mathbf{s}_i, \mathsf{a}_i) \left(\boldsymbol{\phi}(\mathbf{s}_i, \mathsf{a}_i) - \gamma \boldsymbol{\phi}(\mathbf{s}'_i, \pi(\mathbf{s}'_i))\right)^{\top} \\
\boldsymbol{\lambda}^{\star} &= \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N} \boldsymbol{\phi}(\mathbf{s}_i, \mathsf{a}_i) \mathsf{r}_i, \qquad \boldsymbol{\theta}^{\star} = (\boldsymbol{\Lambda}^{\star})^{-1} \boldsymbol{\lambda}^{\star}.
\end{aligned}$$

In principle, (sub-)optimality of $\boldsymbol{\theta}^{\star}$ could be established according to [128, Th 7.1], *i. e.*, the error norm of the performance of the policies w. r. t. the optimal performance is in the limit bounded by some constant, subject to the restrictions of Remark 5.3 concerning online LSPI.

In summary, it is shown that the limit convergence behavior is independent of the specific dictionary sparsification method as long as $|\mathcal{D}|$ is finite, and that further the dictionary expansion and data extrapolation scheme introduced above do not void the general performance behavior of OLSPI. On the contrary, our simulation studies reported in the next section suggest that the speed of convergence may be considerably improved using AOLSPI and the scheme from Sec. 5.7.1. Analogously to the previous line of argumentation, the convergence of the OKLSPI algorithm could be analyzed given the technical assumptions in [144, 180].

5.8.2 Complexity Analysis and Optimized Implementation

Let us briefly argue that the *additional* computational complexity w. r. t. OLSPI induced by our modifications is linear in the number of dictionary elements $n = |\mathcal{D}|$, *i. e.*, an additional $\mathcal{O}(n)$ operations must be performed to implement either of the online kernel least-squares policy iteration (OKLSPI) or AOLSPI algorithms. Consider OLSPI as starting point, as it is the underlying online algorithm in both cases. For the AOLSPI algorithm, the only additional operations are those of lines 8–12 in Tab. 5.3. Note that $\boldsymbol{k}_{\mathrm{S}}(\mathsf{s}_t)$ must be computed to obtain $\hat{\boldsymbol{\phi}}(\mathsf{s}, \mathsf{a})$ of (5.28) one way or the other. Summarizing the remaining elementary scalar operations, we have an additional computational complexity of $\mathcal{O}(n)$ operations. A similar line of reasoning is applicable to OKLSPI: in terms of complexity, we can think of Tab. 5.2 as an instance of OLSPI with a discrete action space. Again, counting the remaining operations to grow the dictionary corresponding to lines 8–12 in Tab. 5.2, the added complexity is $\mathcal{O}(n)$. For implementation, an optimized version of the basic LSTD-Q algorithm is given in [128, Fig. 6], analogously for KLSTD in [97], that avoids the $\mathcal{O}(n^3)$ inversion of $\boldsymbol{\Lambda}$ by means of the matrix-inversion lemma. Our algorithms are amenable to such an approach as well: recall that the dictionary expansion and information extrapolation steps exploit the prevailing diagonal entries in the matrix structure. Therefore, similar steps could be applied when propagating the inverse matrix. Our simulation studies indicate, however, that the performance of the resulting algorithm is much more sensitive w. r. t. the numeric initialization parameter needed to avoid an ill-posed system. We therefore refrain from discussing the details here and suffice it to say that the approximations concerning the block matrix structure with single block diagonal elements remain unaffected by learning the inverse matrix directly. Thus, an optimized implementation of AOLSPI or OKLSPI based on Sherman-Morrison is feasible in principle, albeit at the cost of a more sensitive parameter set.

5.8.3 Car-on-the-Hill Benchmark

We will first illustrate how the OKLSPI algorithm of Tab. 5.2 indeed solves Prob. 5.1. In other words, it is demonstrated that the online dictionary expansion and sparsification measures proposed in Sec. 5.6 and Sec. 5.7.1 are adequate. To this end, let us consider the car on the hill problem, a standard benchmark in approximate RL that can be found in [34] and the references therein. In this task, a point mass (the car) should climb a hill by applying a horizontal force; however, the force is not strong enough to climb the hill directly. Therefore, the car needs to swing back and forth first in order to pump energy in the system. Normalizing quantities to their base SI units, the hill is modeled as a function $H(p)$, where $p \in [-1, 1]$ denotes the horizontal position of the car:

$$H(p) = \begin{cases} p^2 + p, & \text{if } p < 0, \\ \frac{p}{\sqrt{1+5p^2}} & \text{otherwise.} \end{cases}$$

With the discrete control input $u \in \{-4, 4\}$, $g = 9.81$ the gravitational constant, and $\dot{p} \in [-3, 3]$ the velocity of the car, the continuous-time dynamics are given by [34, p. 160]

$$\ddot{p} = \frac{1}{1 + \left(\frac{\mathrm{d}H(p)}{\mathrm{d}p}\right)^2} \left(u - g\frac{\mathrm{d}H(p)}{\mathrm{d}p} - \dot{p}^2 \frac{\mathrm{d}H(p)}{\mathrm{d}p} \frac{\mathrm{d}^2 H(p)}{\mathrm{d}^2 p} \right).$$

With the reward function

$$R = \begin{cases} -1, & \text{if } p < -1 \text{ or } |\dot{p}| > 3 \\ 1, & \text{if } p \geq 1 \text{ and } |\dot{p}| \leq 3 \\ 0, & \text{otherwise }, \end{cases}$$

the cost landscape as well as optimal Q-functions are discontinuous and therefore hard to approximate as shown in [34, Ch. 4.5.4]. The experiments reported next were conducted with MATLAB R2018a, using the `ode45` solver for numeric integration and a sample time of $T_\mathrm{s} = 0.1\,\mathrm{s}$ for discretization.

Let us first give an intuition how the sparsification criterion affects the dictionary growth and the computation times. In order to compare the behavior of OKLSPI with coherence sparsification according to Sec. 5.6.1 to that of ALD sparsification, we also implemented Alg. 5.1 with lines 8–9 replaced by the ALD criterion given from (5.14)–(5.17). Next, a simple parameter sweep over 99 learning runs with OKLSPI is conducted for the threshold parameters δ_0 of ALD chosen in a logarithmic scale between $[10^{-5}, 10^{1}]$, respectively μ_0 of coherence chosen linearly in the interval $[0.01, 0.99]$. The parameters of the OKLSPI algorithm are set according to Tab. 5.4 unless stated differently. Each simulation run consists of 75 trials and during each trial of 2 s, the algorithm is granted $2/0.1 = 20$ interactions with the system before being reset to a random admissible initial state. Being an online algorithm, it is essential to use sufficient exploration during the data generation and we simply use the ε-greedy mechanism. Thereby, the exploration probability in time step t is governed by

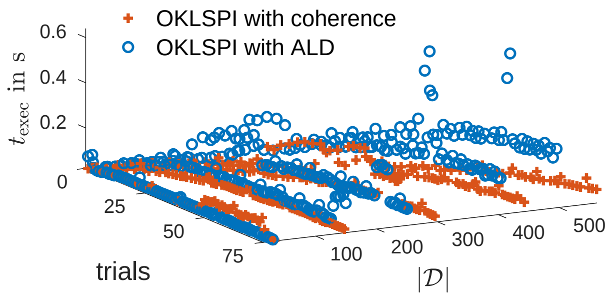
$$\varepsilon_t = \max\left(\varepsilon\Big(1 - \frac{T_\mathrm{s} \cdot t}{0.75 t_\mathrm{max}}\Big), \varepsilon_\mathrm{min}\right), \tag{5.37}$$

where $t_\mathrm{max} = 2\,\mathrm{s}$ is the duration of a single learning trial. We use a Gaussian kernel function

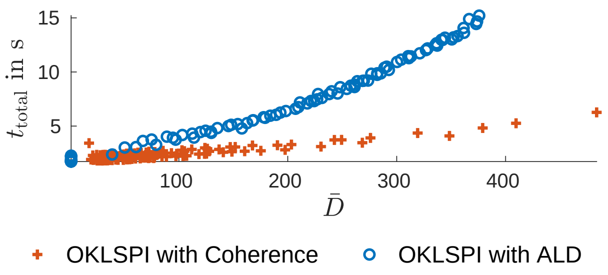
$$k_i(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mathbf{x}_i)\right). \tag{5.38}$$

In order to evaluate the influence of the sparsification criterion on the execution times of the algorithm, we used a straightforward implementation to approximately measure the calculation times t_exec for each trial. The experiment was done on a Linux machine with an Intel processor set to a constant CPU frequency of 1.8 GHz. The results of this experiment are shown in Fig. 5.2.

Figure 5.2a shows how the dictionary size $|\mathcal{D}|$ grows with increasing trials; the depicted runs were obtained by choosing values of δ_0 and μ_0 such that the amount of kernel functions in the dictionary is in the same order of magnitude for both sparsification methods. It can be seen that the execution times increase notably when ALD is used, particularly if the dictionary size is in the magnitude of hundreds. The outliers in the plot are presumably due to the imprecise method of measuring t_exec. In order to show the trend more clearly, Fig. 5.2b depicts the plot of $t_\mathrm{total} = \sum_{i=1}^{75} t_\mathrm{exec}$ over the average dictionary sizes $\bar{D} = \frac{1}{75}\sum_{i=1}^{75} |\mathcal{D}_i|$ for all the 99 runs. The measured results are consistent with the theoretical discussion in Sec. 5.6.1 concerning the complexity of the sparsification criteria. These results illustrate that the per-iteration time remains reasonable using the proposed OKLSPI algorithm with coherence



(a) Execution times t_{exec} for 75 trials of OKLSPI. The depicted runs were obtained (from left to right) for $\delta_0 \in [0.99, 0.1, 10^{-2}, 10^{-4}, 10^{-5}]$ (ALD) and $\mu_0 \in [0.1, 0.75, 0.9, 0.95, 0.97]$ (coherence).



(b) Total execution times t_{total} per trial over the average number of dictionary elements created during the 99 runs with different sparsification thresholds.

Figure 5.2: Comparison of the execution times of OKLSPI in the car on the hill problem. It can be seen that the times increase with increasing dictionary size $|\mathcal{D}|$ and that the increase is much stronger when using ALD sparsification. Therefore, the coherence criterion is more suitable for *online* reinforcement learning control with automatic VFA.

sparsification and K_θ high enough (for the fully optimistic case $K_\theta = 1$, the algorithm performs more expensive policy improvement steps in each iteration).

In order to investigate the performance of the proposed OKLSPI algorithm, the following procedure is used. The algorithm is evaluated over $N_{\text{eval}} = 90$ independent runs, where each run consists of 75 trials each starting from a random initial state and given $t_{\max}/t_{\text{S}} = 20$ interactions with the system for learning. To assess the quality of the policy over time, after each trial, the average return is calculated obtained when following the current policy without exploration for three initial states $\mathcal{S}_0 = \{[-0.8, 0]^\top, [-0.4, 0]^\top, [0, 0]^\top\}$, *i. e.*,

$$\bar{\mathsf{G}} = \frac{1}{|\mathcal{S}_0|} \sum_{i=1}^{|\mathcal{S}_0|} \sum_{j=1}^{N_{\text{test}}} \gamma_j \mathsf{R}_j. \tag{5.39}$$

The second and third initial states do not allow to drive the car up the hill just by applying the maximum input but require the policy to swing back and forth.

Table 5.4: OKLSPI parameter settings for the car on the hill problem

Parameter	Value
discount factor γ	0.97
exploration factor ε	0.95
minimum exploration $\varepsilon_{\min}$	0.05
number of BF ϕ	dynamic
kernel function κ	(5.38)
RBF variance $\boldsymbol{\Sigma}$	diag(0.1, 0.2, 0.1)
update interval K_θ	5

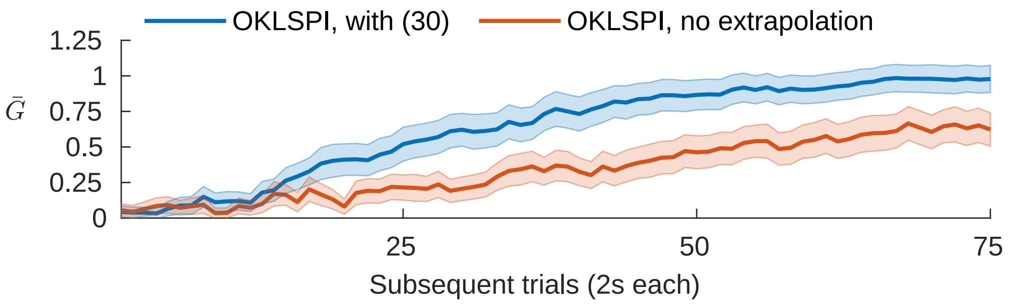


Figure 5.3: Performance of OKLSPI in the car on the hill problem with $\mu_0 = 0.9$, corresponding to an average dictionary size of $|\bar{\mathcal{D}}| \approx 240$. The figure depicts the mean score $\bar{G}$ according to (5.39) over the 90 runs (thick lines) and the corresponding 95% confidence intervals (shaded areas). The TD update information extrapolation after insertion of a new dictionary element is according to Sec. 5.7.1 with the trust radius $p_{\mathsf{e}} = 1$.

A plot of a representative learning curve is shown in Fig. 5.3 for $\mu_0 = 0.9$ and similar plots are obtained for a wide range of the sparsification parameter μ_0. The utility of the TD extrapolation scheme according to (5.34) becomes evident as well, although its effect varies with the number of useful similar dictionary elements, *i. e.*, it depends on μ_0. This example demonstrates how straightforward it is to implement and tune the algorithm, opposed to alternative value-based approaches that require more tedious tuning of the approximation architecture such as fuzzy Q-iteration, *cf.* [34, Ch. 4.5.4].

5.8.4 Automated Feature Construction for the Inverted Pendulum Problem

The second example system is the inverted pendulum with the parameters also taken from [34]. In order to balance the pendulum in the upright position, it is essential to use a continuous action-space representation; otherwise, undesired chattering around the unstable equilibrium will occur. Therefore, AOLSPI will be mainly compared to the relevant baseline algorithm OLSPI in this example. The pendulum system consists of a DC-motor with a pole attached and the goal is to steer the pole into the upright position and balance it there. The dynamics are governed by

$$\ddot{\alpha} = \frac{1}{J}\left(mgl\sin(\alpha) - b\dot{\alpha} - \frac{K^2}{R}\dot{\alpha} + \frac{K}{R}u\right), \tag{5.40}$$

where α describes the current angle of the pole, $\dot{\alpha}$ the angular velocity, and $\ddot{\alpha}$ its angular acceleration. The values of the constants J, m, g, l, b, K, and R are set identically as in [34]. The upright position is defined by $\alpha = 0$. For the simulation study, we employed a 4th order Runge-Kutta solver and a model discretization with sampling time $T_\mathrm{s} = 0.005\,\mathrm{s}$. The variable $u \in \mathscr{A}_\mathrm{p}$ denotes the input torque of the DC motor and is restricted to the continuous interval $\mathscr{A}_\mathrm{p} = [-3\,\mathrm{N\,m}, 3\,\mathrm{N\,m}]$. The state $\mathbf{s} = [\alpha, \dot{\alpha}]^\top$ of the inverted pendulum consists of the angle $\alpha \in [-\pi, \pi]$ and the angular velocity $\dot{\alpha}$, which is bounded by $|\dot{\alpha}| \leq \alpha_\mathrm{max}$, $\alpha_\mathrm{max} = 15\pi\,\mathrm{rad\,s^{-1}}$. In the following, the physical units are omitted for brevity and the quantities are given in SI unless stated differently. The state-space of the system is given by $\mathscr{S}_\mathrm{p} = [-\pi, \pi] \times [-15\pi, 15\pi]$.

The reward function is chosen as $\mathsf{R}_\mathrm{p}(\mathbf{s},\mathsf{a}) = -\,\mathbf{s}^\top \mathrm{diag}(5, 0.1)\,\mathbf{s} - \mathsf{a}^\top \mathsf{a}$ and punishes angular deviations from the upright position, high angular velocities, and large control inputs.

In order to quantify the quality of a policy, we use the following metric: for a finite set of initial states $\mathscr{S}_0$, the average $\bar{\mathsf{G}}_\mathrm{u}$ of the total undiscounted sum of rewards obtained from all initial states of $\mathscr{S}_0$ when using the current policy for $N_\text{test} = 50$ time steps is calculated, *i. e.*,

$$\bar{\mathsf{G}}_\mathrm{u} = \frac{1}{|\mathscr{S}_0|} \sum_{i=1}^{|\mathscr{S}_0|} \sum_{j=1}^{N_\text{test}} \mathsf{R}_j. \tag{5.41}$$

Note that this score function does not discount the rewards. The reward obtained when the pendulum is already swung up and needs to be balanced in the upright position is considered equally important during evaluation as the actual bang-bang like swing-up. Consequently, the effect of a discrete action set is not hidden from the performance score as it could occur with a discounted reward. As the initial state set $\mathscr{S}_0 \subset \mathscr{S}_\mathrm{P}$, we distribute 35 states over $\mathscr{S}_\mathrm{p}$ as

$$\mathscr{S}_0 = \{-\pi, -\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi\} \times \{-10\pi, -3\pi, -\pi, 0, \pi, 3\pi, 10\pi\}.$$

The parameters of each algorithm evaluated in the simulation study are given in Tab. 5.5. To assess the performance of the algorithms, we evaluate $N_\text{eval} = 90$ independent runs per algorithm. Each run consists of 300 trials of 0.75 s of interaction, *i. e.*, the system is reset to a random start state after $N_\text{trial} = 150$ interactions. The exploration in time step t is again governed by (5.37), where $\varepsilon_\text{min} = 0.05$ and $t_\text{max} = 0.75\,\mathrm{s}$ is the duration of a single learning trial.

In order to compare the AOLSPI with its hand-tuned counterpart, let us consider the number and placement of the Gaussian BFs over the state-space $\mathscr{S}_\mathrm{p}$. With the coherence threshold $\mu_0 = 0.5$, the AOLSPI algorithm creates dictionaries with $|\mathcal{D}| = 121.43$ elements on average; the distribution of the dictionary size over the 90 independent runs is depicted in Fig. 5.4. In order to compare the performance to that of OLSPI, we henceforth set the number of BFs to $N_\phi = 121$ and cover the state-space with a regular grid. The resulting placement of the BFs is shown in Fig. 5.5. It can be observed that the automated kernel function selection by AOLSPI results in a less evenly distributed grid. However, the distance between each of the BFs is approximately similar when selected according to the coherence-based update rule (5.22). We also report our findings with the Babel criterion, *cf.* Remark 5.2.

Table 5.5: Parameters used in the inverted pendulum study

Parameter	**OLSPI**	**OKLSPI**	**AOLSPI**
discount factor γ	0.99	0.99	0.99
exploration factor ε	0.95	0.95	0.95
number of BFs $\boldsymbol{\phi}$	11×11	dynamic	dynamic
RBF variance $\boldsymbol{\Sigma}$	$\mathrm{diag}(0.2, 50)$	$\mathrm{diag}(0.2, 50, 0.1)$	$\mathrm{diag}(0.2, 50)$
coherence threshold μ	–	0.5	0.5
update interval K_θ	5	5	5
degree Chebyshev M	2	–	2
action space $\mathscr{A}$	$\mathscr{A}_\mathrm{p}$	$[-3, 0, 3]$	$\mathscr{A}_\mathrm{p}$

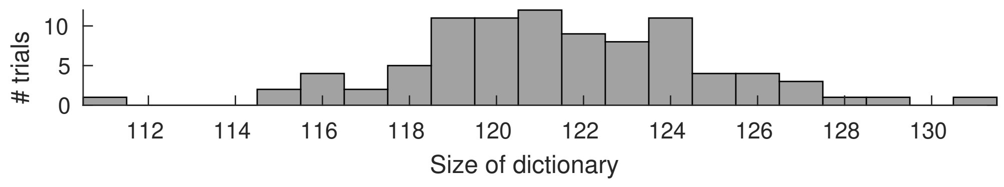


Figure 5.4: Distribution of the size of the dictionary built by AOLSPI

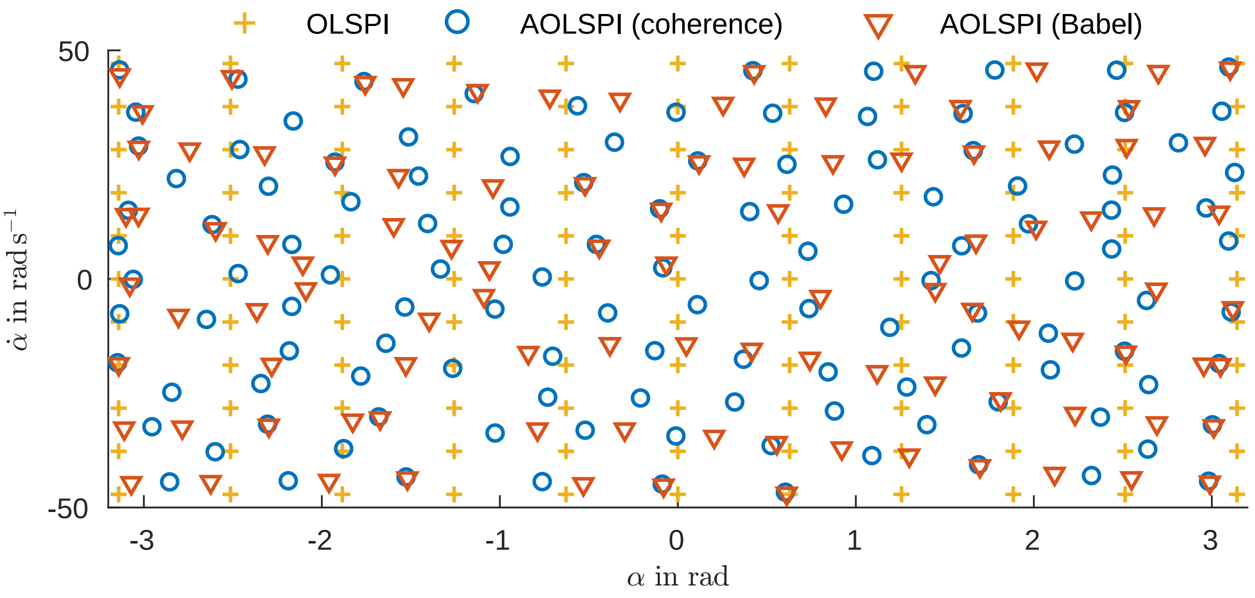


Figure 5.5: Placement of the BFs over the state-space $\mathcal{S}_\mathrm{p}$. The grid had to be set manually for OLSPI (yellow crosses), whereas the AOLSPI VFA bases were obtained automatically. Note that the typical inverted pendulum traces become visible using the Babel criterion (red triangles), whereas coherence sparsification (blue circles) leads to a good approximation throughout the state-space.

This sparsification rule is less suitable for online RL. Intuitively, this is because the BFs are not well spread over the state space. As can be seen in Fig. 5.5, rather many BFs are instead created along a particular trajectory until the threshold is reached; none can be added afterwards. Hence, the generalization capability of the value function $\mathcal{Q}$ suffers severely. This effect will not occur if *i)* the data is supplied in random order to the learning algorithm or *ii)* a suitable forgetting factor is included in the dictionary handling. In the design of OKLSPI and AOLSPI, neither is the case.

Next, the performance of the AOLSPI algorithm is investigated. Figure 5.6 shows the mean score of the 90 independent runs for both the well-tuned OLSPI and the AOLSPI algorithms. On the one hand, with OLSPI it occurs easily that the performance is far worse than depicted; it is not obvious how to select the BF grid parameters appropriately beforehand. On the other hand, note that the placement as shown in Fig. 5.5 and overall necessary number of BF is obtained automatically by AOLSPI. Performance does not suffer from this online BF selection mechanism if the information spreading mechanism from Sec. 5.7.1 is employed. It is also confirmed that the initialization of new matrix/vector entries without extrapolation from previous iterations requires a much higher number of trials until convergence; in our

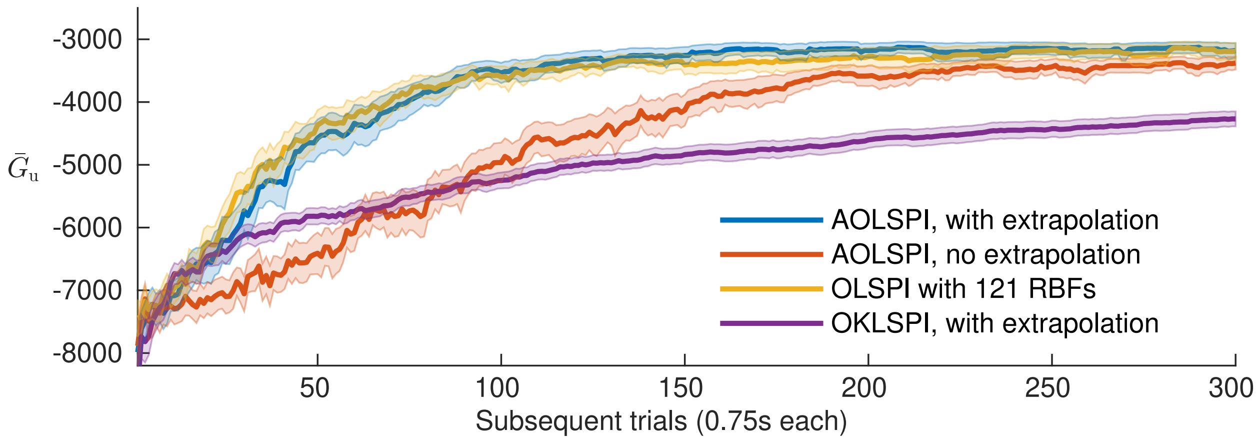


Figure 5.6: Performance comparison of OLSPI and AOLSPI. The figure depicts the mean score according to (5.41) over the 90 runs (thick lines) and the corresponding 95% confidence intervals (shaded areas). The TD update information extrapolation after insertion of a new dictionary element is according to Sec. 5.7.1.

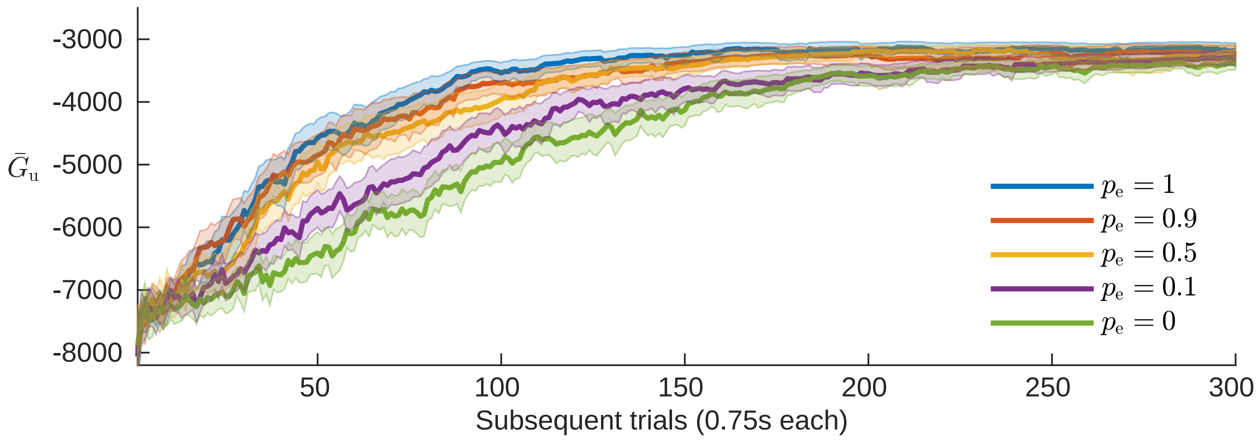
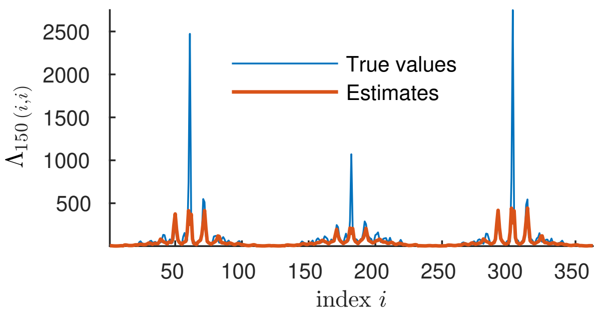


Figure 5.7: Effect of the trust radius on AOLSPI learning performance. The graph depicts the quality of the policy in the subsequent trials computed according to (5.41). A clear improvement in convergence is apparent for approximately $p_e \geq 0.5$, *i. e.*, the 50% most similar features are used for information extrapolation according to (5.35)–(5.36).

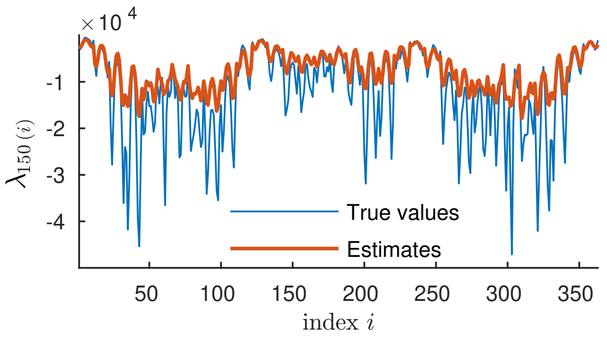
simulation, AOLSPI without extrapolation does not even reach the same performance level within the given 300 trials.

The simulation results shown in Fig. 5.6 further underline the benefit of using a continuous action space representation for the pendulum problem. Note that the performance is measured according to (5.41), *i. e.*, undesired chattering of the pendulum around the unstable equilibrium is notably penalized. Hence, although the OKLSPI algorithm fully uses the kernel trick, it fails to reach a similar level of performance as the other algorithms which employ the continuous action space approximation based on Chebyshev polynomials.

We now examine the influence of the extrapolation from Sec. 5.7.1 closer w. r. t. the performance of AOLSPI. In order to assess the influence, we performed additional runs with AOLSPI and the trust radius varying between only a little ($p_e = 0.1$), a medium amount ($p_e = 0.5$), and nearly full ($p_e = 0.9$) extrapolation. The results are shown in Fig. 5.7. All



(a) Extrapolation of diagonal values of $\boldsymbol{\Lambda}$



(b) Extrapolation of vector $\boldsymbol{\lambda}$ elements

Figure 5.8: As no ground truth is available to reflect the online situation, this graph shows an a posteriori comparison of estimated diagonal entries of $\boldsymbol{\Lambda}_{150}$ and estimated entries of $\boldsymbol{\lambda}_{150}$ w. r. t. their true values. Although this comparison cannot accurately reflect the situation during the online algorithmic execution, it is apparent that the corresponding values will be predicted correctly to a certain extent.

existing BFs may be used to build $\tilde{\mathcal{D}}$ in this particular simulation study. This is expected due to the Gaussian kernel (5.38) and the spread according to Tab. 5.5, which yields low correlations quickly for distant BFs. If, depending on the parameters, the information is not well spread during the dictionary update, it may nonetheless be useful to set $p_{\mathrm{e}} < 1$.

5.8.5 Additional Discussion of the Similarity-Based Extrapolation

With the simulation results reported above, the utility of the proposed TD information update rule is already evident. Let us nonetheless discuss in closer detail how (5.35) and (5.36) predict useful values for the initialization after the dictionary expansion, hence allowing for more efficient TD updates. Unfortunately, a quantitative evaluation of the extrapolation is not feasible because there is no accessible ground truth for yet incompleted dictionaries. Instead, we exemplarily examine the estimation of $\Lambda_{t,\mathrm{new}\,(i,i)}$ and $\lambda_{t+1,\mathrm{new}\,(i)}$ in an a posteriori analysis. To this end, we consider one of the matrices explicitly. Let us take $\boldsymbol{\Lambda}_{150}$ and $\boldsymbol{b}_{150}$ at the end ($t = 150$) of run 1, trial 1. Given $M = 2$ and $N_{\mathrm{S}} = |\mathcal{D}| = 121$ at the end of this trial, we have $\boldsymbol{\Lambda}_{150} \in \mathbb{R}^{363\times 363}$ and $\boldsymbol{\lambda}_{150} \in \mathbb{R}^{363}$. The (diagonal) values of $\boldsymbol{\Lambda}_{150}$ and $\boldsymbol{\lambda}_{150}$ are now one after another set to zero and estimated according to (5.35) and (5.36), based on the remaining (diagonal) values of $\boldsymbol{\Lambda}_{150}$ and $\boldsymbol{\lambda}_{150}$. The result is illustrated in Fig. 5.8. It can be seen that the similarity weighting interpolation approach can reflect the trend of the elements of $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$, although the peaks may be missed. As expected, the estimates are rather conservative because (5.35) and (5.36) essentially compute locally weighted means, *i. e.*, the relevant neighborhood is determined by the variance of the BFs functions. Hence, in order to capture either highly varying or very smooth relations in $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$, one would be forced to tune the variances. At this point, one would not reduce the burden of parameter tuning by means of this approach. However, as shown by Fig. 5.7, it is sufficient to add a rough prediction to improve the convergence speed. In summary, the diagonal similarity-weighting extrapolation (5.35) and (5.36) constitutes a simple yet efficient method to accelerate the online learning process in the face of dynamic dictionary growth.

5.9 Conclusion

In this chapter, we have investigated the well-known least-squares policy iteration algorithms KLSPI and OLSPI in view of their applicability to intelligent real-time automation, *e. g.*, robotic control problems. The KLSPI algorithm is reformulated for incremental data collection, yielding the proposed OKLSPI for online usage. To this end, we adopt an efficient sparsification scheme from kernel adaptive filtering and derive a recursive dictionary expansion scheme with corresponding parameter update rule. The OLSPI can be endowed with an automatic basis function selection method by a similar course of action, effectively reducing the amount of required hand-tuning. The resulting AOLSPI algorithm is applicable to continuous state-action domains as well. A similarity-based temporal difference information extrapolation scheme recovers the learning performance of the basic algorithms and we show that the convergence properties remain unaffected by our modifications. The utility of the novel algorithms is finally demonstrated by means an illustrative simulation study.

The proposed algorithms constitute within the value function based approaches a further step towards the important goal of powerful online learning robot control. While the novel AOLSPI algorithm allows for continuous action space representations, this is not yet the case for OKLSPI, leaving room for future work. Moreover, automating the selection of the kernel hyper-parameters remains an important yet in general challenging research question.

6 Reinforcement Learning and Model-Based Controller Parameterizations: a Synergistic Perspective

The previous chapter analyzed and extended LSPI, a specific class of RL algorithms, with a focus on applicability in robotics applications. In this chapter, we will examine the general interplay of RL with the Q-parameterization of stabilizing controllers. In other words, we will consider a control strategy mixing the ideas behind learning-Q control with a RL mechanism used for performance enhancement. The corresponding control scheme will be referred to as *Reinforcement Learning-Q Control* in the following.

This chapter is structured as follows. First, we introduce a categorization of RL algorithms and disambiguate the terminology w. r. t. control. Next, we consider different architectures of the plug-in filter Q in conjunction with a RL algorithm and discuss their properties from stability and learning point of views. Then, modifications are proposed to enforce a gain bound over value-based RL agents such that robust stability can be established in double-Youla fashion. Finally, the effect of pre-structuring the control loop with a Q-parameterization setup is considered from the RL perspective. The chapter concludes by summarizing the Reinforcement Learning-Q control framework and by providing recommendations for an effective interplay between both model-free learning and the model-based controller parameterization. The student work [205] partly contributed to this chapter.

6.1 Reinforcement Learning of the Performance Enhancement Filter

In this section, we discuss control architectures mixing the parameterization of stabilizing controllers with RL to implement the learning mechanism in the controller. It will be shown that the applicability of RL within the Q-parameterization differs depending on the class of RL algorithm.

6.1.1 Categorization of RL Algorithms

First, the three dominant types of RL algorithms are listed, commonly distinguished according to their core functional principle.

- *Value-based RL:* algorithms of this type aim to find optimal policies indirectly via the value function. A policy $\pi^\star$ that maximizes the return can then be inferred from the optimal state-action value function $\mathcal{Q}^\star$ by

$$\pi^\star(\mathsf{a}_t \mid \mathbf{s}_t) = \underset{\mathsf{a}_t \in \mathscr{A}}{\arg\max}\, \mathcal{Q}^\star(\mathbf{s}_t, \mathsf{a}_t). \tag{6.1}$$

  A typical well-known algorithm of this kind is *Q-learning* [252].
- *Policy-based RL:* these methods do not rely on value functions but rather parameterize the policy $\pi(\boldsymbol{\theta})$ directly, subsequently aiming to learn optimal parameters $\boldsymbol{\theta}^\star$. One of the earliest approaches of this kind is the *REINFORCE* algorithm [254].
- *Actor-critic RL:* the third class of algorithms aims to combine the advantages of both value- and policy-based RL, see *e. g.* [229]. The general idea of these algorithms is to learn a value function, the *critic*, which is used to enhance the optimization process in the policy-based methods, *i. e.*, the *actor*.

In addition, the following terms are needed to classify RL algorithms w. r. t. usage in a *Q*-parameterization.

- *Offline or online*: *offline* algorithms operate on some set of previously collected data, whereas interacting with the environment and on-the-fly learning from the data so obtained refers to *online* learning.
- *Model-based or model-free*: these terms are ambiguously used depending on the background. In the control literature as well as in this thesis, *model-based* refers to control designs upon or including dynamic process or disturbance models that were constructed in advance. From this perspective, all RL algorithms are *model-free* because an *a priori* mathematical model of the environment is not provided to the agent. Contrary to that, in the RL community, *model-free* algorithms learn straightly via a value function, whereas *model-based* RL refers to algorithms that during learning build a model of the decision process from the data; subsequently, planning methods such as DP can be used to infer the policy.
- *Critic-only, actor-only, policy search*: Value-based RL and policy-based RL algorithms are also referred to as *critic-only* and *actor-only*, respectively. RL methods that directly learn a parameterized policy (actor-only), are called *policy search* algorithms.
- *Episodic* learning: in an episodic setting [228, Ch. 3], the system is controlled with a fixed policy for a certain amount of time T, yielding a so-called *roll-out*, *i. e.*, a state/action trajectory and a sequence of rewards $\{\mathsf{R}_0, \mathsf{R}_1, \dots, \mathsf{R}_T\}$. After observing the roll-out, the policy is updated and the learning process iterates with a state reset. This form of RL is closely related to black-box optimization (BBO) [225], specifically when the parameters of the policy have to be updated and only the sum of rewards obtained along the trajectory is available for optimization at the end of an episode.

Naturally, many more criteria can be considered to categorize algorithms, *e. g.*, in terms of approximations used, on- or off-policy learning, (sub-)optimality, convergence behavior, computational complexity, exploration mechanism, and so on. The interested reader is referred to [115, 46] for relevant surveys in this area.

Remark 6.1 *(Terminological ambiguities).* RL terminology originating from artificial intelligence may be different from the notions familiar in automatic control. In this thesis, we always refer to the control-relevant terms unless stated differently. In Tab. 6.1, the relevant shared terms are contrasted in order to clarify their relationship. For example, the notions of *model-based* and *model-free* are differing between RL and control where *indirect* and *direct* [11] more closely reflect the corresponding underlying approach. Therefore, distinct typography is used throughout the thesis to disambiguate RL quantities from the signals occurring in the model-based Q-parameterization. It is moreover emphasized that *learning-Q* (*cf.* Sec. 2.4) control is fundamentally different from the renowned RL algorithm *Q-learning* [252]. $\triangleleft$

6.1.2 Two-Degree-of-Freedom Classification of RL Algorithms

While it is standard to distinguish algorithms according to the categories above, for usage in a two-degree-of-freedom Q-parameterization (Fig. 2.3b) it is vital to classify algorithms w. r. t. feedforward and feedback control. This aspect is rarely explicitly considered in the literature, not even in the review of Buşoniu *et al.* [35] dedicated to specifically cover RL from the viewpoint of the control engineer.

In fact, the specific application in a feedback or feedforward control setting dictates the classes of RL that are suitable.

- *Feedback control*: for learning feedback controllers, critic-only and actor-critic methods are a natural choice. Thereby, a MDP formulation (Def. 5.1) is the basis for a value-based approach of ranking states or state/action pairs. When employing critic-only and actor-critic methods in feedback control, the underlying problem is (often implicitly) considered to match the MDP formalism. The reward function may then correspond to

Table 6.1: Comparison of terminologies, contrasting usage in RL (respectively AI) and control communities

RL (AI)	**Automatic Control**
environment	controlled (generalized) plant G
transition function P_s^a	system dynamics f
agent	controller
state s	measured output $\boldsymbol{y}$, possibly coincides with state $\boldsymbol{x}$
state space $\mathscr{S}$	set of possible outputs $\mathcal{Y}$
action a	control input $\boldsymbol{u}$
action space $\mathscr{A}$	set of admissible control inputs $\mathcal{U}$
policy π	control law K
reward r	performance index, (negative) cost
model-based	indirect, data-driven
model-free	direct
stability	convergence

the performance ranking of the policy in terms of control error which directly depends on the system state and the actions chosen by the agent.

- *Feedforward control*: in contrast, the MDP formulation is ill-suited to cover feedforward learning control problems [205]. As emphasized in Remark 2.2, feedforward elements constitute manipulated inputs $\boldsymbol{u}_{\mathrm{ff}}$ from independent, exogenous inputs $\boldsymbol{w}$. However, it is usually not meaningful to rank independent signals in order to determine rewards; more severely, the transition function $P_s^a : \mathscr{S} \times \mathscr{A} \mapsto \mathscr{S}$ is degenerate because none of the actions taken can influence the transitions of the exogenous signal. Critic approaches are therefore prohibitive when the goal is to learn feedforward control. Concerning actor-only methods, the situation is different. Instead of gathering rewards after each state transition, actor-only methods are mostly applied in episodic fashion. As pointed out above, these algorithms are closely related to optimization and do not rely on the MDP framework to model the environment.

Given this background next we will consider RL in the Q-parameterization.

6.1.3 Architectures for RL in a Q-parameterization: Different Levels of Integration

We discuss four basic different variants how RL can be restricted to yield only stable control loops by means of a parameterization of stabilizing controllers, as well as their feedforward counterparts. These options are visualized in Fig. 6.1. The level of integration and decisive power granted to the RL agent is ascending from $\boxed{1}$ to $\boxed{4}$ in the following architectures.

For now, it is assumed that the Q-parameterization contains an ideal model of the plant under control, *i. e.*, $S = 0$ and the parameterization is according to Prop. 2.1.

$\boxed{1}$ **Learning how to switch between a set of controllers $\mathcal{Q} = \{Q_1, \dots, Q_{N_\mathsf{K}}\}$.** Logic-based switching among a family of candidate controllers constitutes a popular framework to implement adaptive controllers, an approach termed *supervisory* control in the switching systems literature [82]. A discrete decision logic thereby dictates for each time instance which of the $i \in \mathcal{I}$ pre-designed controllers is plugged into the loop. Consequently, one could attempt to learn such a supervisor, *i. e.*, the decision making component of the adaptation mechanism, by means of RL. In other words, the discrete, finite action space is $\mathscr{A} = \mathcal{I}$ and the policy π corresponds to the switching signal σ. Opposed to adaptive control designs that ensure, for example, some dwell-time constraint [137], the RL agent could choose another controller in every time instant. Therefore, for the purpose of switching controllers by means of RL, it is a principled advantage to use a Q-parameterization: as discussed in Sec. 2.3, stability under arbitrary switching between linear controllers can be ensured provided an appropriate state-space realization [80] in the parameterization of stabilizing controllers is used, *cf.* Sec. 3.1.5. Thus, opposed to methods such as [176, 56], the actions of the RL agent cannot lead to switching-induced instability of the feedback loop.

$\boxed{2}$ **Interpolation of a set of controllers $\mathcal{Q}$.** Next, recall from Sec. 2.3 that the stability conditions for arbitrarily fast switching systems are identical to those of certain interpolated systems; see also Prop. A.5 in Appendix A.2. Despite this close relation from the controls

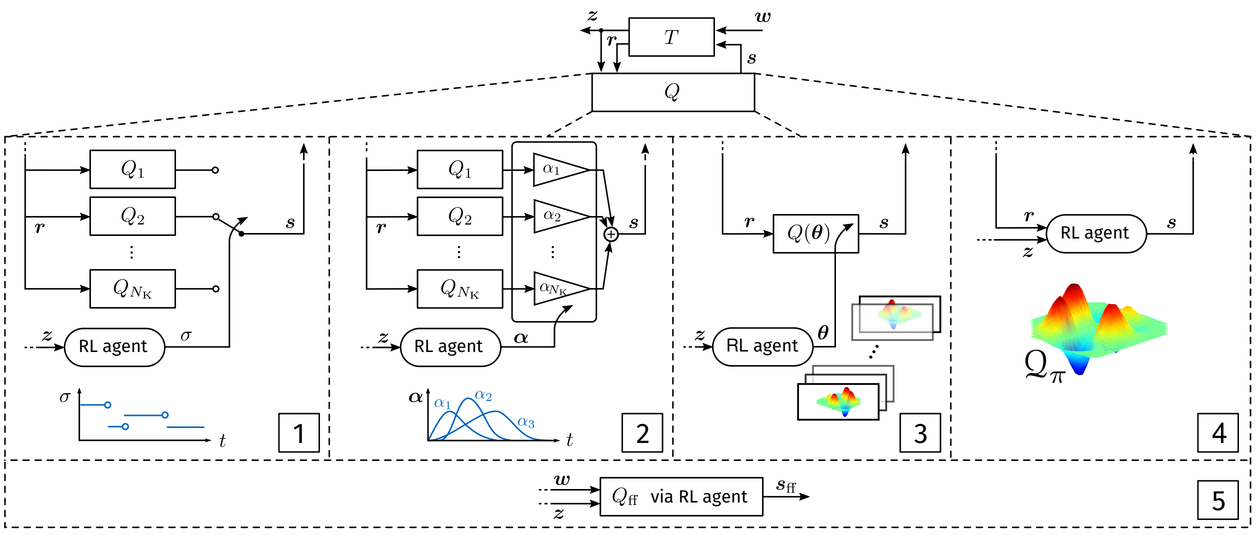


Figure 6.1: Conceptual architectures of learning feedback control in the Q-parameterization by RL. The parameterization is constructed as spelled out in part I of the thesis, yielding a pre-stabilized loop T and a "plug-in" filter Q. The learning agent aims to maximize the return, usually (but not necessarily) defined using the performance signals $\boldsymbol{z}$. Sorting from left to right is according to ascending authority/expressiveness granted to the RL agent.
$\boxed{1}$ Learning to switch between a set of controllers. The policy π dictates the piecewise continuous switching signal σ. $\boxed{2}$ Stable interpolation of some pre-designed controllers. The policy returns a vector-valued continuous interpolation signal $\boldsymbol{\alpha}$. $\boxed{3}$ Learning a parameterized Q-filter. The policy determines the active parameter vector $\boldsymbol{\theta}$. $\boxed{4}$ Replacement of Q-filter by a (value-based) RL agent. The policy directly returns the control input $\boldsymbol{s}$. $\boxed{5}$ Feedforward variants of these architectures.

perspective, for the RL architecture, it makes an important difference if controllers should be switched as in $\boxed{1}$ or interpolated: denote by $\mathcal{A}$ the set of admissible interpolation signals $\boldsymbol{\alpha}$ from (3.5). Consequently, the policy to be learned is a mapping $\pi : \mathbb{R}^{n_z} \mapsto \mathcal{A}$ and, opposed to $\boxed{1}$, a RL algorithm over continuous action spaces is required.

Remark 6.2 *(Prior control designs).* It is clear that a set $\mathcal{Q}$ of controllers is needed in $\boxed{1}$ and $\boxed{2}$ prior to learning the policy. Therefore, these approaches are advantageous particularly if the goal is to learn how to choose or interpolate from some specifically designed controllers. This scenario is particularly relevant to robotics control, for example, the variable impedance control problem from Sec. 3.1.1. For a variable impedance task, hard-coding or pre-programming of appropriate stiffness schedules is tedious in practice, making learning an attractive approach [32, 121]. Nonetheless, it is important that the controllers Q recover basic high or low impedance behavior when plugged in the loop. For other control settings, candidate controllers can be naturally designed by optimization-based methods directly in the parameterization, [28, Ch. 15]. $\triangleleft$

Remark 6.3 *(Low-level multi-controller mixing).* Architecture $\boxed{2}$ refers to the case when the action chosen by the RL agent directly dictates the interpolation signal $\boldsymbol{\alpha}$. This is in contrast to designs such as [126] that implement a low-level multi-controller to generate finely-tuned signals from some candidate controllers pre-selected by a supervisory module. That is, the design of a low-level mixer as in [126] constitutes a way to reduce the algorithmic requirements of the RL agent in architecture $\boxed{2}$ into those of $\boxed{1}$. $\triangleleft$

$\boxed{3}$ **Learning a parameterized policy $Q(\boldsymbol{\theta})$.** While $\boxed{1}$ and $\boxed{2}$ employ a set of pre-designed controllers $\mathcal{Q}$, it is also common to parameterize the filter Q with an adaptively varying parameter vector $\boldsymbol{\theta} \in \mathcal{P}$, where $\mathcal{P}$ is a set of admissible parameters. Consequently, with the adaptation carried out by RL, the policy determines $\pi : \mathbb{R}^{n_z} \mapsto \mathcal{P}$. Care must be taken to ensure that $Q(\boldsymbol{\theta})$ implements a stable filter $\forall \boldsymbol{\theta} \in \mathcal{P}$, just as in (2.29) in Sec. 2.4 of the classic adaptive-Q method. For example, one approach is to form a RITZ approximation [28, Ch. 15]. To this end, let $Q(\boldsymbol{\theta}) \in \mathcal{Q} \subset \mathcal{RH}_\infty$, where $\mathcal{Q}$ is a subspace of all stabilizing controllers spanned by a collection of pre-designed stable filters; in discrete-time, the simplest choice is a set of time-delays [28, Ch. 16.4.1], resulting in a finite impulse response (FIR) filter structure with mutable parameters. In the context of RL in the Q-parameterization, such an approach was taken in [190] and in the laboratory case study reported in Chap. 8.

$\boxed{4}$ **Replacement of plug-in filter by a (value-based) RL policy.** Algorithms of the value-based RL approach encode policies via (6.1) implicitly in the $\mathcal{Q}$-function. Therefore, implementing this approach as a controller in the Q-parameterization of stabilizing controllers corresponds to having the Q-filter replaced by such a policy. In contrast to the widespread, standard approach of learning a policy that directly maps measurements $\boldsymbol{y}$ to inputs $\boldsymbol{u}$, stability of the agent is then sufficient to retain stability of the closed loop. However, the signal $\boldsymbol{r}$ cannot be taken as a state in a critic-only learning approach; this issue will be discussed in greater detail in the next section.

$\boxed{5}$ **Learning feedforward control action.** Recall from the structure of Fig. 2.3b that the parameterization of two-degree-of-freedom controllers can be constructed by augmenting the feedback Q-parameterization with a feedforward filter Q_{ff} that generates an additive signal $\boldsymbol{s}_{\text{ff}}$ from exogenous inputs $\boldsymbol{w}$. Hence, replacing $\boldsymbol{r}$ with $\boldsymbol{w}$ in architectures $\boxed{1}$ to $\boxed{4}$ in principle yield analogous feedforward variants, which we collectively refer to as $\boxed{5}$.

6.1.4 Controller Parameterizations from the RL Point of View

In this section, the architectures of Fig. 6.1 are discussed from a RL perspective.

Affine parameterization and the role of the signal $\boldsymbol{r}$. A key feature of the Q-parameterization is that the closed-loop (2.12) is affine in the parameter system Q, *i. e.*, $T_{zw}(Q) = T_{11} + T_{12} Q T_{21}$. Consequently, controllers can be efficiently designed by convex optimization over the parameter Q for many standard control objectives [28]. It seems natural that this property of the parameterization is also beneficial for an RL approach to learning control. For example, Roberts *et al.* [190] argue that "the convexity of many common cost functions in the YP [...] can result in the convexity of the value function learning must descend [...]."

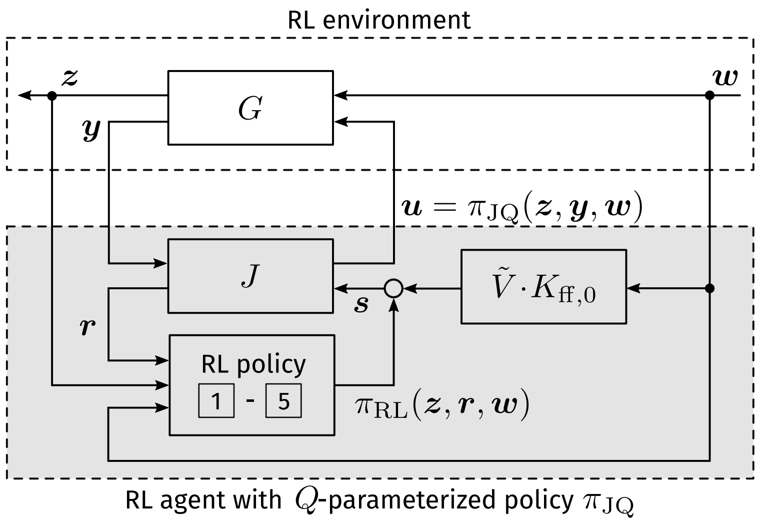

RL environment
z
w
G
y
$u = \pi_{\mathrm{JQ}}(z, y, w)$
J
s
$\tilde{V} \cdot K_{\mathrm{ff},0}$
r
RL policy
1 - 5
$\pi_{\mathrm{RL}}(z, r, w)$
RL agent with Q-parameterized policy π_{JQ}

Figure 6.2: The Q-parameterization can be considered as a specific actor structure (grey box) from the RL point of view.

However, this advantage is not preserved in all architectures presented above. As emphasized in Remark 2.1, $T_{22} = 0$ holds in the parameterization. In consequence, the signal $\boldsymbol{r}$ is essentially *feedforward* from the perspective of the learning agent because modification of Q does not affect its input $\boldsymbol{r}$. Hence, the restrictions of Sec. 6.1.2 concerning applicable algorithmic classes of RL apply. For implementation in architectures $\boxed{4}$ and $\boxed{5}$, critic-only agents using $\boldsymbol{r}$ or $\boldsymbol{w}$ as state are therefore neither suitable to replace the feedback filter Q nor to learn a feedforward controller Q_{ff}. This insight suggests that an episodic actor-only approach as in [190, 57] may be the most effective way to exploit the convexity in Q in the context of RL which holds as long as there is no significant model uncertainty ($S \to 0$).

Classification of Q-parameterization for RL. We also discuss what effect a Q-parameterization of the controller has w. r. t. the policy representation and categorization of architectures $\boxed{1}$ to $\boxed{5}$ in terms of overall RL category from Sec. 6.1.1.

A basic property of actor-only and actor-critic methods is that policies are parameterized in order to facilitate the learning process, resulting effectively in a restriction to a promising subspace of all possible policies. The two-degree-of-freedom Q-parameterization shown in Fig. 2.3b contains the central system J with the nominal controller K_0 included, possibly a nominal stable feedforward controller $K_{\mathrm{ff},0}$, and the free filter systems Q and Q_{ff} serving as parameters. Any kind of RL policy (*i. e.*, actor-only, actor-critic and critic-only) might be employed to realize the policy π_{RL} that plays the role of the Q-filter. Nonetheless, the resulting policy π_{JQ}, obtained from the feedback interconnection of nominal controller with the filter, defines a policy parameterization (actor). Hence, the overall approach constitutes an actor-only or actor-critic method [205], depending on the class of RL methods chosen for π_{RL}. This is depicted in Fig. 6.2.

6.2 Robust Stability: RL with a Gain Bound

In the previous section, different architectures to integrate RL with the Q-parameterization were considered under ideal conditions, *i. e.*, the parameterization is based on a precise model of the plant dynamics. In practice, however, inaccuracies of the model result in a double-Youla parameterization as reviewed in Sec. 2.2. In consequence, additional requirements have to be imposed on the RL agent.

In the following, let us assume that the (robust) initial controller K_0 stabilizes the plant and results in bounded uncertainty of the dual parameter $\|S\|_\infty < \gamma_\mathrm{S}$ such that the actual plant dynamics is contained in $\mathcal{G}(G_0, K_0, \gamma_\mathrm{S})$ according to (2.21). Then, with $S \neq 0$ the closed loop (2.22) is not affine in Q anymore. Moreover, stability of the policy is not sufficient to ensure stability of the closed-loop when $S \neq 0$. According to Prop. 2.8, a gain bound

$$\|\pi_\mathrm{RL}\|_\infty \leq \gamma_\mathrm{Q} = \gamma_\mathrm{S}^{-1} \tag{6.2}$$

must be imposed on the policy implemented by a RL agent. Note that $\|\pi_\mathrm{RL}\|_\infty$ refers to the induced ℓ_2 gain of the policy π.

For actor-only or actor-critic policies corresponding to architecture $\boxed{3}$, the specific structure of the policy parameterization must be considered in order to impose $\|\pi_\mathrm{RL}\|_\infty \leq \gamma_\mathrm{Q}$. To this end, the admissible parameter set $\mathcal{P}$ must be computed and subsequently $\boldsymbol{\theta} \in \mathcal{P}$ enforced during the learning process. For example, these limits could represent the weights of a feedforward neural network. How to systematically determine such parameter bounds depends on the policy structure and is beyond the scope of this thesis. Possible approaches include, for example, the gradient projection method to limit the admissible set in stochastic gradient descent based methods as in [205], or limiting the weights of a neural network by nested sector nonlinearity analyses as in *NLq-Theory* [230].

Here, we restrict attention to architecture $\boxed{4}$ and critic-only methods that infer actions from the value function $\mathcal{Q}$. In the following analysis, for simplicity, we only consider the standard case of scalar actions that are inferred from (6.1). One way to impose (6.2) in this case is to restrict the admissible action set according to the desired maximal gain [205] as follows.

Theorem 6.1 (Gain-Bounded Critic-Only). Let $\gamma_\mathrm{Q} \in \mathbb{R}_0^+$ denote the maximum permissible gain of a critic-only RL policy $\pi : \mathscr{S} \mapsto \mathscr{A}$ characterized via (6.1) by a memoryless state-action value function $\mathcal{Q}_\pi : \mathscr{S} \times \mathscr{A} \mapsto \mathbb{R}$. Then, the modified inference formula

$$\pi_\mathrm{bounded}(\mathsf{s}_t) = \arg\max_{\substack{\mathsf{a}_t \in \mathscr{A} \\ |\mathsf{a}_t| \leq \gamma_\mathrm{Q} \|\mathsf{s}_t\|}} \mathcal{Q}(\mathsf{s}_t, \mathsf{a}_t) \tag{6.3}$$

yields a restricted policy $\pi_\mathrm{bounded} : \mathscr{S} \mapsto \mathscr{A}$ such that

$$\|\pi_\mathrm{bounded}\|_\infty \leq \gamma_\mathrm{Q}. \tag{6.4}$$

Proof. Finite-gain ℓ_2-stability of the policy with ℓ_2-gain γ_Q holds for $\|\mathsf{a}\|_{\ell_2} \leq \gamma_\mathrm{Q} \|\mathsf{s}\|_{\ell_2}$. This bound can be sufficiently achieved by limiting the induced matrix norm $\|\mathsf{a}_t\| \leq \gamma_\mathrm{Q} \|\mathsf{s}_t\|$ in all time steps t,

$$\|\mathsf{a}\|_{\ell_2} = \left(\sum_{-\infty}^{\infty}\|\mathsf{a}_t\|^2\right)^{1/2} \leq \left(\sum_{-\infty}^{\infty}(\gamma_{\mathrm{Q}}\|\mathsf{s}_t\|)^2\right)^{1/2} = \gamma_{\mathrm{Q}}\left(\sum_{-\infty}^{\infty}\|\mathsf{s}_t\|^2\right)^{1/2} = \gamma_{\mathrm{Q}}\|\mathsf{s}\|_{\ell_2}, \tag{6.5}$$

which is particularly simple here as the value function $\mathcal{Q}$ constitutes a time-varying, memoryless function. Hence, with scalar $\mathscr{A} \subset \mathbb{R}$, determining outputs a_t according to (6.3), $\|\mathsf{a}_t\| \leq \gamma_{\mathrm{Q}}\|\mathsf{s}_t\|$ and consequently (6.5) hold $\forall \mathsf{a}_t \in \mathscr{A}, \mathsf{s}_t \in \mathscr{S}$. □

The critic policy from Theorem 6.1 may, however, still not yet be useful in the interplay with the Q-parameterization in architecture $\boxed{4}$ from Fig. 6.1. As discussed in Sec. 6.1.4 this is because the signal $\boldsymbol{r}$ does not usually represent a state appropriate in an MDP modeling framework. Nonetheless, $\boldsymbol{r}$ is suitable to infer the restriction of admissible actions such the gain bound holds, while actually learning over a different signal such as the performance quantities.

Corollary 6.1 (Critic-Only for Reinforcement Learning-Q). Let $\gamma_{\mathrm{Q}} \in \mathbb{R}_0^+$ denote the maximum permissible gain of a critic-only RL policy $\boldsymbol{s}_t = \pi(\boldsymbol{r}_t, \boldsymbol{z}_t)$ employed as in architecture $\boxed{4}$, *i. e.*, considering $\mathsf{s}_t \triangleq \mathrm{col}(\boldsymbol{r}_t, \boldsymbol{z}_t)$ as state of the critic generating actions $\mathsf{a}_t \triangleq s_t$. Then an admissible inference formula is given by

$$\pi_{\mathrm{Q}}(\boldsymbol{r}_t, \boldsymbol{z}_t) = \underset{|s_t| \leq \gamma_{\mathrm{Q}}\|\boldsymbol{r}_t\|}{\arg\max}\ \mathcal{Q}((\boldsymbol{r}_t, \boldsymbol{z}_t), s_t). \tag{6.6}$$

□

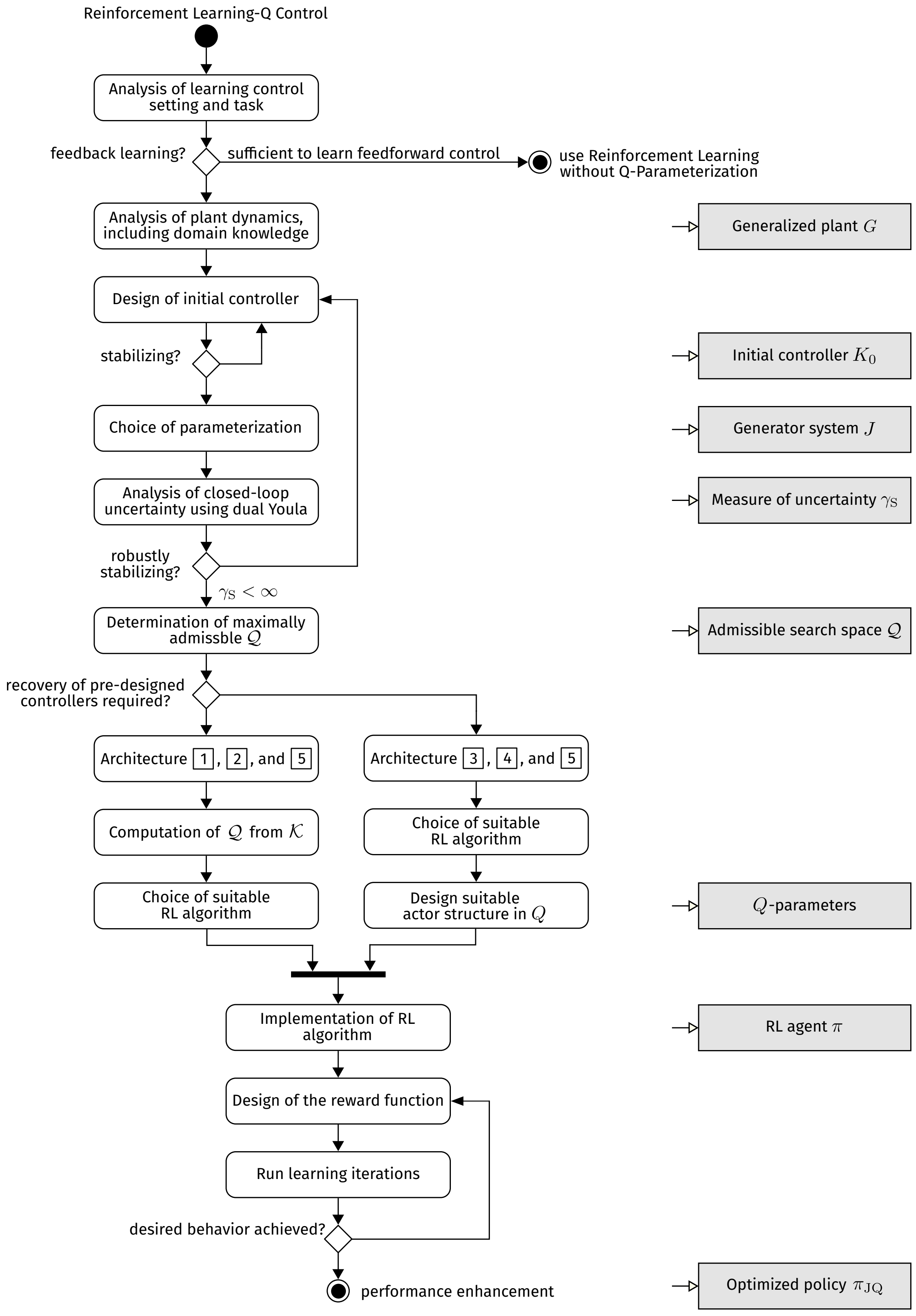
In other words, both the signals $\boldsymbol{r}$ and $\boldsymbol{z}$ are used to implement a critic-only architecture that preserves robust stability of the closed feedback loop in the double-Youla parameterization. In practice, it is sufficient to implement any critic-only algorithm learning over the performance signals $\boldsymbol{z}$ only, while restricting the filtered input $\boldsymbol{s}$ according to (6.6). For example, the AOLSPI algorithm from Chap. 5 can be straightforwardly adapted for usage in this parameterization by replacing the policy exploitation in line 17 of the algorithm in Tab. 5.3 by (6.6).

6.3 Summary of the Reinforcement Learning-Q Framework

In this section, the proposed *Reinforcement Learning-Q Control* strategy is reviewed by summarizing the high-level steps in the workflow from setting up the parameterization to performance enhancement via RL over the parameter Q.

Workflow. The steps to create a learning controller in the framework proposed in this thesis are depicted in Fig. 6.3.

At first, the learning task should be analyzed to understand if it is necessary to learn in feedback. Otherwise, the problem will be simpler to deal with by using a fixed feedback controller and learning only the required feedforward signals, for example using DMPs [93]. Next, the model-based description of the system dynamics is formulated in a generalized plant setting. The resulting G is then used to design the nominal controller K_0 which will subsequently be part of the central system J of the parameterization. If G is modeled accurately, *i. e.*, there is no uncertainty in the closed loop resulting in $\gamma_{\mathrm{S}} = 0$, it is sufficient



Reinforcement Learning-Q Control
Analysis of learning control setting and task
feedback learning?
sufficient to learn feedforward control
use Reinforcement Learning without Q-Parameterization
Analysis of plant dynamics, including domain knowledge
Generalized plant G
Design of initial controller
stabilizing?
Initial controller K_0
Choice of parameterization
Generator system J
Analysis of closed-loop uncertainty using dual Youla
Measure of uncertainty γ_S
robustly stabilizing?
$\gamma_S < \infty$
Determination of maximally admissble $\mathcal{Q}$
Admissible search space $\mathcal{Q}$
recovery of pre-designed controllers required?
Architecture $\boxed{1}$, $\boxed{2}$, and $\boxed{5}$
Architecture $\boxed{3}$, $\boxed{4}$, and $\boxed{5}$
Computation of $\mathcal{Q}$ from $\mathcal{K}$
Choice of suitable RL algorithm
Choice of suitable RL algorithm
Design suitable actor structure in Q
Q-parameters
Implementation of RL algorithm
RL agent π
Design of the reward function
Run learning iterations
desired behavior achieved?
performance enhancement
Optimized policy π_{JQ}

Figure 6.3: Abstracted workflow of the proposed Reinforcement Learning-Q control strategy. Rounded boxes outline the main activities for the design of a learning controller in the framework. The gray boxes on the right symbolize the assets resulting from the corresponding design steps.

to take K_0 stabilizing. Otherwise K_0 should be designed to yield robust stability to the nominal loop in order to ensure that $\gamma_S < \infty$ holds. This bound is subsequently needed to restrict the set of admissible parameters $\mathcal{Q}$ accordingly.

Depending on the task at hand, one of the architectures Fig. 6.1 can be chosen for implementation; these variants were discussed previously in Sec. 6.1.3. Oftentimes, the decisive question will be if the learning control scheme should be able to choose among pre-designed controllers. In this case, $\boxed{1}$ and $\boxed{2}$ are suitable. Otherwise, $\boxed{3}$–$\boxed{4}$ are more attractive; for example, one may aim to keep effort in the manual control design process low. Architecture $\boxed{3}$ then only requires to set up the parameterization itself and provide a basic structure for the search over stabilizing controllers. Such an approach will be pursued in the case study of Chap. 8, learning feedforward elements $\boxed{5}$ as well. As discussed in Sec. 6.1.2, the choice of architecture eventually also determines which classes of RL methods are applicable.

Finally, as in all RL based control schemes, the rewards must be designed to represent the desired learning control objective.

Recommendations. To set up the parameterization, in general, one could aim to exploit as much prior information as possible, including uncertainty structure or characteristics, disturbance models, as well as frequency weighting filters. No trade-off of optimality with stability is required if the dynamic plant model is accurately known ($S = 0$). Otherwise, the set of admissible parameters $\mathcal{Q}$ and correspondingly the solution space in the decision problem solved by the RL agent has to be constrained to leverage the robust stability assertions of a double parameterization. In practice, it may nonetheless be sufficient to construct the nominal model pragmatically, for example by approximate modeling of the most dominant dynamic effects. Such an approach will be taken in the case study in Chap. 8. As discussed in Chap. 4 for the case of robotic manipulators, not only the accuracy of the nominal model but also the nominal controller K_0 influence the uncertainty in the closed-loop measure of the dual Youla parameter. This degree of freedom will be used in the case study of Chap. 7 to create a parameterization for a robot with very imprecisely known dynamics.

From the learning side, the characteristic challenges of RL remain unaffected, *e. g.*, scalability, interpretability, computational times, approximation architecture, and design of the reward function to achieve desired behavior. Nonetheless, as already pointed out in [190], the performance of the learning algorithms generally depends on the controller parameterization. For the ideal case of no model mismatch ($S = 0$), convex cost functions can be constructed due to the actor structure Fig. 6.2 based on the Q-parameterization. Correspondingly, RL algorithms that operate episodically are a natural choice for performance enhancement in the proposed Reinforcement Learning-Q scheme. While Cor. 6.1 provides a basic condition to enforce the required gain bounds over a critic-only agent when $S \neq 0$, in practice actor-only or actor-critic algorithms nonetheless seem to be more suited for use in Reinforcement Learning-Q control. This is due to several reasons. First, the specific behavior during learning with a critic-only algorithm is hard to assess and interpret because one can only examine the learned state-action value function $\mathcal{Q}_\pi$. This is feasible only for low-dimensional state-action spaces. Second, the dimensionality of $\boldsymbol{r}$ and $\boldsymbol{s}$ may be prohibitive for critic-only approaches from a computational point of view, particularly in multiple-input multiple-output (MIMO) control settings. Third, including feedforward action to employ a two-degree-of-freedom control learning agent may yield higher performance and one should refrain from employing critic-only methods to learn feedforward actions due to the reasons discussed in Sec. 6.1.2. Therefore, policy search and actor-critic methods are the preferred classes of algorithms in Reinforcement Learning-Q control. The downside of these algorithms is that one must carefully design the actor parameterization and appropriately restrict the search space over the Q-parameters when there is significant $S \neq 0$.

6.4 Conclusion

In this chapter, a control strategy was introduced that mixes the Q-parameterization, a classic tool from robust control, with RL. Referring to the goal of the original learning-Q control method (*cf.* Sec. 2.4) of generalizing performance enhancement from one trajectory to another, the method considered here was dubbed *Reinforcement Learning-Q Control*. Contrary to standard RL, the approach allows to effectively handle stability in the closed feedback control loop by incorporating prior dynamical model knowledge. Consequently, the Q-parameterization can be conceived of as a structured way to create specific actor policies based on prior information. While the effort to construct the parameterization may be considerably higher compared to using some plain RL, it is an effective tool to leverage RL in control-critical scenarios. For instance, if the plant is unstable and must always be stabilized or if one cannot tolerate instability on hardware. Using RL as a performance enhancement mechanism, the method can be applied in scenarios where adaptive-Q control approaches are hardly applicable. This is the case, for instance, if the parameter search space is badly structured, disturbance models are hard to obtain, or if no model of the cost function is available (*e. g.*, subjective ranking of performance by a human teacher).

Many more opportunities arise from taking a mixed perspective of the parameterization and RL and open questions remain to be explored in future research. In particular, it would be valuable to explore how an ideally generic fallback solution could look like in order to set up the initial controller of the parameterization when uncertainty cannot be estimated or when it is hard to obtain a suitably good initial model. Working with controllers that are robust against normalized coprime factor uncertainty [148] might be a promising approach in this direction. Moreover, concerning RL, our analysis of the general interplay with the parameterization has revealed that particularly the actor-critic class of algorithms will be most promising for further research in the framework. We conjecture that algorithms based on deep NNs, such as Proximal Policy Optimization (PPO) [209] or Asynchronous Advantage Actor Critic (A3C) [151], may provide better performance enhancement capabilities for a wider range of applications. However, the open and very challenging task is then to impose suitable restrictions over the corresponding actors so as to enforce the required norm bounds without deteriorating the learning capabilities of the algorithms.

Part III

Laboratory Case Studies

7

Active Variable Impedance Control with the Parameterization of Stabilizing Controllers

In parts I and II of this thesis, methods were developed from a theoretical point of view to leverage machine learning for robotics based on stabilizing controller parameterizations. This third part serves to illustrate the methods by means of practical examples. To this end, proof-of-concept experimental studies were conducted in the laboratories. These examples showcase implementations of the proposed control strategy on real-world mechatronic devices. Therefore, this part not only provides experimental evidence of the efficacy of the proposed methods but also details our experience concerning an effective implementation on robotic hardware.

The first case study shows that scheme of Sec. 3.1 allows to leverage the benefits of a Q-parameterization on commercially available robotic hardware and to implement active variable impedance behavior. While the general behavior of the control scheme was already discussed in the simulation of Sec. 3.1.8, the following aspects about the implementation on hardware are illustrated.

- Although the dynamics of the underlying robotic control system is not known exactly due to the black-box architecture of the commercial controller, an approximate model of the resulting closed-loop dynamics is suitable to set up an internal model required for the parameterization. To this end, the nominal gains constitute a design degree of freedom to decrease uncertainty in the parameterization, while still allowing to recover via the additional input the desired control modes computed from the Q-parameters.
- This approach to implement the parameterization is enabled due to the separation structure of the control scheme shown in Fig. 4.6, rendering feasible the upgrade of a pre-implemented controller to a Q-parameterization.
- We provide experimental evidence that the proposed control architecture indeed results in stable behavior under interpolation conditions that lead to instability when naively implemented.

The study is structured as follows. We begin in Sec. 7.1 with a brief description of the control interface provided by the experimental platform and the problem setting considered in this chapter. The insights how to implement the parameterization are then provided in detail in Sec. 7.2 and the parameters to recover the desired controllers are derived in Sec. 7.3. The chapter concludes with a discussion of the experiment in Sec. 7.4 and with an outlook to future promising work in Sec. 7.5. The student works [165, 194] partly contributed to the results presented in this chapter.

7.1 Experimental Setup and Problem Setting

Robotic platform. The robot platform that was used to conduct the experiments is a KUKA LWR IV+ [26] controlled via the fast research interface (FRI) [207] in command mode; more details concerning the laboratory setup are summarized in Appendix C.3.

The starting point to augment the robot with the parameterizations of Sec. 3.1 respectively Sec. 4.7 is the built-in joint-specific impedance control mode. The torque commanded to the controlled manipulator is then approximately described by [26]

$$\boldsymbol{\tau}_{\text{cmd}} = \operatorname{diag}(\boldsymbol{k}_{\text{j}})\,(\boldsymbol{q}_{\text{des}} - \boldsymbol{q}_{\text{msr}}) + \boldsymbol{D}(\boldsymbol{d}_{\text{j}}) + \boldsymbol{\tau}_{\text{FRI}} + \boldsymbol{f}_{\text{dynamics}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}). \tag{7.1}$$

Thus, a virtual spring in joint space is realized by the built-in controller, characterized by the stiffness vector $\boldsymbol{k}_{\text{j}} \in \mathbb{R}^7$ and the error between desired position $\boldsymbol{q}_{\text{des}} \in \mathbb{R}^7$ and measured position $\boldsymbol{q}_{\text{msr}} \in \mathbb{R}^7$. The symbol $\boldsymbol{D}(\boldsymbol{d}_{\text{j}})$ denotes a damping term that cannot be modified by the user other than by setting some nominal normalized damping parameter $\boldsymbol{d}_{\text{j}} \in \mathbb{R}^7$, and $\boldsymbol{f}_{\text{dynamics}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ denotes some internal compensation term. Moreover, the interface allows to add a custom torque $\boldsymbol{\tau}_{\text{FRI}} \in \mathbb{R}^7$ to the commanded torques of each joint. The resulting block description is shown in Fig. 7.2a.

Instability in simple variable impedance control. Note that all accessible quantities of (7.1) can be changed in each time step t using the FRI interface, allowing effectively to implement a variable impedance behavior in fashion of simple gain-scheduling by varying the gains $\boldsymbol{k}_{\text{j}}$ and $\boldsymbol{d}_{\text{j}}$. Analogous to (3.2), such an approach constitutes interpolated feedback and is therefore vulnerable to instabilities arising from hidden coupling, just as it was shown in Fig. 3.2 of the simulation study of Sec. 3.1.8. For the case of constant desired inertia, Kronander and Billard [123] provide conditions to verify global uniform (asymptotic) stability for given symmetric, positive definite and continuously differentiable stiffness $\boldsymbol{K}_{\text{P}}(t)$ and damping $\boldsymbol{K}_{\text{D}}(t)$ profiles. Restricting attention to one dimension and constant damping $K_{\text{D}}(t) = K_{\text{D},0}$,

$$m\ddot{q}(t) + K_{\text{D},0}\dot{q}(t) + K_{\text{P}}(t)q(t) = 0,$$

asymptotic stability can be verified by checking if there exists an $\alpha > 0$ such that for all $t \geq 0$ [123, Ex. 1]

$$K_{\text{D},0} > \alpha m, \quad \dot{K}_{\text{P}}(t) < 2\alpha K_{\text{P}}(t). \tag{7.2}$$

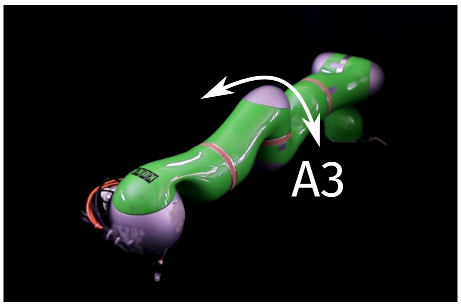
Hence, for constant damping, the rate of stiffness change admissible without instability is upper bounded by the current stiffness.

This condition was investigated experimentally [194]. The initial position $\boldsymbol{q}_0$ of the robot was brought to a horizontal pose with maximum reach as shown in Fig. 7.1a. The 1D interpolation experiment was conducted on joint A3 in the depicted position, leading to a horizontal oscillation while keeping all other joints fixed with maximum stiffness and damping control gains. For interpolation of stiffness of A3, a sinusoidal signal

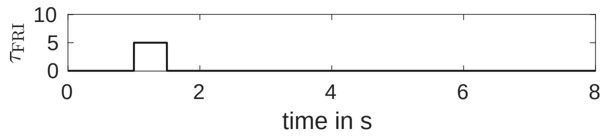
$$k_{\mathrm{A3}}(t) = K_{\mathrm{P}}(t) = 80 + 50\sin(2\pi f_{\mathrm{P}} \cdot t) \tag{7.3}$$

was used directly as input to the FRI block (7.1), varying stiffness values between 30 and 130 with frequency f_{P} as depicted in Fig. 7.1c. The damping was kept low with a constant coefficient of $d_{\mathrm{A3}} = 0.1$ and the reference trajectory for all joints is the constant horizontal pose $\boldsymbol{q}_{\mathrm{des}}(t) = \boldsymbol{q}_0 = [0\ 180\ {-90}\ 0\ 0\ 90\ 0]^\top$ throughout. A push on the robot is emulated by applying the 0.5 s long impulse with amplitude $\tau_{\mathrm{FRI}} = 5$ shown in Fig. 7.1b to the superimposed torque of joint A3. Due to this disturbance, a position error between q_{A3} and $q_{\mathrm{A3,des}}$ exists, leading to new commanded torques according to (7.1).

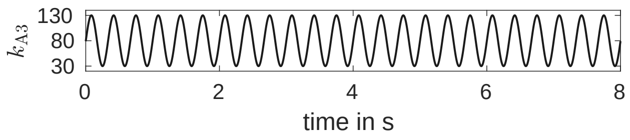
For $f_{\mathrm{P}} \lesssim 3\,\mathrm{Hz}$, the resulting oscillation is damped and converges to the depicted position of 0 deg. However, for larger frequencies $f_{\mathrm{P}} \gtrsim 3\,\mathrm{Hz}$, the magnitude of the oscillation increases. The corresponding trajectory obtained from the experiment with $f_{\mathrm{P}} = 3\,\mathrm{Hz}$ is shown in Fig. 7.1d.



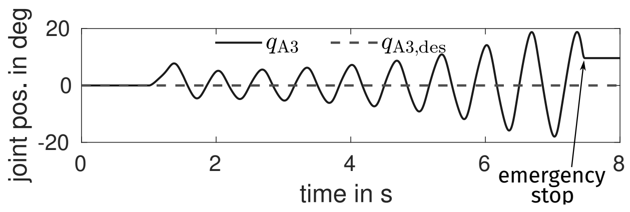
(a) Initial position for stiffness interpolation experiments conducted on joint A3.



(b) Impulse to disturb the joint from the equilibrium position



(c) Stiffness signal (7.3) used via the FRI interface



(d) Joint motion recorded for $f_{\mathrm{P}} = 3\,\mathrm{Hz}$.

Figure 7.1: A stiffness profile that violates the condition (7.2) leads to unstable behavior on the hardware when implemented by direct gain scheduling in the control law (7.1).

While it is experimentally confirmed that instability due to hidden coupling can occur on the KUKA LWR IV+, in this case study, we consider how to avoid this issue by means of the Q-parameterization.

Problem 7.1. Implement a joint-specific variable impedance control scheme on a KUKA LWR IV+ based on a Q-parameterization of stabilizing controllers to allow for arbitrary stable interpolation according to Def. 3.1. ◇

7.2 Choice of Parameterization and Internal Model

The first step in the construction of a Q-parameterization is to determine a model of the plant dynamics and the selection of the specific realization of the parameterization. To this end, the standard approach of employing a model of the open-loop plant dynamics is investigated.

7.2.1 Issues of the Model of the Inner Controlled Loop

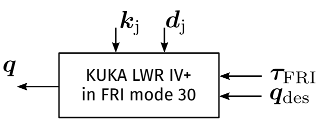
As spelled out in Chap. 4, a linear model based on the perturbed double integrator model is suitable to construct the parameterization and characterize admissible Q-parameters, taking the inner controlled manipulator loop as relevant plant dynamics. The control law (7.1) suggests that some form of AID control law is implemented in the joint-specific impedance mode of the robot. However, neither the precise dynamical model of the robot is publicly known nor which quality of approximative inverse dynamics control is eventually realized by the internal controller of the robot:

- Setting $\boldsymbol{k}_\mathrm{j}$ and $\boldsymbol{d}_\mathrm{j}$ to the lowest admissible values and using $\boldsymbol{q}_\mathrm{des} = \boldsymbol{q}_\mathrm{msr}$ results in
$$\boldsymbol{\tau}_\mathrm{cmd} \approx \boldsymbol{\tau}_\mathrm{FRI} + \boldsymbol{f}_\mathrm{dynamics}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}). \tag{7.4}$$
With $\boldsymbol{\tau}_\mathrm{FRI} = \boldsymbol{0}$, the robot works effectively in gravity compensation as verified experimentally [165].
- Referring to (4.6), ideally, the behavior of the robot would with $\boldsymbol{k}_\mathrm{j} \to 0$ and $\boldsymbol{d}_\mathrm{j} \to 0$ in the FRI controller (7.1) result in $\boldsymbol{\Delta}_\mathrm{M} \to 0$ and $\boldsymbol{\psi} \to 0$, allowing to realize the outer-loop input $\boldsymbol{u}$ determining the effective impedance via $\boldsymbol{\tau}_\mathrm{FRI} = \boldsymbol{u}$. However, both seemingly natural assumptions do not result in a good feedback linearization [165], *i. e.*, neither $\boldsymbol{\tau}_\mathrm{FRI} = \boldsymbol{u}$ nor $\boldsymbol{\tau}_\mathrm{FRI} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{u}$. Consequently, $\boldsymbol{\Delta}_\mathrm{M} \neq 0$ and $\boldsymbol{\psi} \neq 0$ substantially affect the control performance when working with (7.4).
- It is unspecified how the effective gain $\boldsymbol{D}(\boldsymbol{d}_\mathrm{j})$ is determined from the normalized damping parameter $\boldsymbol{d}_\mathrm{j} \in \mathbb{R}^7$.
- High values of the superimposed torques $\boldsymbol{\tau}_\mathrm{FRI}$ are effectively limited to a maximum value and the commanded torque $\boldsymbol{\tau}_\mathrm{cmd}$ is actually realized by another nested control loop of higher sampling frequency.

In summary, the superimposed torque quantity $\boldsymbol{\tau}_\mathrm{FRI}$ constitutes an interface with only limited control over the effective torque used in the KUKA LWR IV+ when realized via the FRI interface as (7.4); hence, the model $\ddot{\boldsymbol{q}} = \boldsymbol{u}$ corresponding to an ideally feedback linearized system does not constitute a reasonably accurate nominal model to build the Q-parameterization in order to determine the control signal for $\boldsymbol{\tau}_\mathrm{FRI}$.

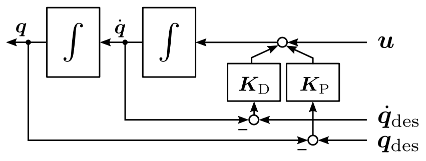
7.2.2 Closed-Loop Modeling

Due to the reasons above, it is not effective trying to circumvent the built-in PD gains of the FRI with the approach of (7.4) in order to set up a standard parameterization. The double integrator plant model obtained from Fig. 7.2b for $\boldsymbol{k}_\mathrm{j} \to 0$ and $\boldsymbol{d}_\mathrm{j} \to 0$ does not reasonably reflect the behavior of the robot hardware (Fig. 7.2a). However, what if these gains were



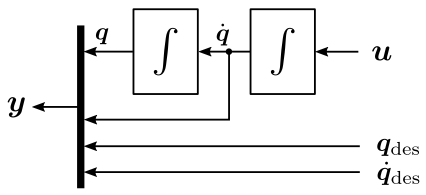
$\boldsymbol{k}_\mathrm{j}$
$\boldsymbol{d}_\mathrm{j}$
$\boldsymbol{q}$
KUKA LWR IV+ in FRI mode 30
$\boldsymbol{\tau}_\mathrm{FRI}$
$\boldsymbol{q}_\mathrm{des}$

(a) Relevant interface for joint impedance control



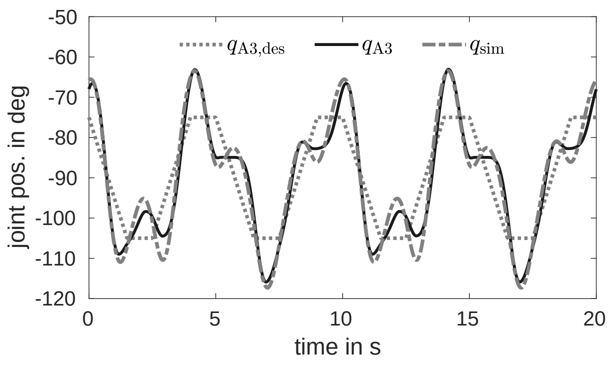
$\boldsymbol{q}$
$\dot{\boldsymbol{q}}$
$\boldsymbol{u}$
$\boldsymbol{K}_\mathrm{D}$
$\boldsymbol{K}_\mathrm{P}$
$\dot{\boldsymbol{q}}_\mathrm{des}$
$\boldsymbol{q}_\mathrm{des}$

(b) Conceptual model of the behavior when keeping the built-in gains active

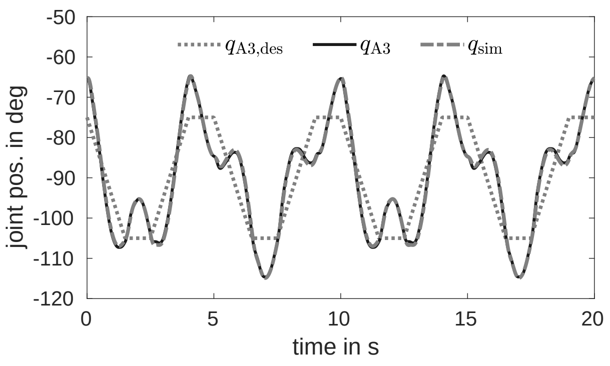


(c) Separating the model of the nominal (built-in) controller yields the corresponding plant

Figure 7.2: The model assumed to describe the relevant closed-loop behavior of the KUKA LWR IV+ robot axes when used in FRI mode 30



(a) Low gains in the pre-implemented control law (7.1) yield large deviations compared to the simulated model



(b) Medium gains in (7.1) yield better accuracy of the closed-loop model of Fig. 7.2b

Figure 7.3: Comparison of the simulation of the dynamic model shown in Fig. 7.2b to measurements on the KUKA LWR IV+, for low and medium gains. For the lower gains, depicted on the left side, the joint trajectory of the simulated model clearly deviates from the measurements. Using higher gains as depicted on the right side leads to a better fit between the simulated and measured trajectories.

not turned off but considered as nominal controller when building the parameterization? In this case, (7.1) should make the manipulator behave like a double integrator under PD state feedback; otherwise, the control law (7.1) would not accomplish its purpose of realizing a joint-impedance controller with stiffness and damping characteristics mutable via $\boldsymbol{k}_\mathrm{j}$ and $\boldsymbol{d}_\mathrm{j}$. In other words, we assume that the model depicted in Fig. 7.2b reflects the behavior of the manipulator Fig. 7.2a more accurately if the gains are active, *i. e.*, $\boldsymbol{k}_\mathrm{j} \not\to 0$ and $\boldsymbol{d}_\mathrm{j} \not\to 0$.

In order to check this assumption, an experiment using joint A3 (*cf.* Fig. 7.1a) was conducted [194], using low and medium internal control gains, as well as the superimposed torque feature. The reference trajectory $q_\mathrm{A3,des}$ is chosen as a trapezoidal motion with amplitude of 15 deg around the reference angle of -90 deg and a period time of 5 s. In the first setting, the internal gains were set to $k_\mathrm{A3} = 20, d_\mathrm{A3} = 0.1$ with the superimposed torque τ_FRI chosen as sinusoidal wave of amplitude 2 and frequency 0.5 Hz. In the second setting, we used $k_\mathrm{A3} = 300, d_\mathrm{A3} = 0.3$ with the superimposed torque sine of amplitude 50 and frequency 0.5 Hz. The model of the loop according to Fig. 7.2b was simulated with the same inputs, assuming $u = \tau_\mathrm{FRI}$ and $K_\mathrm{P} = k_\mathrm{A3}$ and $K_\mathrm{D} = 2 d_\mathrm{A3} \sqrt{k_\mathrm{A3}}$.

The results of this experiment are shown in Fig. 7.3. As can be seen, the trajectories resulting from the experiment (q_{A3}) and the simulation (q_{sim}) show much less deviation with the higher gains in (7.1), hence yielding more accuracy to the model. Taking a closed-loop approach therefore allows to construct a Q-parameterization on this robot although the built-in compensation is not precisely known, given that the parameterization allows to be built on top of the internal controller of the robot as exposed by the interface (7.1).

7.2.3 Implementation of the Central System

Plant model. In the previous section, the model for controller design shown in Fig. 7.2b was derived, while in the hardware experiment, the nominal controller is implemented directly by the interface of the robot depicted in Fig. 7.2a. Therefore, this discrepancy must be considered in the calculation of the Q-parameterization. To this end, we consider which plant resulted from Fig. 7.2b if the nominal control were implemented by the central system of the Q-parameterization. This gives rise to the generalized plant model of Fig. 7.2c and the nominal controller $\boldsymbol{D}_{\mathrm{K},0} = [-\boldsymbol{K}_{\mathrm{P}}, -\boldsymbol{K}_{\mathrm{D}}, \boldsymbol{K}_{\mathrm{P}}, \boldsymbol{K}_{\mathrm{D}}]$.

In the case study, it is essential to compute and implement the parameterization in discrete time for deployment to hardware. Taking the maximal frequency admissible by the FRI which is 1 kHz, the sampling rate is $T_{\mathrm{s}} = 1\,\mathrm{ms}$. Restricting attention to control of a single joint A3 of the robot as in the previous section, the model of the controlled channel is

$$G_{yu} : \left[\begin{array}{cc|c} 1 & 0.001 & 5 \cdot 10^{-7} \\ 0 & 1 & 0.001 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right]. \tag{7.5}$$

In this experiment, the nominal feedback gains are set in the FRI interface as $k_{\mathrm{A3}} = 150$, $d_{\mathrm{A}} = 0.2$, which is modeled by the nominal static feedback. Restricting attention to joint A3, the model of the controlled channel is

$$\boldsymbol{D}_{\mathrm{K},0} = \left[\begin{array}{cccc} -150.0 & -4.8990 & 150.0 & 4.8990 \end{array}\right] \tag{7.6}$$

for (7.5). Correspondingly, the closed-loop behavior of the robot is described by the feedback interconnection $(G_{yu}, \boldsymbol{D}_{\mathrm{K},0})$, resulting in

$$G_{\mathrm{cl,nom}} : \left[\begin{array}{cc|c} 0.9999 & 0.0009976 & 5 \cdot 10^{-7} \\ -0.15 & 0.9951 & 0.001 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \in \mathcal{RH}_{\infty}. \tag{7.7}$$

Using the parameterization of Prop. 4.2 with (4.36) yields $\boldsymbol{F}_{\mathrm{G}} = [-150 \quad -4.8990]$ and the design of the central system

$$J : \left[\begin{array}{cc|ccccc} 0.9999 & 0.0009976 & 0 & 0 & 0 & 0 & 5\cdot 10^{-7} \\ -0.15 & 0.9951 & 0 & 0 & 0 & 0 & 0.001 \\ \hline 0 & 0 & -150.0 & -4.8990 & 150.0 & 4.8990 & 1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{7.8}$$

Note that the dynamic model in J, simulating the controlled loop, precisely corresponds to the dynamics of the manipulator controlled by the pre-built compensation of the FRI mode 30, described by (7.7); this is due to the special coprime factorization chosen by $\boldsymbol{F}_{\mathrm{G}} = \boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2$ as discussed in Sec. 4.7.

Separation of the nominal controller. Note that the system (7.8) is only the design model used in the computations of the Q-parameters later. In the actual hardware implementation, the nominal static feedback is realized by the built-in controller of the FRI. Thus, due the structure of J, the controller of the FRI can be separated from the central system as shown in Fig. 4.6. The augmentation to construct the parameterization is then given by the system

$$J_{\mathrm{aug}} : \left[\begin{array}{cc|ccccc} 0.9999 & 0.0009976 & 0 & 0 & 0 & 0 & 5\cdot 10^{-7} \\ -0.15 & 0.9951 & 0 & 0 & 0 & 0 & 0.001 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{7.9}$$

Anti-windup compensation. In principle, at this point the parameterization would be ready to augment the controller in order to generate via Q commands τ_{FRI} that suitably modify the impedance characteristics of the manipulator. In practice, however, the limitations of the values of τ_{FRI} impose a non-negligible saturation of the superimposed torque. Therefore, we added an anti-windup scheme to the central system. Due to the pre-compensation, the controlled plant is stable and a model of the closed loop is obtained as shown in the previous section. Therefore, the full dynamic anti-windup scheme by Turner and Postlethwaite [241] can be utilized.

According to [241], the LMI

$$\begin{bmatrix} -\boldsymbol{Q}_{\mathrm{AW}} & -\boldsymbol{L}_{\mathrm{AW}}^{\top} & \boldsymbol{0} & \boldsymbol{Q}_{\mathrm{AW}}\boldsymbol{C}_2^{\top} + \boldsymbol{L}_{\mathrm{AW}}^{\top}\boldsymbol{D}_{22}^{\top} & \boldsymbol{Q}_{\mathrm{AW}}\boldsymbol{A} + \boldsymbol{L}_{\mathrm{AW}}^{\top}\boldsymbol{B}_2^{\top} \\ \star & -2\boldsymbol{U}_{\mathrm{AW}} & \boldsymbol{I} & \boldsymbol{U}_{\mathrm{AW}}\boldsymbol{D}_{22}^{\top} & \boldsymbol{U}_{\mathrm{AW}}\boldsymbol{B}_2^{\top} \\ \star & \star & -\mu_{\mathrm{AW}}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \star & \star & \star & -\boldsymbol{I} & \boldsymbol{0} \\ \star & \star & \star & \star & -\boldsymbol{Q}_{\mathrm{AW}} \end{bmatrix} \prec 0, \tag{7.10}$$

is solved, where $\boldsymbol{A}, \boldsymbol{B}_2, \boldsymbol{C}_2$ and $\boldsymbol{D}_{22}$ denote the state-space model of the plant dynamics, and the matrices $\boldsymbol{Q}_{\mathrm{AW}} \succ 0$, $\boldsymbol{U}_{\mathrm{AW}} = \operatorname{diag}(\mu_1, \ldots, \mu_m)$, $\boldsymbol{L}_{\mathrm{AW}} \in \mathbb{R}^{n \times m}$ and the scalar $\mu_{\mathrm{AW}} > 0$ are determined from the LMI. If it is solvable, a gain matrix $\boldsymbol{F}_{\mathrm{AW}}$ is given by $\boldsymbol{F}_{\mathrm{AW}} = \boldsymbol{L}_{\mathrm{AW}} \boldsymbol{Q}_{\mathrm{AW}}^{-1}$ and the anti-windup compensator is obtained from

$$\Theta_{\mathrm{AW,full}} : \left[\begin{array}{c|c} \boldsymbol{A} + \boldsymbol{B}_2 \boldsymbol{F}_{\mathrm{AW}} & \boldsymbol{B}_2 \\ \hline \boldsymbol{F}_{\mathrm{AW}} & \boldsymbol{0} \\ \boldsymbol{C}_2 + \boldsymbol{D}_{22} \boldsymbol{F}_{\mathrm{AW}} & \boldsymbol{D}_{22} \end{array}\right]. \tag{7.11}$$

Note that the closed-loop (pre-stabilized) model constitutes the relevant plant dynamics for anti-windup synthesis in this case study, *i. e.*, $\boldsymbol{A}, \boldsymbol{B}_2, \boldsymbol{C}_2$ and $\boldsymbol{D}_{22}$ are taken from (7.7). Using convex optimization [67], the LMI (7.10) is solved which results in the compensator

$$\Theta_{\mathrm{AW}} : \left[\begin{array}{cc|c} 0.9999 & 0.0003221 & 5 \cdot 10^{-7} \\ -0.1159 & -0.3557 & 0.001 \\ \hline 34.08 & -1351.0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right]. \tag{7.12}$$

Overall structure of parameterization. The structure of the overall parameterized control system is depicted in Fig. 7.4. It consists of the robot controlled via the FRI which implements the nominal gains as well as the interface to superimpose the control signal $\boldsymbol{u}$ which is in turn determined from the augmentation system (7.9) with anti-windup compensation (7.12). The model for the signal limitation of τ_{FRI} is chosen as $\tau_{\mathrm{FRI,A3}} = 95$ in order to accommodate for the saturation which experimentally occurred from $|\tau_{\mathrm{FRI,satA3}}| \gtrsim 100$.

7.3 Realization of Variable Impedance Filters

After having constructed a suitable parameterization based on an internal model of the closed loop, a Q-filter is designed to realize the desired impedance behaviors. To this end, the desired stiffness profile (7.3) and damping $d_{\mathrm{A3}} = 0.1$ is reformulated as a convex combination of the feedback gains $\boldsymbol{D}_{\mathrm{K},1} = [-130.0\ -2.280\ 130.0\ 2.280]$, $\boldsymbol{D}_{\mathrm{K},2} = [-30.00\ -1.095\ 30.00\ 1.0954]$ that represent the stiffness for $k_{\mathrm{A3}} = 130$ and $k_{\mathrm{A3}} = 30$, respectively. The corresponding interpolation signal is given by $\boldsymbol{\alpha}(t) = 0.5 \begin{bmatrix} 1 + \sin(2\pi \cdot 3t) \\ 1 - \sin(2\pi \cdot 3t) \end{bmatrix}$. The dynamic filters to achieve recovery of the stiffness/damping characteristics in the design points are given from (B.12). Following similar steps as in Sec. B.4 and noting that (7.5) is strictly proper, a state-space formula to realize the filters is obtained from (B.12) as

$$Q_i : \left[\begin{array}{c|c} \boldsymbol{A} + \boldsymbol{B}_2 \boldsymbol{D}_{\mathrm{K},i} \boldsymbol{C}_2 & \boldsymbol{B}_2 \left(\boldsymbol{D}_{\mathrm{K},0} - \boldsymbol{D}_{\mathrm{K},i}\right) \\ \hline \left(\boldsymbol{D}_{\mathrm{K},0} - \boldsymbol{D}_{\mathrm{K},i}\right) \boldsymbol{C}_2 & \boldsymbol{D}_{\mathrm{K},i} - \boldsymbol{D}_{\mathrm{K},0} \end{array}\right],$$

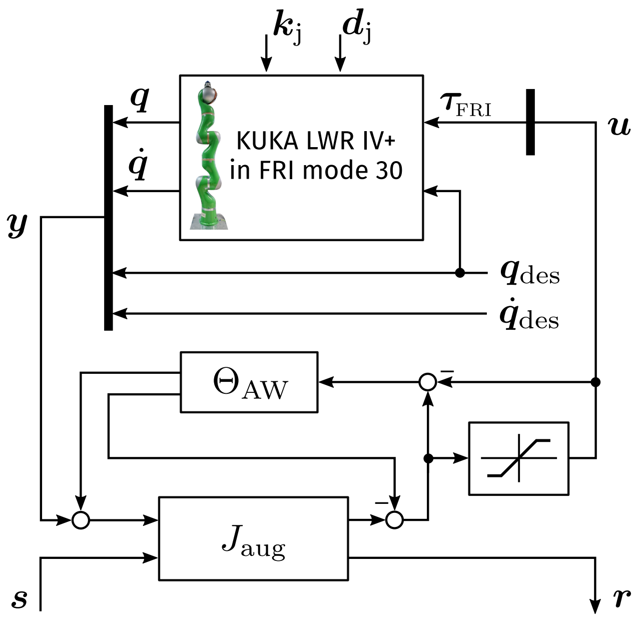


Figure 7.4: Structure to implement the central system of a Q-parameterization effectively on a KUKA LWR IV+. The numerical values used in the experiment on joint A3 are given by (7.9) and (7.12), with the limit in the saturation block set to 95 and $k_{A3} = 150$, $d_{A3} = 0.2$.

which reduces for the case of state feedback with $\boldsymbol{C}_2 = \boldsymbol{I}$ to (3.10). Numerically, for the considered joint this results in

$$Q_{\mathrm{tmp},1}: \left[\begin{array}{cc|cccc} 0.99994 & 0.0009989 & -0.00001 & -0.0000013093 & 0.00001 & 0.0000013093 \\ -0.13 & 0.99772 & -0.02 & -0.0026186 & 0.02 & 0.0026186 \\ \hline -20.0 & -2.6186 & 20.0 & 2.6186 & -20.0 & -2.6186 \end{array}\right],$$

$$Q_{\mathrm{tmp},2}: \left[\begin{array}{cc|cccc} 0.99998 & 0.00099945 & -0.00006 & -0.0000019018 & 0.00006 & 0.0000019018 \\ -0.03 & 0.99891 & -0.12 & -0.0038035 & 0.12 & 0.0038035 \\ \hline -120.0 & -3.8035 & 120.0 & 3.8035 & -120.0 & -3.8035 \end{array}\right].$$

These systems do not share a CQLF. Using the transformation of Sec. 3.1.5, we obtain

$$Q_1: \left[\begin{array}{cc|cccc} 0.99992 & 0.011381 & -0.0021684 & -0.00028392 & 0.0021684 & 0.00028392 \\ -0.011406 & 0.99774 & -0.30178 & -0.039513 & 0.30178 & 0.039513 \\ \hline -0.1163 & -0.17337 & 20.0 & 2.6186 & -20.0 & -2.6186 \end{array}\right],$$

$$Q_2: \left[\begin{array}{cc|cccc} 0.99995 & 0.0054699 & -0.023873 & -0.00075668 & 0.023873 & 0.00075668 \\ -0.005475 & 0.99894 & -2.6247 & -0.083193 & 2.6247 & 0.083193 \\ \hline -1.0012 & -0.16753 & 120.0 & 3.8035 & -120.0 & -3.8035 \end{array}\right]$$

which share $\boldsymbol{P}_{\mathrm{Q}} = \boldsymbol{I}$ as Lyapunov matrix.

In the experiments, we used the LQN architecture (3.11), for implementation of $Q(\boldsymbol{\alpha})$ to interpolate Q_1 and Q_2.

7.4 Results and Discussion

7.4.1 Experimental Results

The variable impedance behavior obtained [194] from the novel feedback interpolation scheme using the specific Q-parameterization is depicted in Fig. 7.5.

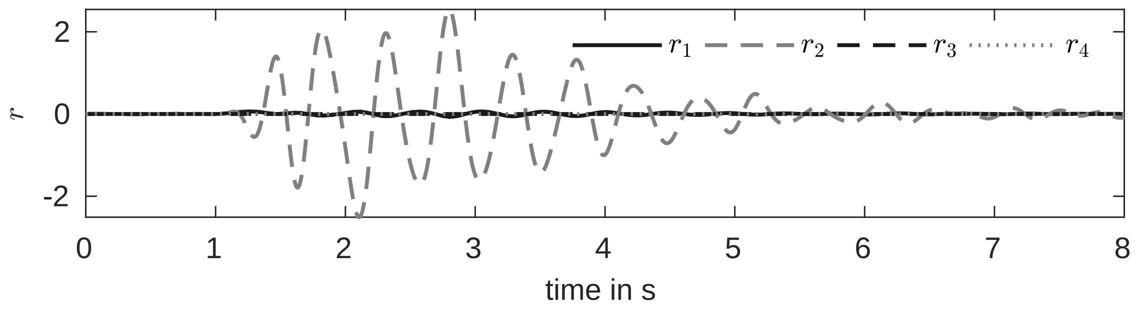
In Fig. 7.5a, the time evolution of the signal $\boldsymbol{r}$ is shown. Naturally, only the error signals concerning position and velocity of the joint corresponding to r_1 and r_2 excite the parameter system Q. There is no uncertainty associated with feedforward signals q_{des} and $\dot{q}_{\text{des}}$; hence, the corresponding entries r_3 and r_4 are zero. Figures 7.5b and 7.5c show the resulting output of the interpolated Q-parameter which is becoming the additional torque input τ_{FRI}. Note that, apart from the impulse to simulate a push on the joint, the control input u is equal to s which is due to the coprime factors chosen with (7.8). The trajectory of the joint position is shown in Fig. 7.5d. While the system movement exhibits some transient oscillations, the joint eventually returns to the desired position. The small oscillation remaining from 6 s on is resulting from the model uncertainty inherited from the black-box character of the controlled robot interface which is only inaccurately modeled by (7.7).

7.4.2 Discussion

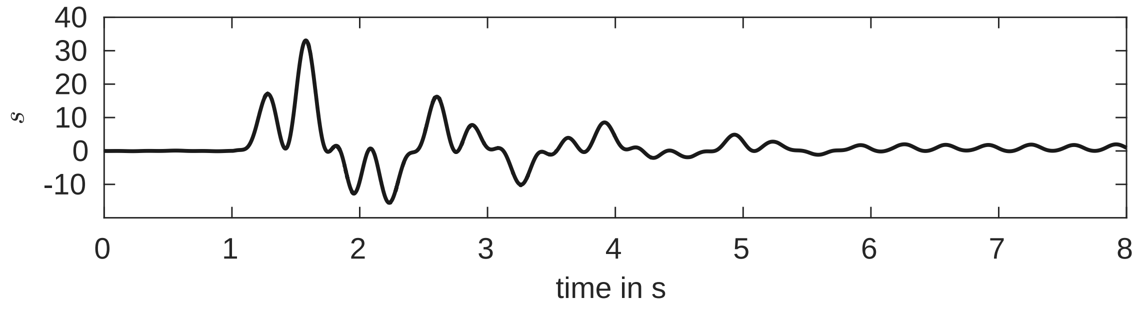
The case study presented in this chapter illustrates the superior performance achievable using a Q-parameterization compared to ad-hoc gain scheduling on the robotic platform. While the experiment confirmed the conditions derived in [123] for stability in the ad-hoc gain scheduling approach, the method based on the parameterization presented in this thesis does not impose such restrictions on admissible interpolation signals. Therefore, for learning of variable impedance control skills as in [32], the method constitutes an ideal starting point because the stability problem is separated from the learning problem for the desired task.

The following remarks are in order concerning different aspects of the approach.

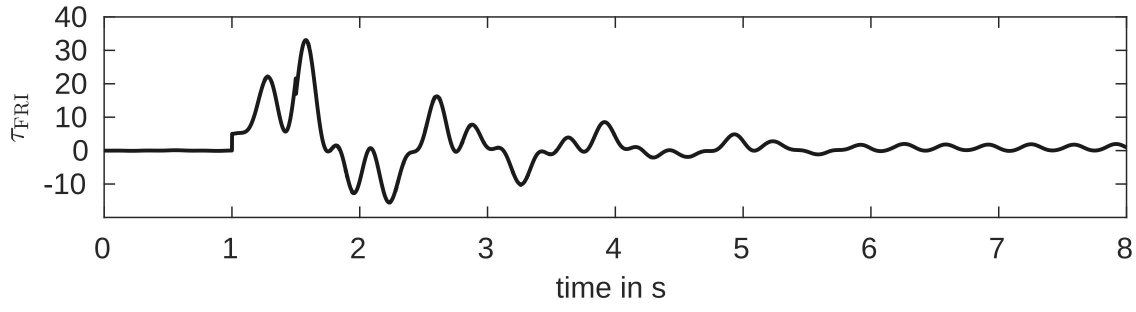
Choice of parameterization and model. The results of this case study could not be obtained with the standard observer-based interpretation of the Q-parameterization depicted in Fig. 2.2. This is because the basic PD behavior of the FRI controller interface of the KUKA LWR IV+ robot cannot be circumvented without deteriorating model quality as discussed above. The parameterization of Sec. 3.1 respectively Sec. 4.7 builds on the terminal connections (*cf.* requirement R2 in Sec. 3.1.2) and is therefore implementable on the robot. The key idea here is that usage of a model of the closed internal loop in the parameterization actually yields better accuracy; this property in turn can be exploited because the proposed parameterization admits a separation of the nominal controller from the central system. Therefore, the built-in controller of the robotic system did not have to be sidestepped but rather can be used as starting point. While this is the key advantage of the proposed parameterization on this robotic system in comparison to the observer-based Q-parameterization, also the comments of Sec. 3.1.7 are applicable. Apart from that, the experiment confirms our theoretical findings of Chap. 4 that the gains of the nominal robot controller



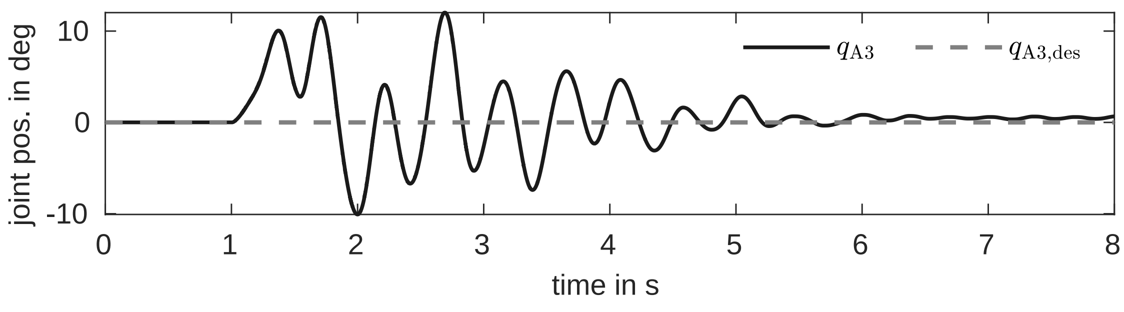
(a) Evolution of the signal $\boldsymbol{r}$ during the experiment.



(b) Output of interpolated $Q(\boldsymbol{\alpha})$.



(c) The superimposed torque τ_{FRI}, consisting of the simulated push as in Fig. 7.1b (rectangular impulse from $1\,\mathrm{s}$ to $1.5\,\mathrm{s}$) and the added torque u generated by the parameterization.



(d) Resulting joint motion

Figure 7.5: The implementation of the specific Q-parameterization over the FRI of the KUKA LWR IV+ robot provides experimental evidence for the results of the simulation of Fig. 3.3 shown in the theoretical illustrative study in Sec. 3.1.8.

substantially affect the closed-loop uncertainty and therefore constitute a valuable degree of freedom for the design of robot controllers with the parameterization approach.

Choice of numeric parameters. The desired impedance was chosen lightly damped ($d_{\mathrm{A3}} = 0.1$) in order to make the system exhibit instability under ad-hoc gain scheduling. For this reason, the initial gain used in the parameterization (7.6) could only be set to slightly higher damping because the gains of the controller parameters Q_1 and Q_2 for recovery of the basic impedance behaviors are becoming high otherwise, leading to a less robust overall system as discussed in Sec. 3.1.9. Clearly this case study has pushed at the boundary of system stability and in a practical robotic application, higher damping should be employed. Numerically, the poles are very close to 1 and it is emphasized again that it is crucial to compute the generator system J as well as the parameters Q_i with the state-space models obtained from symbolical simplification of the formulae involving the coprime factors. In our experience, attempting to compute the systems numerically directly from (2.8) and (B.12) leads to very badly conditioned matrices, prohibiting successful deployment to the physical system. Moreover, it is noted that a quasi-continuous approach, *i. e.*, computing the controllers in the continuous-time domain and subsequent discretization, in our experience does not deliver the desired performance on hardware. It is crucial to work upfront in the discrete-time domain for deployment of the control structure to digitally controlled systems.

Experimental limitations on the hardware system. While feasibility of the method was shown only on a single rotational joint in this chapter, additional experiments with the control structure were conducted which controlled up to five joints on hardware using the proposed parameterization [194]. The first five joints of the kinematic chain of the KUKA LWR IV+ were controlled successfully with the scheme. The remaining two joints A5 and A6 were subject to hardware limitations; *i. e.*, on the robot (Fig. C.1) used during the experiments, the friction compensation in the last two joints was too ineffective to allow even for the closed-loop modeling approach of Sec. 7.2.2. Other than that, the number of dynamical states and correspondingly the computational complexity resulting from the control structure for more than five joints limited the admissible sampling rate. With five joints or less, the maximal possible sampling frequency of 1 kHz could be used. Operating more joints with this frequency and the control structure, our setup ran into violations of the real-time requirements. However, this issue can be simply mitigated by choosing a slower control rate, admissible due to the discrete-time domain computation of the numerical values of the parameterization. Our experiments indeed confirmed that the control scheme works also effectively when operating the FRI with only 500 Hz.

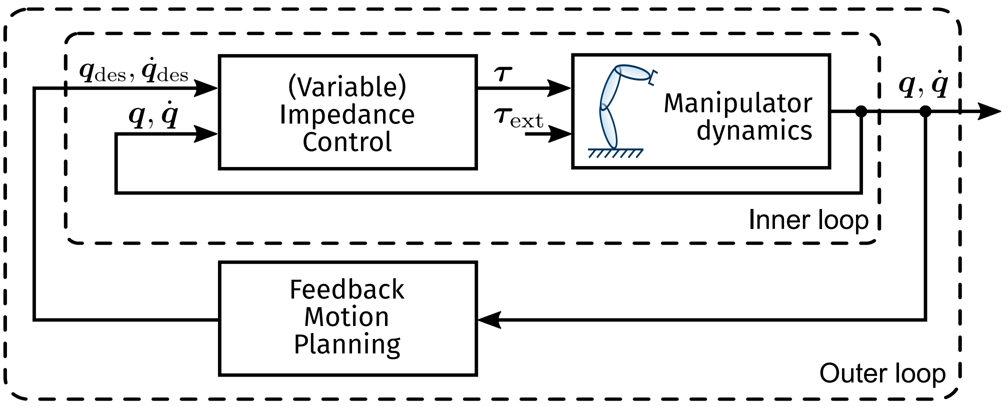
7.5 Conclusion

The problem of learning variable impedance control robot skills poses challenges from both control as well as learning point of views. The methods developed in this thesis contribute to this challenging field using an approach based on the parameterization of stabilizing controllers. Thereby, the control problem is structured such that it becomes admissible to learn gain schedules in straightforward fashion. While the method was developed theoretically in Chap. 3.1, the case study presented in this chapter demonstrates that the methods can be implemented on commercially relevant robotic hardware and that the experimental results confirm the theoretical advantages of the approach.

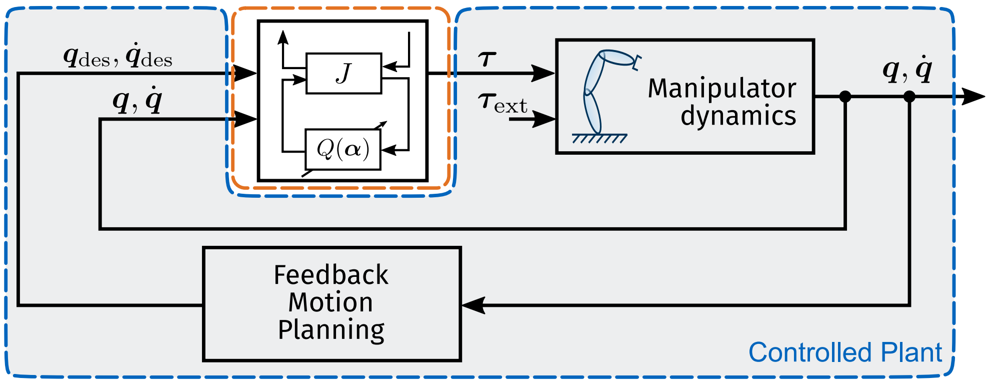
In this case study, attention was restricted to illustrate the stability properties, using a fixed reference position. The problem of learning variable impedance skills, however, in general requires to learn both motion and impedance characteristics of a robotic task. This is particularly challenging when motion is represented as a dynamical system, *e. g.*, using DMPs as in [32] or a GMM as in [107]. The nested control loops resulting in this case are depicted in Fig. 7.6a, where the impedance controller constitutes the inner controller with the dynamical system acting as feedback motion planning (FMP) wrapped around. As pointed out in [105], stability is hard to analyze in this setup due to the mutual coupling of the nested loops and variable gains.

While [105] proposes to avoid this setup completely by unifying FMP and impedance control into a single system to be learned [108], the novel architecture based on the Q-parameterization to implement variable impedance offers an alternative approach to deal with this problem. To this end, the key idea is to learn the dynamical system representing only the motion with established methods such as [32, 107] first and *subsequently consider this system as part of the controlled plant.* Consequently, due to the Q-parameterization, *the dynamical model and interconnection structure used for FMP automatically becomes an internal model of the (impedance) controller.* Due to the versatility of the Q-parameterization in the generalized plant setting, stability could then be guaranteed using the controller realization and interpolation scheme analogously as shown above as long as the FMP systems are (quadratically) stable. This approach is illustrated in Fig. 7.6b.

First steps in this direction were worked out in greater detail in [194], using a DMP as the dynamical system to generate the motion, an $\mathcal{H}_\infty$ approach [114] to design the impedance controller, and a Q-parameterization to implement the stability-preserving interpolation scheme on the KUKA LWR IV+ as discussed above. While the results of [194] are promising and the robot felt very intuitive during interaction, research towards a rigorous evaluation of the performance of the approach should be conducted in more extensive human-robot interaction studies. To this end, future work should also investigate how to realize variable impedance in task-space using the proposed scheme. Moreover, the extension to GMM dynamical feedback motion generators is a topic for future research, *e. g.*, exploiting LPV parameterizations such as [185].



(a) A widely adopted approach in robotics is to build the feedback motion planning block around the impedance control loop. This setup is vulnerable to instability because stability analyses conducted separately for the control loops are usually not sufficient to ensure stability of the overall system. While in practice, the setup often works well enough when impedances are fixed, clearly, implementing variable impedance control laws in the inner loop increases the danger of instability. See [105, 108] for a discussion of these issues.



(b) Implementing the variable impedance control architecture similarly as done in this case study allows to interpolate the impedance characteristics via Q, mitigating instabilities occurring otherwise due to hidden coupling effects (orange dashed box). Just as it is standard to add, for example, an integrator to the plant in order to model the stationary accuracy requirement, the dynamical systems representing the feedback motion planning may be considered as part of the plant under control as well (blue dashed box). Then, the Q-parameterization contains an internal model of the dynamics driving the robot manipulator, effectively circumventing mutual interference between FMP and variable impedance feedback control.

Figure 7.6: Sketch of a realization scheme for stable and decoupled variable impedance robot skills, using dynamical systems based feedback motion planning and the proposed novel controller parameterization approach.

8

Tracking Control Performance Enhancement

The previous chapter discussed how the novel parameterization for interpolated feedback control can be implemented and leveraged for variable impedance control on a commercial robotic platform. The second case study is oriented towards the classic goal of achieving high-precision trajectory tracking by means of episodic learning from trajectories. In this chapter, elements of all three technical areas considered in this thesis are employed to safely enhance the performance of the tracking controller with machine learning directly on hardware. These contributions were published previously in [57].

This case study illustrates the following aspects of the proposed approach.

- In order to implement controllers in terms of their Q-parameter, an internal model of the plant is needed in order to set up the parameterization. It was theoretically shown in Chap. 4 that a crude model can be sufficient in robotic manipulator control practice to create and use a Q-parameterization. This study serves to exemplify how a suitable model can be constructed based on limited prior knowledge without much modeling effort. It is demonstrated that the nominal model need not be accurate as long as the control loop is being internally stabilized by the initial controller.
- The parameterization was developed to extend a simple PD controller on joint level. Therefore, this study is representative for a control architecture frequently occurring in practice. As discussed in Sec. 4.9.1, the higher the gain of the nominal controller, the less accurate the dynamic process model required.
- In order to avoid obfuscation of the interplay of embedded RL in a Q-parameterization in the overall control strategy, we use a general-purpose, simple to implement, episodic learning approach in this study. In each episode, a trajectory is run with a fixed set of parameters; subsequently, the parameters are updated according to an observed numerical reward and the process iterates with the next episode. Such an approach to RL can be conceived of as black-box optimization [225]. Therefore, a black-box optimization algorithm is used as a straightforward way to implement the learning mechanism.
- Due to the beneficial stability properties of the Q-parameterization, the overall architecture allows to employ reinforcement learning on hardware from the start. This is in sharp contrast to the so-called mental rehearsal approaches [115, Sec. 6.1] to robot RL that run learning in simulation until convergence and only apply the final controller to hardware. The manipulator used in the experiment is shown in Fig. C.2.

The outline of this chapter is as follows. In Sec. 8.1, the problem of tracking control performance using RL and the Q-parameterization is stated. The design of the components of the learning control scheme are then presented in Sec. 8.2 before the overall architecture is discussed in Sec. 8.3. Then, in Sec. 8.4 the simulation study is explained before the results are in Sec. 8.5 compared to the data experimentally obtained on the hardware platform.

8.1 System Description and Problem Statement

We consider a robot manipulator described by the standard rigid-body model (4.1)-(4.2), repeated here for convenience,

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{8.1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the joint displacement vector, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the torque input vector, $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is the symmetric, positive definite inertia matrix, and the vector $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ summarizes other terms such as the Coriolis, centrifugal, gravity and friction forces.

Problem 8.1. Consider a robot manipulator (8.1) and an initial controller K_0 designed to achieve robust stability of the robot control loop. Augment the controller to a Q-parameterization of stabilizing controllers and enhance the closed-loop performance safely using a reinforcement learning algorithm, generalizing improvement to other trajectories. $\diamond$

8.2 Design of Parameterization, Filter System and Learning Approach

Robot dynamics nominal model. In order to derive a linear stabilizing controller, the inertia matrix $\boldsymbol{M}(\boldsymbol{q})$ is split into a constant diagonal matrix $\boldsymbol{M}_0 = \operatorname{diag}(\overline{\boldsymbol{m}})$ and the remaining part which depends on the joint coordinates:

$$\boldsymbol{M}(\boldsymbol{q}) = \boldsymbol{M}_0 + \Delta\boldsymbol{M}(\boldsymbol{q}).$$

The approximate mean values of the moments of inertia related to each joint are collected in the vector $\overline{\boldsymbol{m}}$, and the effect of the cross-coupling due to the components of $\Delta\boldsymbol{M}(\boldsymbol{q})$ can be absorbed into $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. Thus, the dynamics (8.1) can be written separately for each joint as

$$\overline{m}_i \ddot{q}(t) = \tau_i(t) - n_i(t), \quad \forall i = 1, \dots, n,$$

i. e., a joint can be considered as a double integrator with unknown input disturbance terms $n_i(t) = n_i(\boldsymbol{q}, \dot{\boldsymbol{q}})$. Defining the position/velocity state $\boldsymbol{x}_i \triangleq [q_i, \dot{q}_i]^\top$ and taking as input $u_i \triangleq \tau_i$, a state-space description for each joint is accordingly given as

$$\dot{\boldsymbol{x}}_i(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \boldsymbol{x}_i(t) + \begin{pmatrix} 0 \\ \frac{1}{\overline{m}_i} \end{pmatrix} u_i(t) + \begin{pmatrix} 0 \\ -\frac{1}{\overline{m}_i} \end{pmatrix} n_i(t). \tag{8.2}$$

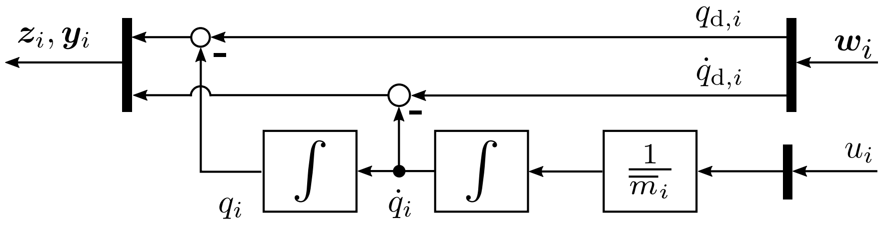


Figure 8.1: Crude model of nominal dynamics for i-th joint.

Considering the nonlinearities $n_i(t)$ as uncertainties and ensuring robust stability of the resulting closed loop, several decentralized, robust robot control strategies [211, 62, 87] rely on this model.

Generalized plant interconnection for tracking control. In order to construct the coprime factors (2.3) of the plant, the robot joint space dynamics description (8.2) is used, where the unknown terms $n_i(t)$ are considered as disturbance inputs acting on the nominal, linear model of the i-th joint. Considering the aim of motion control to track a desired trajectory $\boldsymbol{q}_\mathrm{d}$, for each joint we arrive at the generalized plant setup shown in Fig. 8.1, assuming output measurements and performance signals corresponding to both position and velocity errors. As the controller is digitally implemented, we directly proceed in the discrete time domain with a constant sampling time T_s. Discretizing the model (8.2) of each joint and parallel grouping yields the matrices of the generalized plant (A.4) for the complete n-link robot; relevant for subsequent feedback controller design are the matrices

$$\begin{aligned} &\boldsymbol{A} = \boldsymbol{I}_{n\times n} \otimes \begin{bmatrix} 1 & T_\mathrm{s} \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{B}_2 = \mathrm{blkdiag}\left(\left[0, \frac{T_\mathrm{s}}{m_1}\right]^\top, \ \dots, \ \left[0, \frac{T_\mathrm{s}}{m_n}\right]^\top \right), \\ &\boldsymbol{C}_2 = -\boldsymbol{I}_{2n\times 2n}, \quad \boldsymbol{D}_{22} = \boldsymbol{0}. \end{aligned} \tag{8.3}$$

Note that the plant is strictly proper, as there is no direct feedthrough from the control input torque to the measurements. Furthermore, $(\boldsymbol{A}, \boldsymbol{B}_2)$ and $(\boldsymbol{A}, \boldsymbol{C}_2)$ are stabilizable and detectable pairs, respectively.

Central system. In order to set up the Q-parameterization, we use the scheme based on a static feedback in order to build upon a simple PD controller $\boldsymbol{D}_{\mathrm{K},0}$, *i. e.*, the central system J is constructed according to Prop. 4.2. In discrete time, we have from (4.35)

$$J : \begin{cases} \boldsymbol{x}_\mathrm{J}(t+1) = & (\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F})\, \boldsymbol{x}_\mathrm{J}(t) + \boldsymbol{B}_2\boldsymbol{s}(t), \quad \boldsymbol{x}_{\mathrm{J},0} = \boldsymbol{0} \\ \boldsymbol{u}(t) = & (\boldsymbol{F} - \boldsymbol{D}_{\mathrm{K},0}\boldsymbol{C}_2)\, \boldsymbol{x}_\mathrm{J}(t) + \boldsymbol{D}_{\mathrm{K},0}\, \boldsymbol{y}(t) + \boldsymbol{s}(t), \\ \boldsymbol{r}(t) = & -\boldsymbol{C}_2\boldsymbol{x}_\mathrm{J}(t) + \boldsymbol{y}(t). \end{cases} \tag{8.4}$$

Selection of Q-filter system. The next design choice is how the system $Q \in \mathcal{RH}_\infty$ is parameterized. Here, a filter is used which stores N_fb past values of the signal $\boldsymbol{r} \in \mathbb{R}^{2n}$, and allows for subsequent weighting of this information depending on some signal $\boldsymbol{\nu} \in \mathcal{V} \subset \mathbb{R}^{n_\nu}$:

$$\boldsymbol{s}_\mathrm{fb}(t) = \Xi_1(\boldsymbol{\nu}(t))\, \boldsymbol{r}(t) + \Xi_2(\boldsymbol{\nu}(t))\, \boldsymbol{r}(t-1) + \ldots + \Xi_{N_\mathrm{fb}+1}(\boldsymbol{\nu}(t))\, \boldsymbol{r}(t - N_\mathrm{fb}).$$

In other words, the system Q is representing a FIR-like filter with varying coefficients. A feedforward control signal can also be included in the setup of Q,

$$\begin{aligned} \boldsymbol{s}_{\mathrm{ff}}(t) &= \Xi_{\mathrm{ff},1}(\boldsymbol{\nu}(t))\, \boldsymbol{w}(t) + \ldots + \Xi_{\mathrm{ff},N_{\mathrm{ff}}+1}(\boldsymbol{\nu}(t))\, \boldsymbol{w}(t-N_{\mathrm{ff}}), \\ \boldsymbol{s} &= \boldsymbol{s}_{\mathrm{fb}} + \boldsymbol{s}_{\mathrm{ff}} = [Q_{\mathrm{fb}} \quad Q_{\mathrm{ff}}] \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{w} \end{bmatrix}. \end{aligned} \tag{8.5}$$

In the resulting two-degree-of-freedom control scheme, the feedback and feedforward characteristics can be tuned independently, while preserving the stability characteristics of Thm. 2.1. Hence, we are in the convenient situation that $\boldsymbol{\nu}$ may contain time, reference signals or even states without stability concern as long as the coefficients are finite. The signal $\boldsymbol{\nu}$ is henceforth called *query signal* over the *query space* $\mathcal{V}$. Thus, in general the filter coefficient functions $\Xi_i : \mathbb{R}^{n_\nu} \mapsto \mathcal{B}_{\max}^{n_{\mathrm{u}} \times n_{\mathrm{y}}}(m)$, $i = 1, \ldots, N_{\mathrm{fb}}+1$, and $\Xi_{\mathrm{ff},i} : \mathbb{R}^{n_\nu} \mapsto \mathcal{B}_{\max}^{n_{\mathrm{u}} \times n_{\mathrm{w}}}(m)$, $i = 1, \ldots, N_{\mathrm{ff}}+1$ may encode information about time-varying or even systematic nonlinearities, where $\mathcal{B}_{\max}(m)$ denotes the set of matrices of appropriate size with elementwise m-bounded entries. Depending on the layout of the filter coefficient matrices Ξ, cross-coupling between variables could be introduced. However, in order to keep the presentation simple, in this study, we use a diagonal layout, *i. e.*, position and velocity errors will be fed back only to the corresponding joint. For the feedforward path, one channel per controlled joint will be used.

Consequently, it remains to represent and learn $n_{\mathrm{p}} = n\,(2(N_{\mathrm{fb}}+1) + (N_{\mathrm{ff}}+1))$ functions to determine the coefficients in the block-diagonal FIR structure, denoted $\boldsymbol{\xi} \in \mathbb{R}^{n_{\mathrm{p}}}$ subsequently. The coefficients depend on the query signal, *i. e.*, we have

$$\boldsymbol{\xi} = \left[\xi_1(\boldsymbol{\nu}(t)), \xi_2(\boldsymbol{\nu}(t)), \ldots, \xi_{n_{\mathrm{p}}}(\boldsymbol{\nu}(t))\right] \tag{8.6}$$

over the query space $\mathcal{V}$, which is a hyperbox

$$\mathcal{V} = [\min(\nu_1), \max(\nu_1)] \times \cdots \times [\min(\nu_{n_\nu}), \max(\nu_{n_\nu})].$$

In the following, each parameter function ξ_i, $i = 1, \ldots, n_{\mathrm{p}}$, is represented as a linearly parameterized approximator given a vector $\boldsymbol{\phi}_i$ of n_{b} basis functions which are multiplied by the weights $\boldsymbol{\theta}_i$

$$\xi_i(\boldsymbol{\nu}(t)) = \sum_{j=1}^{n_{\mathrm{b}}} \phi_{i,j}(\boldsymbol{\nu}(t))\theta_{i,j} = \boldsymbol{\phi}_i^\top(\boldsymbol{\nu}(t))\, \boldsymbol{\theta}_i. \tag{8.7}$$

In order to allow for local spreading of the information during learning, a set of radial basis functions (RBFs) is distributed on the query space $\mathcal{V}$, *i. e.*,

$$\phi_{i,j}(\boldsymbol{\nu}(t)) = \exp\left(-\frac{1}{2\sigma_j^2}(\boldsymbol{\nu}(t) - \boldsymbol{\gamma}_j)^\top(\boldsymbol{\nu}(t) - \boldsymbol{\gamma}_j)\right) \tag{8.8}$$

with $\boldsymbol{\gamma}_j \in \mathcal{V}$ representing the center point and $\sigma_j^2 > 0$ the variance of each of the RBFs.

In summary, with this setup, a parameter weight vector $\boldsymbol{\theta} \in \mathbb{R}^{n_{\mathrm{p}} \cdot n_{\mathrm{b}}}$ is characterizing the dynamic plug-in controller Q.

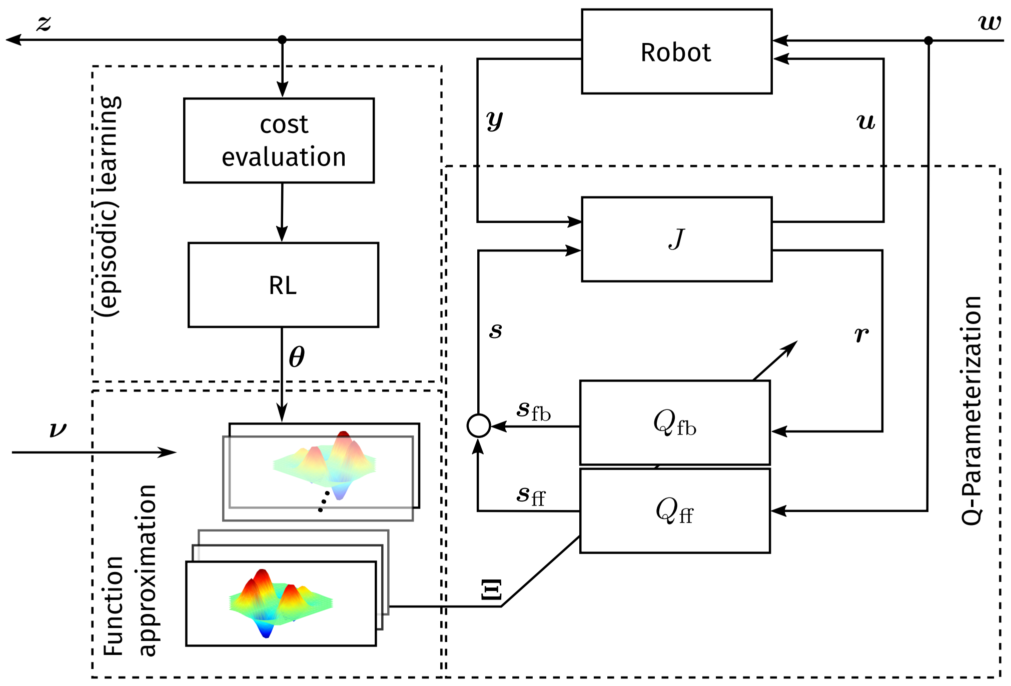


Figure 8.2: Overview of the control scheme employed in this case study: function approximation and RL techniques are combined with a Q-parameterization around a static controller and idealized robot dynamics, to build a stability-by-design learning architecture suitable for hardware implementation.

Learning nonlinear filter coefficients. For learning high precision motion control, a standard quadratic cost function is employed in this study, given by

$$C = \sum_{t=0}^{T} \boldsymbol{z}(t)^{\top} \boldsymbol{R}_z \boldsymbol{z}(t), \tag{8.9}$$

where $T \in \mathbb{N}$ denotes the number of time steps and $\boldsymbol{R}_z \succ 0$ is a suitable symmetric positive definite weighting matrix.

8.3 Discussion of the Overall Architecture

Combining the previous sections, we now discuss the overall control scheme graphically depicted in Fig. 8.2.

Effect of model simplification. An essential premise for applicability of the method is that the nominal controller $\boldsymbol{D}_{\text{K},0}$ is robustly stabilizing, otherwise the Q-parameterization built on the idealized robot dynamics will not be effective. Recall from Remark 2.1 that in an ideal Q-parameterization, the gain from $\boldsymbol{s}$ to $\boldsymbol{r}$ is zero; this cannot fully be expected here due to the inherent model mismatch given the nonlinear robot dynamics (8.1) and the idealized model (8.2) that serves as baseline for the parameterization. The neglected model mismatch causing a non-ideal gain from $\boldsymbol{s}$ to $\boldsymbol{r}$ can be quantified by the norm of a dual Youla operator S as discussed in Chap. 4. Such analyses were already exemplified in detail in Sec. 4.9. For the purpose of this case study, suffice to say, using a high gain linear PD controller yields robust stability to the nonlinear rigid body system as discussed in Sec. 4.9.1. As long as this is the case for the robot at hand, the parameterization is expected to work well in practice, which is confirmed by our experimental study reported in Sec. 8.5.

Type of plug-in filter Q. First, consider the case $Q = 0$, *i. e.*, no plug-in filter is active; consequently, $\boldsymbol{s} = 0, \delta\boldsymbol{u} = 0$ and in this case simply the original controller K_0 is recovered, the nominal PD controller.

Including a feedback $Q_{\mathrm{fb}} \neq 0$ allows to generate the additional control signal $\delta\boldsymbol{u}$ to enhance performance where the PD controller cannot perform well, *e. g.*, due to friction or nonlinearities. Note that the filter coefficients Ξ may well be varying with time or state; therefore, nonlinear disturbance effects can be suppressed without compromising the stability of the loop. This would not be the case if one were to add such a filter directly in parallel to the initial PD controller; it is a consequence due to the Q-parameterization.

Finally, with a stable feedforward $Q_{\mathrm{ff}} \neq 0$, the scheme may behave like a learning version of a “PD with feedforward compensation” controller oftentimes employed in robotics.

Learning and function approximation. The performance enhancement achievable by the plug-in compensation scheme naturally depends on the function approximation employed in the Q filter. While theoretically, the query signal $\boldsymbol{\nu}$ could contain any signal of interest, in practice one has to carefully select the query space $\mathcal{V}$, as the number of RBFs is exponential in $\dim\mathcal{V}$. In our simulation study, a typical choice is the desired trajectory $\boldsymbol{\nu} = \boldsymbol{q}_{\mathrm{d}}$. In order to keep the dimensionality low, another approach is to maintain the decentralized architecture of the PD controller also in the function approximation, *i. e.*, for each parameter function, a separate query space $\mathcal{V}_i$ could be used corresponding only to the desired state of each joint $i = 1, \dots, n$.

The amount of grid points $\boldsymbol{\gamma}$ is a compromise between good approximation accuracy and convergence speed during learning. Analogously, the widths σ_j of the RBFs determine the rate of decay around the grid points. Thus, they should be chosen such that both learning of local information as well as generalization of the learned coefficients to new trajectories is possible. Per dimension of the grid (8.7), we equally distribute n_γ RBFs with the width σ_j set to half of the grid spacing in order to achieve smooth spreading over the space $\mathcal{V}$.

In the next section, we clarify the general methodology proposed in this section by means of an example robotic system.

8.4 Evaluation in Simulation

Consider the two DoF horizontal planar elbow manipulator consisting of two revolute joint links, shown in Fig. C.2 and the corresponding model Fig. 8.3. The robot is mounted in a horizontal plane and the parameters of this manipulator are given in Tab. C.2.

In order to conduct a simulation study, a nonlinear model in the form (8.1) of the robot manipulator is used instead of the real robot hardware. The model was obtained by adaptive system identification [72] and captures effects such as position-dependent cross-coupling,

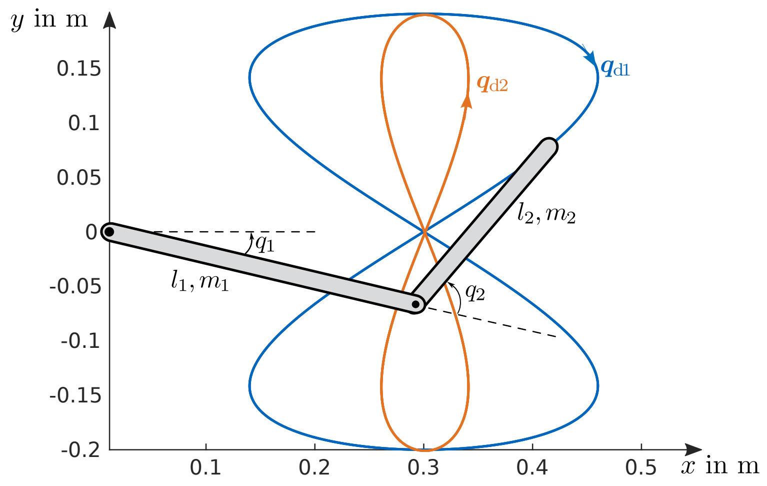


Figure 8.3: Coordinate system of the simulation and hardware experiments, top view. The initial joint position in all rollouts is $(0, \pi/2)$. The solid blue '8' is the trajectory $\boldsymbol{q}_{\mathrm{d}1}$ used for learning; the orange trajectory $\boldsymbol{q}_{\mathrm{d}2}$ will be used to show how the learned Q-filter generalizes the performance to unseen trajectories.

Coriolis and centrifugal forces. The inertia matrix $\boldsymbol{M}(\boldsymbol{q})$ and the vector $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ of such a manipulator are

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q}) &= \begin{bmatrix} m_{11} + \tilde{m}_{11}\cos q_2 & m_{12} + \tilde{m}_{12}\cos q_2 \\ m_{21} + \tilde{m}_{21}\cos q_2 & m_{22} \end{bmatrix}, \\ \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \begin{bmatrix} -n_{11}\dot{q}_2^2 \sin q_2 - n_{12}\dot{q}_1\dot{q}_2 \sin q_2 + f_{11}\dot{q}_1 \\ n_{21}\dot{q}_1^2 \sin q_2 + f_{22}\dot{q}_2 \end{bmatrix}. \end{aligned} \tag{8.10}$$

The numeric values of the robot model are given by (C.4) in Appendix C.4.

The task is to track an '8'-shaped path 4 times per roll-out, each taking a duration of $t_{\mathrm{period}} = 10\,\mathrm{s}$. The path is shown in Fig. 8.3. The performance of the nominal high-gain PD controller is shown in Fig. 8.4, showing deficiencies in the regions with high nonlinearities.

We proceed to build the Q-parameterization as described in Sec. 8.2. First, to obtain the idealized system dynamics, the mean values of the joint-specific inertia are obtained by neglecting the cosine and off-diagonal terms in the inertia matrix, yielding $\overline{m}_1 = m_{11} = 0.442$ and $\overline{m}_2 = m_{22} = 0.222$. Next, the state feedback gain $\boldsymbol{F}$ appearing in (8.4) that determines the specific coprime factors (2.16) is designed by solving the discrete-time algebraic Riccati equation (DARE) [21]

$$\boldsymbol{P} = \boldsymbol{A}^\top \left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}_2(\boldsymbol{B}_2^\top \boldsymbol{P}\boldsymbol{B}_2 + \boldsymbol{R})^{-1}\boldsymbol{B}_2^\top \boldsymbol{P}\right)\boldsymbol{A} + \boldsymbol{Q},$$

yielding a suitable

$$\boldsymbol{F} = -(\boldsymbol{B}_2^\top \boldsymbol{P}\boldsymbol{B}_2 + \boldsymbol{R})^{-1}\boldsymbol{B}_2^\top \boldsymbol{P}\boldsymbol{A}. \tag{8.11}$$

Using

$$\boldsymbol{Q} = \mathrm{diag}(1000, 1, 1000, 1), \quad \boldsymbol{R} = 10^{-4} \cdot \boldsymbol{I}_{4x4}$$

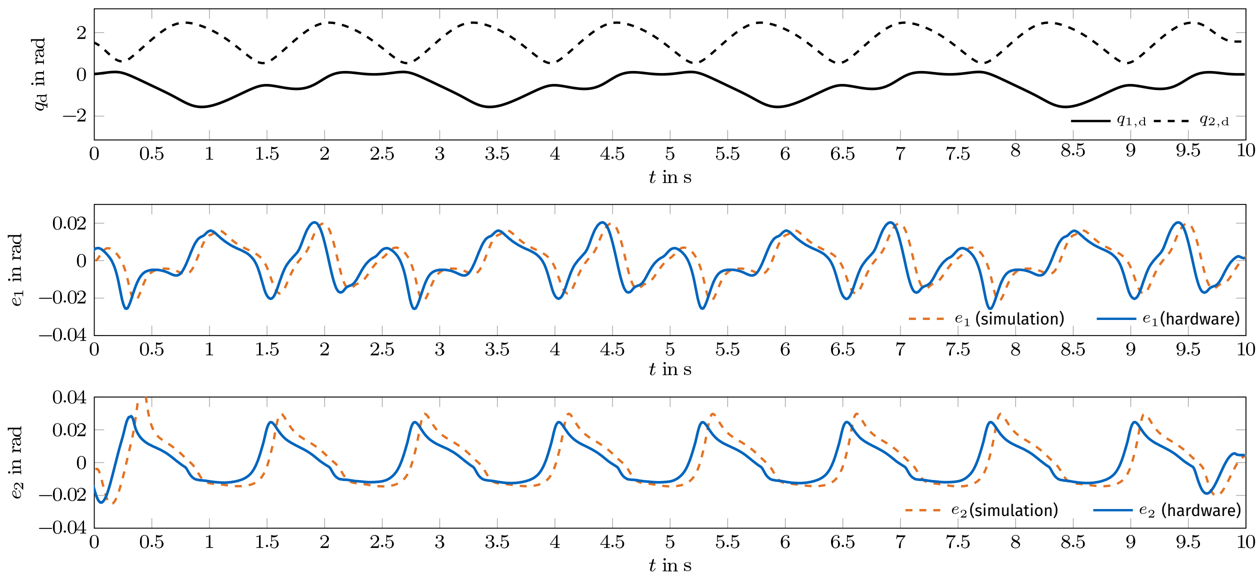


Figure 8.4: Top graph: the desired trajectory for both joints. Middle and bottom graphs: the corresponding tracking errors for both joints with the initial high-gain PD controller in simulation (orange, dashed) and measured in the hardware experiment (blue, solid).

one obtains

$$\boldsymbol{F} = \begin{bmatrix} -2772 & -101.7 & 0 & 0 \\ 0 & 0 & -2453 & -85.10 \end{bmatrix},$$

which was used in the following experiments. For practical implementation, we also added a saturation to the system J of Fig. 4.6 in order to model the actuator limits.

We now bring in a operator Q constructed as in (8.5), with a block-diagonal structure so that the decentralized architecture of the PD controller is kept by feeding back only entries of $\boldsymbol{r}$ corresponding to the respective joint, *i. e.*,

$$Q_{\text{fb}} = \begin{bmatrix} Q_{\text{P},1} & Q_{\text{D},1} & 0 & 0 \\ 0 & 0 & Q_{\text{P},2} & Q_{\text{D},2} \end{bmatrix}.$$

Taking $\boldsymbol{w} = [\boldsymbol{q}_\text{d}, \dot{\boldsymbol{q}}_\text{d}, \ddot{\boldsymbol{q}}_\text{d}]^\top$, also a feedforward compensation part involving the desired acceleration is added as

$$Q_{\text{ff}} = \begin{bmatrix} \boldsymbol{0}_{1\times 4} & Q_{\text{C},1} & 0 \\ \boldsymbol{0}_{1\times 4} & 0 & Q_{\text{C},2} \end{bmatrix}.$$

Each channel is represented as a sixth order FIR filter in the feedback and a simple gain on the feedforward path, *i. e.*, $N_{\text{fb}} = 6$ and $N_{\text{ff}} = 0$, respectively. This choice is motivated on the one hand from a practical point of view: the feedforward trajectory is designed to be smooth, contrary to the noisy feedback signal $\boldsymbol{r}$. Therefore, a higher order filter reduces hardware wear by generating a smoother additional control input $\delta\boldsymbol{u}$. On the other hand, from a theoretical point of view, higher order FIR filters in Q yield a RITZ approximation to cover the space of stabilizing controllers, *cf.* [28].

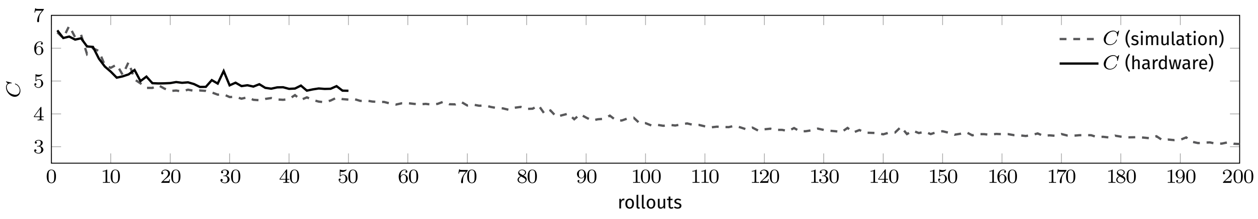


Figure 8.5: Cost evolution of the simulation and hardware experiment (finished after 50 rollouts).

Consequently, we learn a parameter function vector $\boldsymbol{\xi}$ with $n_\mathrm{p} = 2 \cdot (2 \cdot 7 + 1) = 30$ components to improve the tracking behavior. As query signal we select the desired velocity of joint 1 and the position of joint 2, $\boldsymbol{\nu} = [\dot{q}_{\mathrm{d},1},\ q_{\mathrm{d},2}]$, because these two quantities have the strongest effect on the neglected terms in (8.10). Finally, a compromise between performance and learning rate for $n_\gamma = 3$ RBFs per dimension is reasonable— using more RBFs could be done to achieve higher accuracy; however, the resulting slower convergence rate may become impractical on hardware. Summarizing, we have $n_\gamma^2 \cdot n_\mathrm{p} = 270$ entries in the weight vector $\boldsymbol{\theta}$ to be learned.

We used the state-of-the-art black-box optimization algorithm ROCK* [91] in order to learn the parameters $\boldsymbol{\theta}$ of the plug-in filter. All parameters were left unchanged w. r. t. their default values [90], apart from the initial covariance matrices of the weights $\boldsymbol{\theta}$. We used $\Sigma_\mathrm{fb} = 5 \cdot \mathbf{1}_{2n(N_\mathrm{fb}+1)\times 2^{n_\gamma}}$ for the feedback and $\Sigma_\mathrm{ff} = 5 \cdot 10^{-2} \cdot \mathbf{1}_{n(N_\mathrm{ff}+1)\times 2^{n_\gamma}}$ for the feedforward weights, respectively. Thus, the learning algorithm employs higher values on the feedback gain than on the feedforward channel; note that the magnitude of the input values is much higher on the feedforward path (desired acceleration) than of the residuals in the feedback path. In order to judge the performance of a single roll-out, *i. e.*, tracking 4 periods of the '8' track, (8.9) is evaluated with $T = t_\mathrm{period}/T_\mathrm{s}$ and $\boldsymbol{R}_z = \mathrm{diag}(4, 0, 1, 0)$, corresponding to a squared weighted ℓ_2-norm of the tracking error. In simulation, 200 roll-outs were performed, starting from $Q = 0$. By optimizing the FIR parameter function weights, the tracking error is gradually reduced as shown in Fig. 8.5. The typical landscape of a learned FIR parameter function is visualized in Fig. 8.6.

8.5 Experimental Validation

We implemented the method to control the robot shown in Fig. C.2, running compiled code on a real-time capable Linux kernel. While direct online adaptation of the PD gains often results in undesired trajectories after some roll-outs (*i. e.*, the first joint violates its angular limits imposed by hardware wiring), the proposed methodology did not yield any unstable trajectory roll-out throughout the hardware experiment.

The residual signal $\boldsymbol{r}$ generated from the parameterization while running one roll-out is shown in Fig. 8.7. It can be seen that the signal interconnection of Fig. 4.6 yields an estimate of the position and velocity errors which is nearly independent of the additional signal $\boldsymbol{s} \neq 0$. Thus, the experiment confirms the explanations of Sec. 8.3 that the Q-parameterization is appropriate, resulting in a small S although the model (8.3) is imprecise. This is achievable due to the nominal diagonal inertia matrix model in combination with the high gain and robustness of the nominal controller, *cf.* Sec. 4.9.1 for a detailed discussion of this matter.

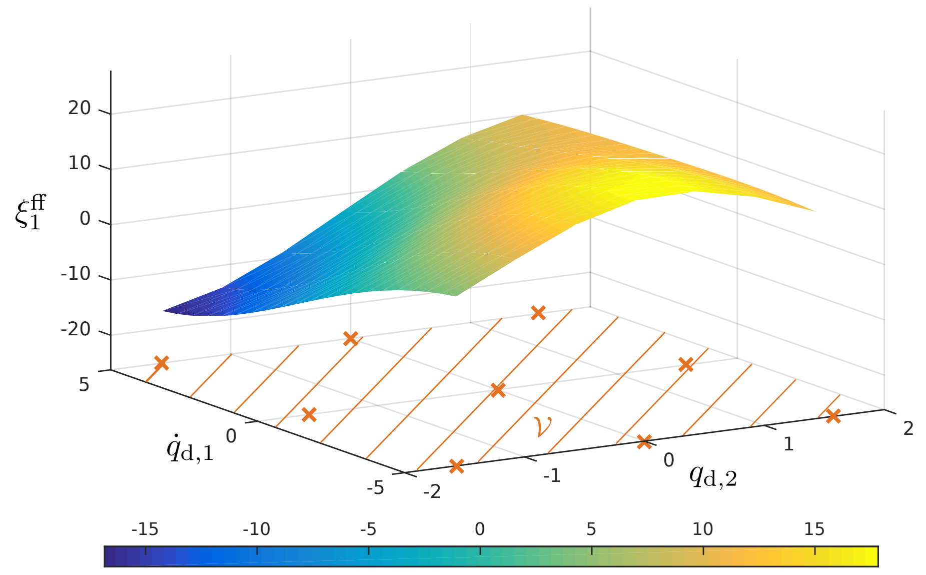


Figure 8.6: Landscape of a typical FIR filter coefficient function. Depicted is the coefficient function $\xi_{\mathrm{ff},1}$ on the first feedforward channel after 50 rollouts. The orange crosses show the centers γ_i of the RBFs on the space $\mathcal{V}$.

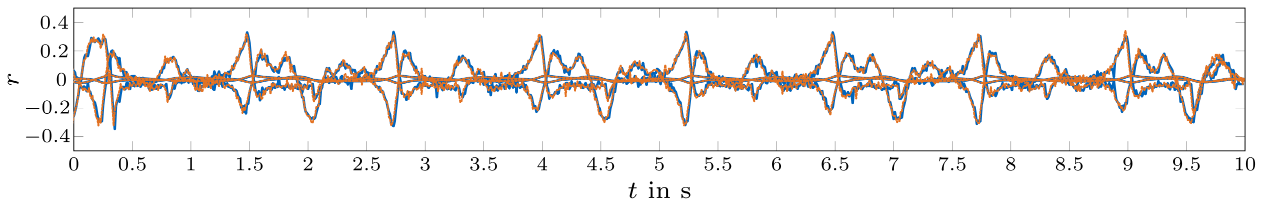


Figure 8.7: Evolution of $\boldsymbol{r}$ over the time of one roll-out, measured on hardware. Blue, solid: first roll-out, *i. e.*, $\boldsymbol{s} = 0, \delta\boldsymbol{u} = 0$. Orange, dashed: last roll-out, $\boldsymbol{s} \neq 0, \delta\boldsymbol{u} \neq 0$. The residual signal is hardly affected by the additional input $\boldsymbol{s}$, indicating that the Q-parameterization works with $S \to 0$.

The additional control signal $\boldsymbol{\delta u}$ recorded during roll-out 50 and the corresponding tracking error are reported in Fig. 8.8. As indicated by the evolution of the cost over the rollouts shown in Fig. 8.5, the tracking errors could be reduced compared to the initial roll-out. Though the experiment finished on the robot after 50 rollouts, the simulation study suggests that the error could be further reduced by running more iterations (Fig. 8.5).

In a last experiment, we tested the capability of the learned filter to improve other trajectories. Thus, the filter which was obtained by episodic learning with the wider trajectory ($\boldsymbol{q}_{\mathrm{d1}}$ in Fig. 8.3) was used while running a more narrow figure in a faster pace ($\boldsymbol{q}_{\mathrm{d2}}$ in Fig. 8.3, with 5 periods per roll-out) and in reverse direction. The results are summarized in Tab. 8.1. It can be seen that the performance is also enhanced for this test trajectory compared to the initial controller. This is due to the filter parameterization with coefficient functions that depend with $\boldsymbol{\nu} = [\dot{q}_{\mathrm{d},1},\ q_{\mathrm{d},2}]$ on the desired states of the system.

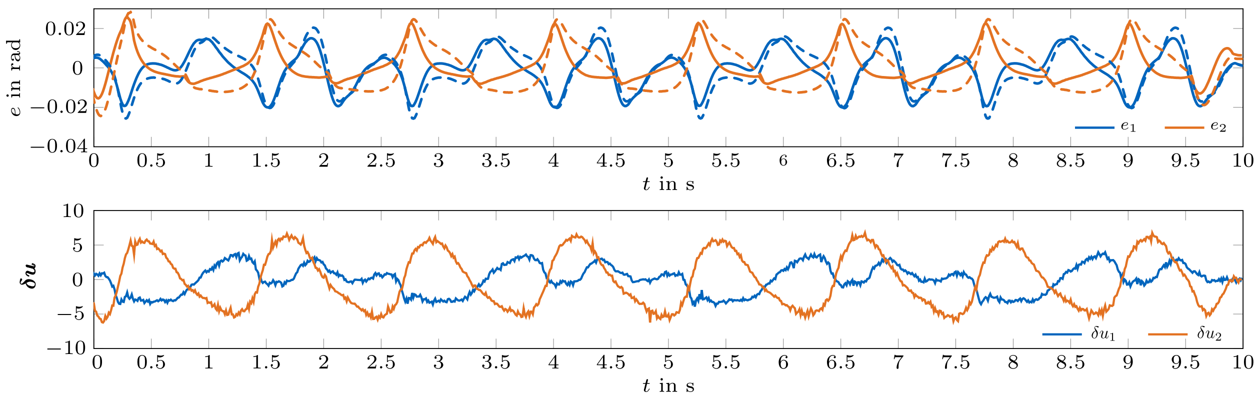


Figure 8.8: Top graph: tracking behavior of joint 1 (blue) and joint 2 (orange) in the hardware experiment after 50 rollouts (solid) compared to initial error (dashed). Bottom graph: the corresponding additive signal $\boldsymbol{\delta u}$ generated by the parameterization constructed as in Fig. 4.6.

8.6 Conclusion

In this case study, the reinforcement learning-Q control method was exemplified by means of a robotic tracking control experiment: the episodic black-box optimization of functional parameters in a two-degree-of-freedom Q-parameterization of stabilizing controllers constitutes an instance of the architectures $\boxed{3}$ and $\boxed{5}$ of Fig. 6.1 presented in Sec. 6.1.3. The experiment was conducted both in simulation and on hardware and demonstrates the stability of the closed loop during each roll-out as well as throughout all rollouts. This is due to the fact that the controller parameterization used in the actor-only learning approach constitutes, as depicted in Fig. 6.2, a specific actor structure tailored to the robot based on the domain model of nominal dynamics for each joint depicted in Fig. 8.1.

Table 8.1: Generalization capability of learned Q-filter. The results are from the hardware experiment, $Q = 0$ is the initial controller and Q with $\boldsymbol{\theta}_{\text{best}}$ denotes the augmented controller using the best parameters obtained after 50 rollouts. The filter was learned using only trajectory $\boldsymbol{q}_{\text{d1}}$, but the corresponding controller also improves the tracking behavior on other trajectories.

Trajectory (**Fig. 8.3**)	**Cost** (8.9) $Q = 0$	**Cost** (8.9) Q **with** $\boldsymbol{\theta}_{\text{best}}$
Wide ‘8’ $\boldsymbol{q}_{\text{d1}}$	6.33	4.59
Narrow ‘8’ $\boldsymbol{q}_{\text{d2}}$	1.89	1.08
Narrow ‘8’ $\boldsymbol{q}_{\text{d2}}$, reverse dir.	1.89	1.11

9

Conclusion and Future Directions

The aim of this thesis has been to investigate the parameterization of stabilizing controllers for the purpose of machine learning control, robotic feedback control, and finally the combination of both. Model-free, purely data-driven general reinforcement learning approaches do not ensure stability during the learning process in closed feedback loops yet. Therefore, this thesis has explores model-based results of control theory to suitably structure the search space for safe learning of feedback controllers, effectively incorporating prior domain knowledge. Particularly the control of robot manipulators has been deeply investigated as a field with a substantial amount of such domain knowledge. The significance of the contributions lies in the provision of a deterministic framework tailored for stability while allowing to employ machine learning online in the closed feedback loop. Feasibility was demonstrated not only in simulation but also in laboratory case studies involving physical robotic hardware.

9.1 Concluding Remarks

Part I is focused on model-based stabilizing controller parameterizations with particular attention to robotics. After reviewing in Chap. 2 the most common state-space interpretation in the form of an observer-based structure, a lesser-known realization is discussed that builds upon a static state feedback nominal controller. Next, in Chap. 3 it is shown that such a parameterization is advantageous for implementation in robotic manipulator control use cases, where interpretability and recovery of PD-like controllers are central requirements, *e. g.*, for learning active variable impedance skills. We therefore presented a novel architecture that is simpler than previous structures but allows for arbitrary controller interpolation. Previous state-of-the-art robotics methods, in contrast, need to constrain admissible interpolations to ensure stability. Next, in the spirit of locally linear control, we investigated how adaptive-Q extends to switching systems, exploring the synergy between adaptation and hybrid control that exists due to the parameterization. In Chap. 4, a new robot manipulator control framework was presented that assures stability to the loop in spite of online modifications of the feedback controller, allowing for a variety of offline and online learning control methods. Two distinct features of the approach are crucial. First, a clear separation of the types of uncertainty is carried throughout the method and second, the dual Youla operator is leveraged for the purpose of robotic manipulator control. Hence, we derived a new uncertainty measure to quantify the effect of approximations in the inverse dynamics in the interplay with the outer-loop controller. The resulting double-Youla parameterization for manipulator control confirmed intuition that the search space for learning over the Q-parameter is the more restricted the less accurate the model and the more uncertain the nominal loop is. The framework is applicable to a wide range of approximate inverse dynamics configurations, two-degree-of-freedom (feedforward/feedback) controller design, and numerous methods for the design of the Q-parameter with the admissible set $\mathcal{Q}$ systematically tightened to contain only robustly stabilizing controllers for robot manipulators.

In part II of the thesis, the realm of model-based control was intermittently left in Chap. 5 to develop a least-squares policy iteration algorithm that operates online, over continuous state and action spaces, with automatic handling of the value function approximation. The motivation to contribute progress to this class of algorithms was primarily fueled by the overarching aim of the thesis in mind of combining reinforcement learning with both robotics as well as the parameterization approach. Therefore, the general interplay of RL algorithms with the model-based parameterization is spelled out in the subsequent Chap. 6. Clearly distinguishing controller architecture including feedforward/feedback signals, classes of algorithmic approach to RL, and amount of uncertainty in the prior dynamical process model ultimately guides our way to construct the overall combination effectively. We propose to name the approach *Reinforcement Learning-Q Control*. It may be interpreted as a specific actor-only or actor-critic method for the purpose of granting stability to RL in a feedback loop based on domain knowledge in the form of prior dynamical models of process and controllers.

Part III finally presents laboratory case studies to demonstrate the suitability of the approach for deployment and effectiveness on physical robotic platforms. The problem considered in Chap. 7 is to implement active variable impedance control without the need of interpolation restrictions. Our experiments reveal that the danger of instability due to neglected hidden coupling may indeed occur on hardware, an issue mostly neglected by state-of-the-art robotic learning literature. Using the custom-tailored parameterizations derived in part I of the thesis, we were able to implement stable interpolation upon the fast research interface on a KUKA LWR IV+. The control method therefore allows to overcome a significant challenge in the design of learning algorithms for variable impedance skills. Precisely, the need to restrict the stiffness schedules in order to ensure stable execution of the skill in the reproduction phase on the robot is overcome. The experiments on this platform also confirmed the theoretical findings earlier in the thesis. Higher nominal gains tend to robustify the closed loop and alleviate the need of accurate dynamical robot models for implementation of the parameterization, such that the approach could eventually be implemented on a black-box kind of robot. Chapter 8 finally presents a case study to exemplify the overall method. A parameterization is obtained upon intentionally simple-minded but nevertheless useful domain modeling of the robot manipulator dynamics. Performance was enhanced by episodic learning from rollouts conducted directly on hardware. The experiment confirms the efficacy of the framework in terms of ensuring stability while allowing to generalize learned performance enhancement to other trajectories, which is the ultimate goal of any such robotic learning method.

To sum up, our approach bridges techniques and perspectives from both classical feedback control theory with current machine learning techniques. The robustness of the nominal controller is a prerequisite for successful employment of the methods on the specific robot manipulators used in the laboratory studies. The methods developed in this thesis might therefore be considered as a learning control framework in spirit similar to the philosophy of combined robust and adaptive control. For example, Lavretsky and Wise "[...] have found that it is not robust versus adaptive but rather a combination of both controllers that works best, in the sense of maintaining closed-loop stability, enforcing robustness to uncertainties, and delivering the desired performance [...]." [129, p. 4]

9.2 Recommendations for Future Research

The topic investigated in this thesis is strongly interdisciplinary and establishes a basis for exciting extensions towards each of the adjacent research areas. Both the fields of stable machine learning control as well as of robotic learning control are open for plenty more contributions stemming from the stabilizing parameterizations approach. While detailed pointers towards possible future work are given in the corresponding chapters, further research directions are listed below.

Controller parameterization. Particularly the double parameterizations over polytopic LPV dynamics are very promising in the context of machine learning and robotics. Firstly, a large number of regression algorithms construct locally linear models [224] and secondly, LPV systems can be used to model the nonlinear robot dynamics. Apart from that, recent results reported in emerging systems level synthesis [9] allow to assess the performance degradation of a robust controller due to uncertainties and to treat the input-output transfer matrices of the closed loop directly as design parameters [63]. This might alleviate the need of an initial stabilizing controller and therefore seems very exciting to explore with machine learning.

Parameterizations and robotics. The parameterizations developed in this thesis constitute a promising way to realize variable impedance feedback motion planning skills and further research is required starting from the approach outlined in Sec. 7.5. In addition, future work should explore how knowledge of the dynamic manipulator model can serve to construct a nonlinear parameterization with suitable extensions [4].

Learning of closed-loop uncertainty. In this thesis, uncertainty is considered via the dual parameter S estimated from prior process knowledge. Future research could aim to learn about the closed-loop uncertainty statistics in order to refine the admissible controller set iteratively, in spirit similar to the windsurfer approach to adaptive control [130]. Extensions towards using spatially refined local dynamic models and uncertainty information, supplied by machine learning algorithms, is a very promising open direction.

Learning-Q with adaptive kernel identification. In our laboratory case study of Chap. 8, the achievable performance enhancement was limited with a fixed set of basis functions combined with episodic learning. Meshfree function approximation techniques combined with online learning in the parameterization would constitute a more flexible architecture. Beyond that, adaptive kernel learning methods [139] are a very exciting way to implement the Q-filter. It should be possible to leverage the learning and generalization capabilities of these powerful signal processing algorithms in the parameterization; *i. e.*, given some history of state, estimation residual and performance quantity we can predict from previous filtered values a suitable next input to improve the control performance. A number of challenging research questions need to be answered. How to select the algorithmic hyper parameters, including the update rates, the initial signal values and the time embedding necessary for the control problem at hand? Is it required to design special kernels [178] to generalize performance to unseen trajectories and encode stability in the filter? How to ensure for robust stability in the uncertain loop a gain bound over kernel adaptive filters?

Part IV

Appendices

A

Background on Dynamical Systems and Coprime Factorization

This appendix summarizes some basic necessary backgrounds on dynamical systems, stability concepts, and coprime factorizations which are used throughout the monograph, based mainly on the textbooks [231, 269, 217]; for more complete treatments, see additionally *e. g.* [103, 10, 81]. While the notation is to some extent similar to that employed in [231], we mostly adopt the convention of [269] to draw the flow of signals in system diagrams from right to left in order to account for consistency with the corresponding matrix manipulations. Moreover, in the formulae that apply to both the continuous- and discrete-time domains, the symbol δ represents the derivative operator $\delta\boldsymbol{x}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t)$ in continuous-time and the one-step shift operator $\delta\boldsymbol{x}(t) = \boldsymbol{x}(t+1)$ in the discrete-time setting. Accordingly, stability of a matrix $\boldsymbol{A}$ refers to a Hurwitz matrix $\boldsymbol{A}$ with strictly negative real part of all eigenvalues, *i. e.*, $\mathrm{Re}(\lambda_i) < 0, \boldsymbol{\lambda} = \mathrm{eig}(\boldsymbol{A})$ in continuous time, or a Schur matrix $\boldsymbol{A}$ with strictly negative spectral radius $\rho(\boldsymbol{A}) < 1$ in the discrete-time case, respectively. For the sake of readability, dependencies on time $t \in \mathcal{T}$ are omitted whenever both time (in)variance and the applicable time set $\mathcal{T}$ are implicitly clear from the context.

A.1 Dynamical Systems

Many physical systems can be represented by a nonlinear state-space model of the form

$$\begin{aligned} \delta\boldsymbol{x} &= \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) \end{aligned} \tag{A.1}$$

where the state vector is denoted $\boldsymbol{x} \in \mathbb{R}^n$, the initial state $\boldsymbol{x}_0 \in \mathbb{R}^n$, the measured signals are $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{y}}$, and $\boldsymbol{u} \in \mathbb{R}^{n_\mathrm{u}}$ denotes the control inputs. In physical systems, the nonlinear vector functions $\boldsymbol{f} : \mathbb{R}^{n+n_\mathrm{u}} \mapsto \mathbb{R}^n$ and $\boldsymbol{g} : \mathbb{R}^{n+n_\mathrm{u}} \mapsto \mathbb{R}^{n_\mathrm{y}}$ are often smooth and continuously differentiable.

Linearizing about a nominal value or trajectory or if the system (A.1) is linear, one has the form

$$\begin{aligned} \delta\boldsymbol{x} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}, \end{aligned} \tag{A.2}$$

where the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is referred to as *state matrix*, $\boldsymbol{B} \in \mathbb{R}^{n\times n_\mathrm{u}}$ is the *input matrix*, $\boldsymbol{C} \in \mathbb{R}^{n_\mathrm{y}\times n}$ is the *(measured) output matrix*, and $\boldsymbol{D} \in \mathbb{R}^{n_\mathrm{y}\times n_\mathrm{u}}$ is called *feedthrough matrix*. Depending on the context, these matrices may be constant or varying with time, yielding a linear time-invariant (LTI) or a linear time-varying (LTV) system, respectively.

To model the effect of a system in some environment it is operated in, it is useful to generalize the model further by introducing the vector of *exogenous inputs* $\boldsymbol{w} \in \mathbb{R}^{n_\mathrm{w}}$ (containing *e. g.* reference signals, disturbances, noise), and a *performance quantity* $\boldsymbol{z} \in \mathbb{R}^{n_\mathrm{z}}$ that is used to assess the performance of the system. The resulting *plant model* of a system G maps the generalized input $\mathrm{col}(\boldsymbol{w}, \boldsymbol{u})$ to the generalized output $\mathrm{col}(\boldsymbol{z}, \boldsymbol{y})$, *i. e.*, in operator notation

$$\begin{bmatrix}\boldsymbol{z}\\ \boldsymbol{y}\end{bmatrix} = \begin{bmatrix} G_{zw} & G_{zu}\\ G_{yw} & G_{yu}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}\\ \boldsymbol{u}\end{bmatrix}. \tag{A.3}$$

This setup is henceforth also referred to as *generalized plant*, where a state model is correspondingly given as

$$G : \begin{cases} \delta\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_1\boldsymbol{w} + \boldsymbol{B}_2\boldsymbol{u}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0,\\ \boldsymbol{z} = \boldsymbol{C}_1\boldsymbol{x} + \boldsymbol{D}_{11}\boldsymbol{w} + \boldsymbol{D}_{12}\boldsymbol{u},\\ \boldsymbol{y} = \boldsymbol{C}_2\boldsymbol{x} + \boldsymbol{D}_{21}\boldsymbol{w} + \boldsymbol{D}_{22}\boldsymbol{u}.\end{cases} \tag{A.4}$$

Analogously, the shorthand notation

$$G : \left[\begin{array}{c|cc} \boldsymbol{A} & \boldsymbol{B}_1 & \boldsymbol{B}_2\\ \hline \boldsymbol{C}_1 & \boldsymbol{D}_{11} & \boldsymbol{D}_{12}\\ \boldsymbol{C}_2 & \boldsymbol{D}_{21} & \boldsymbol{D}_{22}\end{array}\right] \tag{A.5}$$

is used to describe a state-space realization of a system G.

Remark A.1 *(Subscript convention in generalized plant realization and transfer operators).* In this thesis, the following convention is used. For transfer operators, subscripts indicate the quantities the operator is acting on, *e. g.*, G_{yu} for the mapping from control input $\boldsymbol{u}$ to measured outputs $\boldsymbol{y}$; for state-space realizations, the subscripts are numbers that denote the indices of the subblock of the corresponding matrix, *e. g.*, $\boldsymbol{B}_2$ for the controlled input matrix above. $\triangleleft$

Remark A.2 *(Obtaining a transfer function matrix from the generalized plant state-space model).* It occurs often that the nominal plant model, *i. e.*, the matrix transfer function for a linear system model, corresponds to the mapping between G_{yu} in the generalized plant (A.4). In this case, a realization of G_{yu} is obtained from (A.4) as $G_{yu} : \left[\begin{array}{c|c}\boldsymbol{A} & \boldsymbol{B}_2\\ \hline \boldsymbol{C}_2 & \boldsymbol{D}_{22}\end{array}\right]$. $\triangleleft$

Next, let us assume that there are no unstable modes in the system that are not controllable using the available control inputs, or not observable from the selected measured variables, or both.

Assumption A.1 *(Stabilizability, Detectability).* Unless stated differently, in this thesis, it is assumed that the system is output feedback stabilizable, *i. e.*, $(\boldsymbol{A}, \boldsymbol{B}_2)$ and $(\boldsymbol{A}, \boldsymbol{C}_2)$ are stabilizable and detectable pairs, respectively. Consequently, $\exists \boldsymbol{F} \in \mathbb{R}^{n_\mathrm{u}\times n}$ and $\exists \boldsymbol{L} \in \mathbb{R}^{n\times n_\mathrm{y}}$ such that $\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A} + \boldsymbol{L}\boldsymbol{C}_2$ are stable. $\diamond$

In practice, lack of stabilizability or detectability of a system indicates that more actuators respectively sensors may be needed or that a different set of quantities should be measured.

Polytopic and switched linear systems. Some classes of nonlinear systems are particularly relevant in this thesis. Consider the *linear parameter-varying (LPV)* system [212] described by the state-space equations

$$G_{\mathrm{LPV}}: \begin{cases} \delta \boldsymbol{x} = \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y} = \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{x} + \boldsymbol{D}(\boldsymbol{\theta})\boldsymbol{u}, \end{cases} \tag{A.6}$$

where $\boldsymbol{\theta} : \mathcal{T} \mapsto \mathbb{R}^{N_{\mathrm{par}}}$ is an (unknown time-varying) bounded and piecewise continuous function with a finite number of discontinuities in any finite interval whose values $\boldsymbol{\theta}(t)$ belong to some set $\mathcal{P} \subseteq \mathbb{R}^{N_{\mathrm{par}}}$. Analogously to (A.5), the shorthand notation

$$G_{\mathrm{LPV}}: \left[\begin{array}{c|c} \boldsymbol{A}(\boldsymbol{\theta}) & \boldsymbol{B}(\boldsymbol{\theta}) \\ \hline \boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{D}(\boldsymbol{\theta}) \end{array}\right] \tag{A.7}$$

is meant to refer to the system (A.6).

Suppose that for $\boldsymbol{\theta} \in \mathcal{P}$, any matrix $\begin{bmatrix} \boldsymbol{A}(\boldsymbol{\theta}) & \boldsymbol{B}(\boldsymbol{\theta}) \\ \boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{D}(\boldsymbol{\theta}) \end{bmatrix}$ can be written as the convex combination of N_{sys} matrices $\begin{bmatrix} \boldsymbol{A}_i & \boldsymbol{B}_i \\ \boldsymbol{C}_i & \boldsymbol{D}_i \end{bmatrix}$, $i = 1, \ldots, N_{\mathrm{sys}}$. Then for any $\boldsymbol{\theta} \in \mathcal{P}$ there exists $\alpha_i \geq 0$ with $\sum_{i=1}^{N_{\mathrm{sys}}} \alpha_i = 1$ such that

$$\begin{bmatrix} \boldsymbol{A}(\boldsymbol{\theta}) & \boldsymbol{B}(\boldsymbol{\theta}) \\ \boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{D}(\boldsymbol{\theta}) \end{bmatrix} = \sum_{i=1}^{N_{\mathrm{sys}}} \alpha_i \begin{bmatrix} \boldsymbol{A}_i & \boldsymbol{B}_i \\ \boldsymbol{C}_i & \boldsymbol{D}_i \end{bmatrix}.$$

In this case, the system is referred to as *polytopic* LPV system [29, 202], *i. e.*,

$$G_{\mathrm{polytopic}}: \mathrm{conv}\left\{ \left[\begin{array}{c|c} \boldsymbol{A}(\alpha_1) & \boldsymbol{B}(\alpha_1) \\ \hline \boldsymbol{C}(\alpha_1) & \boldsymbol{D}(\alpha_1) \end{array}\right], \ldots, \left[\begin{array}{c|c} \boldsymbol{A}(\alpha_{N_{\mathrm{sys}}}) & \boldsymbol{B}(\alpha_{N_{\mathrm{sys}}}) \\ \hline \boldsymbol{C}(\alpha_{N_{\mathrm{sys}}}) & \boldsymbol{D}(\alpha_{N_{\mathrm{sys}}}) \end{array}\right] \right\}. \tag{A.8}$$

Switched systems are a class of hybrid systems, characterized by a number of continuous dynamics switching at (isolated) discrete events [137]. A switched system is modeled as

$$\delta \boldsymbol{x} = \boldsymbol{f}_i(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad i \in \mathcal{I} = \{1, 2, ..., N_{\mathrm{sys}}\}, \tag{A.9}$$

where $\mathcal{I}$ denotes a finite index set of N_{sys} discrete modes of the family of dynamics $\boldsymbol{f}_i$. In other words, the dynamics of the system can switch as governed by a switching signal $\sigma : \mathcal{T} \mapsto \mathcal{I}$ and the index $i = \sigma(t)$ is called *active mode* at time instant t. In the case where $\boldsymbol{f}_i$ is linear for all $i \in \mathcal{I}$, the system (A.9) is called *switched linear system.* Such a system may be seen as a special case in the LPV framework, taking a polytopic system (A.8) dependent on the scalar parameter $\theta = \sigma(t)$. Thus, in each time instant, the dynamics of the system are determined only by one of the vertices of the polytope, corresponding to the active mode.

A.2 Stability Concepts

The following notions of stability are used in this monograph.

Definition A.1 *(BIBO stability [231, p. 30]).* A system is called *BIBO stable* if any bounded input $\boldsymbol{u}$ yields a bounded output $\boldsymbol{y}$. $\diamond$
This is a special case for $p = \infty$ of the more general notion of finite-gain $\mathcal{L}$ stability defined next.

Definition A.2 *(Finite-gain $\mathcal{L}$ stability [103, Def. 5.1]).* A mapping $G : \mathcal{L}_p \mapsto \mathcal{L}_p$ is *finite-gain $\mathcal{L}$-stable* if there exist scalars $\gamma \geq 0$ and $\beta \geq 0$ such that

$$\|\boldsymbol{y}\|_{\mathcal{L}} \leq \gamma \|\boldsymbol{u}\|_{\mathcal{L}} + \beta \tag{A.10}$$

for all inputs $\boldsymbol{u} \in \mathcal{L}_p$ and corresponding output signals $\boldsymbol{y} \in \mathcal{L}_p$. The smallest number $\gamma \geq 0$ such that (A.10) is satisfied for the system G is called *$\mathcal{L}_p$ gain* of the system. $\diamond$

For $p = 2$, one obtains for γ the $\mathcal{L}_2$ gain of the system, corresponding to the root mean square (RMS) energy gain of the system; if the system is linear, the $\mathcal{L}_2$ gain is equivalent to the value of the $\mathcal{H}_\infty$ norm [103, Th. 5.4]. Using the ℓ_p signal norms over sequences, analogous definitions exist for the discrete-time case.

Next, consider the feedback interconnection of G and K depicted in Fig. A.1a.

Definition A.3 *(Well-posed feedback system [269, Ch. 5.2]).* The feedback system (G, K) is *well-posed* if all closed-loop transfer matrices are well-defined and proper. The interconnection in Fig. A.1a is well-posed iff $\boldsymbol{I} - K(\infty)G(\infty)$ is invertible, or equivalently, $\boldsymbol{I} - G(\infty)K(\infty)$ is invertible or $\begin{bmatrix} \boldsymbol{I} & -K(\infty) \\ -G(\infty) & \boldsymbol{I} \end{bmatrix}$ is invertible. $\diamond$

Most stability results using the parameterization of stabilizing controllers refer to the notion of internal stability as defined next.

Definition A.4 *(Internal stability [217, Def. 4.4], [231, p. 31]).* A system is *internally stable* if there are no hidden unstable modes in any of its components and any bounded input at any place in the closed-loop system results in bounded-output signals everywhere within the loop. $\diamond$

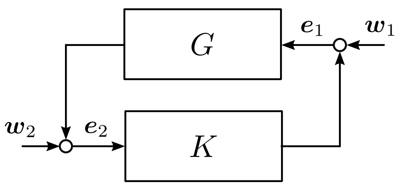
Proposition A.1 (Internal stability condition [231, Th. 2.3.1]). The feedback loop of Fig. A.1a is internally stable iff

$$\begin{bmatrix} I & -K \\ -G & I \end{bmatrix}^{-1} \in \mathcal{RH}_\infty \tag{A.11}$$

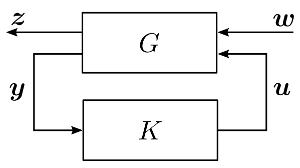
$$\Leftrightarrow \begin{bmatrix} (I - KG)^{-1} & K(I - GK)^{-1} \\ G(I - KG)^{-1} & (I - GK)^{-1} \end{bmatrix} \in \mathcal{RH}_\infty \tag{A.12}$$

If either holds, (G, K) is a *stabilizing pair.*

This proposition can be readily applied to the general feedback interconnection of Fig. A.1b.



(a) Basic feedback loop for internal stability analysis



(b) Closed-loop system with exogenous signals

Figure A.1: Basic plant/controller interconnections

Proposition A.2 (Internal stability condition for LFT setup [231, Th. 2.3.2]). The feedback loop of Fig. A.1b is internally stable iff

$$\begin{bmatrix} I & -\begin{bmatrix} 0 & 0 \\ 0 & K \end{bmatrix} \\ -G & I \end{bmatrix}^{-1} \in \mathcal{RH}_\infty. \tag{A.13}$$

Note that G is a partitioned operator here as in (A.3).

If two stable systems are interconnected by feedback, the *small-gain theorem* [48] is useful to establish stability.

Proposition A.3 (Small-gain [103, Ch. 5.4]). Let G_1 and G_2 be finite-gain $\mathcal{L}$ stable systems with induced system gains γ_1 and γ_2. Assume that the feedback system is well-defined, *i. e.*, there are by $(\boldsymbol{u}_1, \boldsymbol{y}_1)$ and $(\boldsymbol{u}_2, \boldsymbol{y}_2)$ given corresponding input/output pairs for G_1 and G_2, respectively. Then, the feedback interconnection formed by $\boldsymbol{e}_1 = \boldsymbol{u}_1 - \boldsymbol{y}_2$, $\boldsymbol{e}_2 = \boldsymbol{y}_1 + \boldsymbol{u}_2$ is finite-gain $\mathcal{L}$ stable if $\gamma_1 \cdot \gamma_2 < 1$.

If the operators G_1 and G_2 are linear, the following alternative formulation is often more useful and sets the foundation for many robust stability tests.

Proposition A.4 (Small-gain, alternative formulation [269, Th. 9.1]). Suppose $G_1 \in \mathcal{RH}_\infty$ and let $\gamma > 0$. The feedback loop is internally stable $\forall G_2 \in \mathcal{RH}_\infty : \|G_2\|_\infty \le \frac{1}{\gamma}$ iff $\|G_1\|_\infty < \gamma$, or equivalently, $\forall G_2 \in \mathcal{RH}_\infty : \|G_2\|_\infty < \frac{1}{\gamma}$ iff $\|G_1\|_\infty \le \gamma$.

Remark A.3. Note that Prop. A.3 gives a sufficient condition, whereas Prop. A.4 represents a necessary and sufficient condition. If G_2 were fixed with $\|G_2\|_\infty \le \frac{1}{\gamma}$, the small-gain theorem would usually be very conservative. However, as Prop. A.4 refers to all G_2 with $\|G_2\|_\infty \le \frac{1}{\gamma}$, the condition is also necessary. ◁

Next, consider the switched linear system

$$\delta \boldsymbol{x} = \boldsymbol{A}_{\sigma(t)} \boldsymbol{x} \tag{A.14}$$

and recall that stability of each of the matrices $\boldsymbol{A}_1, \dots, \boldsymbol{A}_{N_{\text{sys}}}$ is not sufficient to conclude stability of the overall system. For example, switching among two individually asymptotically stable systems can lead to an unstable switched system [137, Ch. 2]. In the study of switching systems, one usually distinguishes between the conditions to guarantee stability for arbitrary switching signals and conditions to constrain switching such that asymptotic stability of the switched system is obtained [137]. In this monograph, attention is restricted to the case of arbitrary switching signals. Consequently, it is necessary that all subsystems are stable, *i. e.*, for any fixed value $\sigma(t) = i$, the system $\delta \boldsymbol{x} = \boldsymbol{A}_i \boldsymbol{x}$ is stable. Moreover, the system (A.14) is called *switching stable* if it is asymptotically stable for any switching signal σ.

Similarly to switching, also interpolation of stable systems may lead to instability [212], for example in a polytopic system such as

$$\delta \boldsymbol{x} = \boldsymbol{A}(t) \boldsymbol{x}, \quad \boldsymbol{A}(t) \in \operatorname{conv}\Big\{ \boldsymbol{A}_1, \dots, \boldsymbol{A}_{N_{\text{sys}}} \Big\}. \tag{A.15}$$

However, a sufficient condition to ensure stability of (A.14) and (A.15) is the existence of a quadratic function $V(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{P} \boldsymbol{x}, \boldsymbol{P} \succ 0$ that decreases along every nonzero trajectory of the system. If such a matrix $\boldsymbol{P}$ exists, then the systems (A.15) and (A.14) are said to be *quadratically stable* and V is called *common quadratic Lyapunov function (CQLF).* Note that quadratic stability is a special class of exponential stability, which in turn implies asymptotic stability [103].

Proposition A.5 (LMI conditions for CQLF [29, 138]). Consider a family of $n \times n$ system matrices $\boldsymbol{A}_1, \dots, \boldsymbol{A}_{N_{\text{sys}}}$. There exists a positive definite symmetric matrix $\boldsymbol{P} \succ 0, \boldsymbol{P} \in \mathbb{R}^{n \times n}$ characterizing a CQLF $V(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{P} \boldsymbol{x}$ for the system (A.14) under arbitrary switching and for the polytopic system (A.15) iff the following linear matrix inequalities hold simultaneously

$$\text{(continuous-time)} \quad \boldsymbol{P}\boldsymbol{A}_i + \boldsymbol{A}_i^\top \boldsymbol{P} \prec 0, \quad \forall i = 1, \dots, N_{\text{sys}}. \tag{A.16a}$$

$$\text{(discrete-time)} \quad \boldsymbol{A}_i^\top \boldsymbol{P} \boldsymbol{A}_i - \boldsymbol{P} \prec 0, \quad \forall i = 1, \dots, N_{\text{sys}}. \tag{A.16b}$$

Some remarks concerning this compact summary are in order. First, note that quadratic stability is a restrictive property in that there exist switching linear plants that are asymptotically stable under arbitrary switching yet do not admit a CQLF, as shown for example in [137, Ch. 2.1.5] and [29, p. 73f]. However, the conditions (A.16) are conclusive to quadratic stability for both (A.14) and (A.15). Precisely, Lin and Antsaklis show “that the asymptotic stability problem for switched linear systems with arbitrary switching is equivalent to the robust asymptotic stability problem for polytopic uncertain linear time-variant systems [...]” [138] in both continuous and discrete time. Note that the stability of the polytopic LPV system (A.8) is determined from dynamics (A.15) with the vertex matrices $\boldsymbol{A}_1 = \boldsymbol{A}(\alpha_1), \dots, \boldsymbol{A}_{N_{\text{sys}}} = \boldsymbol{A}(\alpha_{N_{\text{sys}}})$. Therefore, confining to quadratic stability, the classes of arbitrarily switching linear systems and polytopic linear parameter-varying systems can be considered interchangeably in this thesis from a stability point of view.

Finally, a technical result is needed that establishes quadratic stability for system dynamics characterized by a block triangular form of quadratically stable matrices.

Lemma A.1 (Quadratic stability for block triangular form). For all $\boldsymbol{\theta} \in \mathcal{P}$, let $\boldsymbol{A}_{\boldsymbol{\theta},11}$ and $\boldsymbol{A}_{\boldsymbol{\theta},22}$ denote two quadratically stable matrices, each sharing a CQLF as defined in Prop. A.5 and let $\boldsymbol{A}_{\boldsymbol{\theta},12}$ denote a bounded matrix of appropriate size. Then the block matrix $\begin{bmatrix} \boldsymbol{A}_{\boldsymbol{\theta},11} & \boldsymbol{A}_{\boldsymbol{\theta},12} \\ \boldsymbol{0} & \boldsymbol{A}_{\boldsymbol{\theta},22} \end{bmatrix}$ is quadratically stable.

Proof. The lemma was presented in continuous-time in [24, Lemma A.1], based on a stricter version first presented in [256, Lemma 2], and in discrete-time in [17, Lemma 2]. □

A.3 Coprime Factorizations

For the derivation of stabilizing controller parameterizations, the mathematical framework of coprime factor descriptions are employed. For a comprehensive overview and detailed mathematical treatment of the *factorization approach*, the reader is referred to [246]. The following definitions are summarized from [269, 231].

Definition A.5 *(Right coprime over $\mathcal{RH}_\infty$ [269, Def. 5.3]).* Two matrices $M, N \in \mathcal{RH}_\infty$ are *right coprime over $\mathcal{RH}_\infty$* if they have the same number of columns and if the left inverse of the matrix $\begin{bmatrix} M \\ N \end{bmatrix}$ exists in $\mathcal{RH}_\infty$, *i. e.*, there exist matrices $X_\mathrm{r}, Y_\mathrm{r} \in \mathcal{RH}_\infty$ such that

$$[X_\mathrm{r} \quad Y_\mathrm{r}] \begin{bmatrix} M \\ N \end{bmatrix} = X_\mathrm{r} M + Y_\mathrm{r} N = I. \tag{A.17}$$

◇

Definition A.6 *(Left coprime over $\mathcal{RH}_\infty$ [269, Def. 5.3]).* Two matrices $\tilde{M}, \tilde{N} \in \mathcal{RH}_\infty$ are *left coprime over $\mathcal{RH}_\infty$* if they have the same number of rows and if the right inverse of the matrix $[\tilde{M} \quad \tilde{N}]$ exists in $\mathcal{RH}_\infty$, *i. e.*, there exist matrices $X_\mathrm{l}, Y_\mathrm{l} \in \mathcal{RH}_\infty$ such that

$$[\tilde{M} \quad \tilde{N}] \begin{bmatrix} X_\mathrm{l} \\ Y_\mathrm{l} \end{bmatrix} = \tilde{M} X_\mathrm{l} + \tilde{N} Y_\mathrm{l} = I. \tag{A.18}$$

◇

Equations of the form (A.17) are called BEZOUT *identity*[1] in this thesis. A more intuitive view on coprimeness of a factorization over $\mathcal{RH}_\infty$ is that, given the two factors are coprime, unstable pole/zero cancellations are excluded in the fractional representation of the transfer function defined as follows.

[1]As pointed out in [246], this naming is due to T. Kailath, while Kučera [125] coined the term DIOPHANTINE equation, and Vidyasagar [246] prefers to call it ARYABHATTA*'s identity.* Given this historical perspective found in [246, Preface], we adopt the convention predominant in the English literature, referring to it as Bezout identity.

Definition A.7 *(Coprime factorizations of G [269, Ch. 5.4]).* Let G be a proper real-rational matrix. A *right-coprime factorization* of G is a factorization $G = NM^{-1}$ where N and M are right-coprime over $\mathcal{RH}_\infty$, a *left-coprime factorization* in turn is given by $G = \tilde{M}^{-1}\tilde{N}$ such that $\tilde{N}$ and $\tilde{M}$ are left-coprime over $\mathcal{RH}_\infty$. If there exists a left-coprime factorization, a right-coprime factorization, and $X_\mathrm{r}, Y_\mathrm{r}, X_\mathrm{l}, Y_\mathrm{l} \in \mathcal{RH}_\infty$ such that

$$\begin{bmatrix} X_\mathrm{r} & Y_\mathrm{r} \\ -\tilde{N} & \tilde{M} \end{bmatrix} \begin{bmatrix} M & -Y_\mathrm{l} \\ N & X_\mathrm{l} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \tag{A.19}$$

the matrix G is said to have a *double coprime factorization.* ◇

The utility of using coprime factorizations is due to the following connection with internal stability.

Proposition A.6 (Internal stability of the feedback loop via coprime factorization [269, Lemma 5.10]). Consider the feedback loop of Fig. A.1a and let $G = NM^{-1} = \tilde{M}^{-1}\tilde{N}$ and $K = UV^{-1} = \tilde{V}^{-1}\tilde{U}$ be any right and left coprime factorizations of G and K with $N, M, \tilde{N}, \tilde{M}, U, V, \tilde{U}, \tilde{V} \in \mathcal{RH}_\infty$. Then, the following conditions are equivalent:

1. The feedback system is internally stable.
2. $\begin{bmatrix} M & U \\ N & V \end{bmatrix}^{-1} \in \mathcal{RH}_\infty$.
3. $\begin{bmatrix} \tilde{V} & -\tilde{U} \\ -\tilde{N} & \tilde{M} \end{bmatrix}^{-1} \in \mathcal{RH}_\infty$.
4. $\left(\tilde{M}V - \tilde{N}U\right)^{-1} \in \mathcal{RH}_\infty$.
5. $\left(\tilde{V}M - \tilde{U}N\right)^{-1} \in \mathcal{RH}_\infty$.

Note that conditions 1–5 can equivalently be expressed by the following *double Bezout equation.*

Proposition A.7 (Double Bezout equation [231, Lem. 2.4.1]). Under the conditions of Prop. A.6, (G, K) is a stabilizing pair iff

$$\begin{bmatrix} \tilde{V} & -\tilde{U} \\ -\tilde{N} & \tilde{M} \end{bmatrix} \begin{bmatrix} M & U \\ N & V \end{bmatrix} = \begin{bmatrix} M & U \\ N & V \end{bmatrix} \begin{bmatrix} \tilde{V} & -\tilde{U} \\ -\tilde{N} & \tilde{M} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{A.20}$$

In summary, the coprime factorization approach entails some advantages, three of which are particularly relevant for this thesis.

1. Due to Prop. A.6, it is relatively easy to impose the requirement of internal stability just by employing matrices over $\mathcal{RH}_\infty$ exclusively.
2. The methodology is applicable to both continuous- and discrete-time systems.
3. Coprime factors over $\mathcal{RH}_\infty$ can conveniently be constructed in state space, allowing to work with efficient symbolic and/or numeric procedures. These realizations are summarized next.

State-space realizations. A set of coprime factors that satisfy (2.3)–(2.4) can be obtained directly in state space by the formulae[2] given in [231, 234].

In order to obtain state-space realizations of the coprime factorizations of G from (A.5) and an arbitrary stabilizing linear dynamical controller $K_0 = \left[\begin{array}{c|c} \boldsymbol{A}_\mathrm{K} & \boldsymbol{B}_\mathrm{K} \\ \hline \boldsymbol{C}_\mathrm{K} & \boldsymbol{D}_\mathrm{K} \end{array}\right]$, construct state feedback gain matrices $\boldsymbol{F}$ and $\boldsymbol{F}_\mathrm{K}$ such that $\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A}_\mathrm{K} + \boldsymbol{B}_\mathrm{K}\boldsymbol{F}_\mathrm{K}$ are stable. Define $\boldsymbol{Y} \triangleq (\boldsymbol{I} - \boldsymbol{D}_\mathrm{K}\boldsymbol{D}_{22})^{-1}$ and $\boldsymbol{Z} \triangleq (\boldsymbol{I} - \boldsymbol{D}_{22}\boldsymbol{D}_\mathrm{K})^{-1}$. Coprime factors can then be obtained by

$$\begin{bmatrix} M_0 & U_0 \\ N_0 & V_0 \end{bmatrix} : \left[\begin{array}{cc|cc} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{0} & \boldsymbol{B}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_\mathrm{K}+\boldsymbol{B}_\mathrm{K}\boldsymbol{F}_\mathrm{K} & \boldsymbol{0} & \boldsymbol{B}_\mathrm{K} \\ \hline \boldsymbol{F} & \boldsymbol{C}_\mathrm{K}+\boldsymbol{D}_\mathrm{K}\boldsymbol{F}_\mathrm{K} & \boldsymbol{I} & \boldsymbol{D}_\mathrm{K} \\ \boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F} & \boldsymbol{F}_\mathrm{K} & \boldsymbol{D}_{22} & \boldsymbol{I} \end{array}\right], \tag{A.21a}$$

$$\begin{bmatrix} \tilde{V}_0 & -\tilde{U}_0 \\ -\tilde{N}_0 & \tilde{M}_0 \end{bmatrix} : \left[\begin{array}{cc|cc} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{Y}\boldsymbol{D}_\mathrm{K}\boldsymbol{C}_2 & \boldsymbol{B}_2\boldsymbol{Y}\boldsymbol{C}_\mathrm{K} & -\boldsymbol{B}_2\boldsymbol{Y} & \boldsymbol{B}_2\boldsymbol{Y}\boldsymbol{D}_\mathrm{K} \\ \boldsymbol{B}_\mathrm{K}\boldsymbol{Z}\boldsymbol{C}_2 & \boldsymbol{A}_\mathrm{K}+\boldsymbol{B}_\mathrm{K}\boldsymbol{Z}\boldsymbol{D}_{22}\boldsymbol{C}_\mathrm{K} & -\boldsymbol{B}_\mathrm{K}\boldsymbol{Z}\boldsymbol{D}_{22} & \boldsymbol{B}_\mathrm{K}\boldsymbol{Z} \\ \hline \boldsymbol{F}-\boldsymbol{Y}\boldsymbol{D}_\mathrm{K}\boldsymbol{C}_2 & -\boldsymbol{Y}\boldsymbol{C}_\mathrm{K} & \boldsymbol{Y} & -\boldsymbol{Y}\boldsymbol{D}_\mathrm{K} \\ \boldsymbol{Z}\boldsymbol{C}_2 & -(\boldsymbol{F}_\mathrm{K}-\boldsymbol{Z}\boldsymbol{D}_{22}\boldsymbol{C}_\mathrm{K}) & -\boldsymbol{Z}\boldsymbol{D}_{22} & \boldsymbol{Z} \end{array}\right]. \tag{A.21b}$$

[2]The formulas provided in [231, p. 40] are not entirely correct in the case when the controller is not strictly proper ($\boldsymbol{D}_\mathrm{K} \neq \boldsymbol{0}$), as one can verify by checking the double Bezout identity (2.4). It is required to add unary minus signs to the feedthrough matrix as in (A.21b), yielding the state-space realizations (A.21) also documented in the literature [234] and corresponding to the special case when the plant is strictly proper reported in [96].

B

Proofs

The following auxiliary results are introduced.

Lemma B.1. Let $\boldsymbol{v}_1 \in \mathbb{R}^{n_1}, \boldsymbol{v}_2 \in \mathbb{R}^{n_2}, \dots \boldsymbol{v}_m \in \mathbb{R}^{n_m}$ and define $\boldsymbol{v} = \left[\boldsymbol{v}_1^\top, \boldsymbol{v}_2^\top, \dots, \boldsymbol{v}_m^\top\right]^\top$. Then,

$$\|\boldsymbol{v}\| = \left\| \begin{pmatrix} \|\boldsymbol{v}_1\| \\ \|\boldsymbol{v}_2\| \\ \vdots \\ \|\boldsymbol{v}_m\| \end{pmatrix} \right\|.$$

Proof: See the proof of Lemma 2.10 in [269, p. 31]. ■

Lemma B.2. Let $a_1, a_2 \in \mathbb{R}$ and $\boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathbb{R}^n$, then

$$[a_1, a_2] \begin{bmatrix} \|\boldsymbol{h}_1\| \\ \|\boldsymbol{h}_2\| \end{bmatrix} \leq \|[a_1, a_2]\| \left\| \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \end{bmatrix} \right\|. \tag{B.1}$$

Proof: Follows with straightforward algebraic manipulations from the definition of the induced vector 2-norm. ■

B.1 Proof of Theorem 4.1

Proof: The state- and control-dependent nonlinear terms of $\boldsymbol{w}_\Delta^{\mathrm{U}}$ in (4.19) are collected separately in a vector $\boldsymbol{w}_\Delta = \mathrm{col}(\boldsymbol{w}_\Delta^{\mathrm{M}}, \boldsymbol{w}_\Delta^{\psi})$. We then have

$$\|\boldsymbol{w}_\Delta\| = \left\| \begin{bmatrix} \boldsymbol{w}_\Delta^{\mathrm{M}} \\ \boldsymbol{w}_\Delta^{\psi} \end{bmatrix} \right\| \leq \left\| \begin{bmatrix} \|\boldsymbol{\Delta}_{\mathrm{M}}\, \boldsymbol{u}\| \\ \|\boldsymbol{M}^{-1}\, \tilde{\boldsymbol{n}}\| \end{bmatrix} \right\| \leq \left\| \begin{bmatrix} \|\boldsymbol{\Delta}_{\mathrm{M}}\| \|\boldsymbol{u}\| \\ \|\boldsymbol{M}^{-1}\| \|\tilde{\boldsymbol{n}}\| \end{bmatrix} \right\|,$$

where the last inequality is due to the submultiplicative property of compatible induced matrix norms. Inserting the bounds (4.11) and (4.9a) yields

$$\|\boldsymbol{w}_\Delta\| \leq \left\| \begin{bmatrix} \|\boldsymbol{\Delta}_{\mathrm{M}}\| \|\boldsymbol{u}\| \\ M_{\mathrm{u}}\alpha_0 + M_{\mathrm{u}}\alpha_1 \|\boldsymbol{x}\| \end{bmatrix} \right\|.$$

At this point, one cannot simply treat the constant part of $\Phi(\|\boldsymbol{x}\|)$ as if it were a bounded additive disturbance $\boldsymbol{w}_{\text{ext}}$ such that $\|\boldsymbol{w}_{\text{ext}}\| \leq M_{\text{u}}\alpha_0$, similarly as it could be done with the external disturbances $\boldsymbol{w}_{\text{dist}}$. Here, such a step would void conservatism because $\|\text{col}(\|\boldsymbol{u}\|, a + \|\boldsymbol{x}\|)\| \not\leq \|\text{col}(\|\boldsymbol{u}\|, \|\boldsymbol{x}\|)\| + a$ in general. In order to proceed nonetheless, introduce a fictitious perturbation signal $z^{\text{f}}_{\Delta} \triangleq 1 = \|z^{\text{f}}_{\Delta}\|$ to rewrite the sum $M_{\text{u}}\alpha_0 + M_{\text{u}}\alpha_1\|\boldsymbol{x}\|$ as a dot product

$$\|\boldsymbol{w}_{\Delta}\| \leq \left\| \begin{bmatrix} \|\boldsymbol{\Delta}_{\text{M}}\|\|\boldsymbol{u}\| \\ [M_{\text{u}}\alpha_0,\ M_{\text{u}}\alpha_1] \cdot \begin{bmatrix} \|z^{\text{f}}_{\Delta}\| \\ \left\| \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \right\| \end{bmatrix} \end{bmatrix} \right\|.$$

By application of Lemma B.2, we have

$$\|\boldsymbol{w}_{\Delta}\| \leq \left\| \begin{bmatrix} \|\boldsymbol{\Delta}_{\text{M}}\|\|\boldsymbol{u}\| \\ \|[M_{\text{u}}\alpha_0,\ M_{\text{u}}\alpha_1]\| \cdot \left\| \begin{bmatrix} z^{\text{f}}_{\Delta} \\ \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \right\| \end{bmatrix} \right\| = \left\| \begin{bmatrix} \|\boldsymbol{\Delta}_{\text{M}}\| & 0 \\ 0 & \|[M_{\text{u}}\alpha_0,\ M_{\text{u}}\alpha_1]\| \end{bmatrix} \begin{bmatrix} \|\boldsymbol{u}\| \\ \left\| \begin{bmatrix} z^{\text{f}}_{\Delta} \\ \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \right\| \end{bmatrix} \right\|.$$

Due to the submultiplicative property of induced matrix norms, and by invoking Lemma B.1, the bound becomes

$$\|\boldsymbol{w}_{\Delta}\| \leq \left\| \begin{bmatrix} \|\boldsymbol{\Delta}_{\text{M}}\| & 0 \\ 0 & \|[M_{\text{u}}\alpha_0,\ M_{\text{u}}\alpha_1]\| \end{bmatrix} \right\| \left\| \begin{bmatrix} \|\boldsymbol{u}\| \\ \left\| \begin{bmatrix} z^{\text{f}}_{\Delta} \\ \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \right\| \end{bmatrix} \right\| \leq \left\| \begin{bmatrix} \alpha_{\text{M}} & 0 \\ 0 & \alpha_{\Psi} \end{bmatrix} \right\| \cdot \left\| \begin{bmatrix} \boldsymbol{u} \\ z^{\text{f}}_{\Delta} \\ \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \right\|, \tag{B.2}$$

where the abbreviations

$$\alpha_{\text{M}} \triangleq \|\boldsymbol{\Delta}_{\text{M}}\| \overset{(4.10)}{=} \alpha, \qquad \alpha_{\Psi} \triangleq \|[M_{\text{u}}\alpha_0,\ M_{\text{u}}\alpha_1]\| \tag{B.3}$$

are introduced. Denote by $\boldsymbol{z}_{\Delta} \triangleq \text{col}(\boldsymbol{u}, z^{\text{f}}_{\Delta}, \boldsymbol{q}, \dot{\boldsymbol{q}})$ the vector of signals exciting uncertainty. Thus, by the inequality (B.2), a conservative gain bound such that $\|\boldsymbol{w}_{\Delta}\|_{\mathcal{L}_\infty} \leqslant \|\Delta \boldsymbol{z}_{\Delta}\|_{\mathcal{L}_\infty}$ is given by

$$\|\boldsymbol{\Delta}\| \leq \|\text{diag}(\alpha_{\text{M}}, \alpha_{\Psi})\| = \max(\alpha_{\text{M}}, \alpha_{\Psi}). \tag{B.4}$$

To obtain the gain bound (B.4), note that the spatial norm in the definition of the $\mathcal{L}_\infty$-norm can be any vector p-norm [103]; here, $p = 2$. Finally, in (B.4), $\boldsymbol{\Delta} \in \mathbb{R}^{2n \times (3n+1)}$ is any real matrix of appropriate size and norm. From their definitions, however, it is known that $\boldsymbol{w}^{\text{M}}_{\Delta}(\boldsymbol{u})$ and $\boldsymbol{w}^{\psi}_{\Delta}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are decoupled. It follows that the relevant uncertain matrices can be restricted to a set $\mathcal{D}_{\Delta}$ of block diagonal matrices with separate norm bounds

$$\mathcal{D}_{\Delta} = \left\{ \boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\Delta}_{\text{M}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Delta}_{\Psi} \end{bmatrix} :\ \boldsymbol{\Delta}_{\text{M}} \in \mathbb{R}^{n \times n}, \boldsymbol{\Delta}_{\Psi} \in \mathbb{R}^{n \times (2n+1)}, \|\boldsymbol{\Delta}_{\text{M}}\| \leq \alpha_{\text{M}}, \|\boldsymbol{\Delta}_{\Psi}\| \leq \alpha_{\Psi} \right\}.$$

■

B.2 Proof of Theorem 4.2

Proof: Begin by checking that $(\boldsymbol{A}_{11}, \boldsymbol{B}_{13})$ is stabilizable and $(\boldsymbol{A}_{11}, \boldsymbol{C}_{31})$ detectable, which can be easily verified, for example, the generalized plant given by (4.22)–(4.26). Further, $\boldsymbol{D}_{33} = 0$, *i. e.*, there is no direct feedthrough in the control channel of plant. Then, for the general controller (4.29), a coprime factorization can be obtained directly in state-space by adopting the formulae (A.21) with $\boldsymbol{Z} = \boldsymbol{Y} = \boldsymbol{I}$, yielding

$$\begin{bmatrix} M_0 \, U_0 \\ N_0 \, V_0 \end{bmatrix} : \left[\begin{array}{cc|cc} \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{F}_{\mathrm{G}} & \boldsymbol{0} & \boldsymbol{B}_{13} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{\mathrm{K}} + \boldsymbol{B}_{\mathrm{K}}\boldsymbol{F}_{\mathrm{K}} & \boldsymbol{0} & \boldsymbol{B}_{\mathrm{K}} \\ \hline \boldsymbol{F}_{\mathrm{G}} & \boldsymbol{C}_{\mathrm{K}} + \boldsymbol{D}_{\mathrm{K}}\boldsymbol{F}_{\mathrm{K}} & \boldsymbol{I} & \boldsymbol{D}_{\mathrm{K}} \\ \boldsymbol{C}_{31} & \boldsymbol{F}_{\mathrm{K}} & \boldsymbol{0} & \boldsymbol{I} \end{array}\right], \tag{B.5}$$

$$\begin{bmatrix} \tilde{V}_0 & -\tilde{U}_0 \\ -\tilde{N}_0 & \tilde{M}_0 \end{bmatrix} : \left[\begin{array}{cc|cc} \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31} & \boldsymbol{B}_{13}\boldsymbol{C}_{11} & -\boldsymbol{B}_{13} & \boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}} \\ \boldsymbol{B}_{\mathrm{K}}\boldsymbol{C}_{31} & \boldsymbol{A}_{\mathrm{K}} & \boldsymbol{0} & \boldsymbol{B}_{\mathrm{K}} \\ \hline \boldsymbol{F}_{\mathrm{G}} - \boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31} & -\boldsymbol{C}_{\mathrm{K}} & \boldsymbol{I} & -\boldsymbol{D}_{\mathrm{K}} \\ \boldsymbol{C}_{31} & -\boldsymbol{F}_{\mathrm{K}} & \boldsymbol{0} & \boldsymbol{I} \end{array}\right]. \tag{B.6}$$

Next, an expression for T_Δ from (4.28) has to be calculated, which is repeated here for convenience:

$$\begin{bmatrix} T_{\Delta,11} & T_{\Delta,12} \\ T_{\Delta,21} & T_{\Delta,22} \end{bmatrix} = \begin{bmatrix} G_{z_\Delta w_\Delta} + G_{z_\Delta u} U_0 \tilde{M}_0 G_{y w_\Delta} & G_{z_\Delta u} M_0 \\ \tilde{M}_0 G_{y w_\Delta} & 0 \end{bmatrix}.$$

Let us also explicitly state the following systems determined from (4.22) as

$$G_{z_\Delta w_\Delta} : \left[\begin{array}{c|c} \boldsymbol{A}_{11} & \boldsymbol{B}_{11} \\ \hline \boldsymbol{C}_{11} & \boldsymbol{D}_{11} \end{array}\right], \quad G_{y w_\Delta} : \left[\begin{array}{c|c} \boldsymbol{A}_{11} & \boldsymbol{B}_{11} \\ \hline \boldsymbol{C}_{31} & \boldsymbol{D}_{31} \end{array}\right], \quad G_{z_\Delta u} : \left[\begin{array}{c|c} \boldsymbol{A}_{11} & \boldsymbol{B}_{13} \\ \hline \boldsymbol{C}_{11} & \boldsymbol{D}_{13} \end{array}\right].$$

The steps to obtain a compact expression of $T_{\Delta,11}$ are to insert the coprime factors U_0 and $\tilde{M}_0$ from (B.6) and algebraic simplification, subsequent application of a state similarity transformation such that $\boldsymbol{\xi} = \overline{\boldsymbol{T}}\boldsymbol{x}$ with

$$\overline{\boldsymbol{T}} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{I} & 0 & 0 & 0 \\ 0 & -\boldsymbol{I} & 0 & 0 & \boldsymbol{I} & 0 & 0 \\ 0 & 0 & \boldsymbol{I} & 0 & 0 & 0 & 0 \\ \boldsymbol{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\boldsymbol{I} & 0 & \boldsymbol{I} & 0 \\ -\boldsymbol{I} & 0 & 0 & 0 & 0 & 0 & \boldsymbol{I} \end{bmatrix}$$

and removal of 4 uncontrollable and 1 unobservable modes. A realization is finally obtained as

$$T_{\Delta,11} : \left[\begin{array}{cc|c} \boldsymbol{A}_{11} + \boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31} & \boldsymbol{B}_{13}\boldsymbol{C}_{11} & \boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31} + \boldsymbol{B}_{11} \\ \boldsymbol{B}_{\mathrm{K}}\boldsymbol{C}_{31} & \boldsymbol{A}_{11} & \boldsymbol{B}_{\mathrm{K}}\boldsymbol{D}_{31} \\ \hline \boldsymbol{D}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31} + \boldsymbol{C}_{11} & \boldsymbol{D}_{13}\boldsymbol{C}_{\mathrm{K}} & \boldsymbol{D}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31} + \boldsymbol{D}_{11} \end{array}\right].$$

By analogous steps, for $T_{\Delta,12}$ and $T_{\Delta,21}$ one obtains

$$T_{\Delta,12}:\left[\begin{array}{c|c}\boldsymbol{A}_{11}+\boldsymbol{B}_{13}\boldsymbol{F}_{\mathrm{G}} & \boldsymbol{B}_{13}\\ \hline \boldsymbol{D}_{13}\boldsymbol{F}_{\mathrm{G}}+\boldsymbol{C}_{11} & \boldsymbol{D}_{13}\end{array}\right]$$

and

$$T_{\Delta,21}:\left[\begin{array}{cc|c}\boldsymbol{A}_{11}+\boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31} & \boldsymbol{B}_{13}\boldsymbol{C}_{11} & \boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31}+\boldsymbol{B}_{11}\\ \boldsymbol{B}_{11}\boldsymbol{C}_{31} & \boldsymbol{A}_{11} & \boldsymbol{B}_{\mathrm{K}}\boldsymbol{D}_{31}\\ \hline \boldsymbol{C}_{31} & -\boldsymbol{F}_{\mathrm{K}} & \boldsymbol{D}_{31}\end{array}\right],$$

where one uncontrollable and one unobservable mode were removed, respectively. Consequently, assuming $\Delta \triangleq \boldsymbol{\Delta}$ as a non-dynamic matrix and defining

$$\tilde{\boldsymbol{D}} \triangleq \boldsymbol{I}-\boldsymbol{D}_{11}\boldsymbol{\Delta}-\boldsymbol{D}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta},$$

a realization for $(I-T_{\Delta,11}\Delta)^{-1}$ is

$$(I-T_{\Delta,11}\Delta)^{-1}:\left[\begin{array}{c|c}\bar{\boldsymbol{A}} & \bar{\boldsymbol{B}}\\ \hline \bar{\boldsymbol{C}} & \bar{\boldsymbol{D}}\end{array}\right], \tag{B.7}$$

where

$$\begin{aligned}
\bar{\boldsymbol{A}} &= \begin{bmatrix}\bar{\boldsymbol{A}}_{11} & \bar{\boldsymbol{A}}_{12}\\ \bar{\boldsymbol{A}}_{21} & \bar{\boldsymbol{A}}_{22}\end{bmatrix},\\
\bar{\boldsymbol{A}}_{11} &= \boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31}+\boldsymbol{B}_{11}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{C}_{11}+\boldsymbol{B}_{11}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31}+\boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{C}_{11}\\
&\quad +\boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31}+\boldsymbol{A}_{11},\\
\bar{\boldsymbol{A}}_{12} &= \boldsymbol{B}_{13}\boldsymbol{C}_{\mathrm{K}}+\boldsymbol{B}_{11}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{C}_{\mathrm{K}}+\boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{C}_{\mathrm{K}},\\
\bar{\boldsymbol{A}}_{21} &= \boldsymbol{B}_{\mathrm{K}}\boldsymbol{C}_{31}+\boldsymbol{B}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{C}_{11}+\boldsymbol{B}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31},\\
\bar{\boldsymbol{A}}_{22} &= \boldsymbol{A}_{\mathrm{K}}+\boldsymbol{B}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{C}_{\mathrm{K}},\\
\bar{\boldsymbol{B}} &= \begin{bmatrix}\boldsymbol{B}_{11}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}-\boldsymbol{B}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\\ -\boldsymbol{B}_{\mathrm{K}}\boldsymbol{D}_{31}\boldsymbol{\Delta}\tilde{\boldsymbol{D}}^{-1}\end{bmatrix},\\
\bar{\boldsymbol{C}} &= \left[-\tilde{\boldsymbol{D}}^{-1}\boldsymbol{C}_{11}-\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{D}_{\mathrm{K}}\boldsymbol{C}_{31}-\tilde{\boldsymbol{D}}^{-1}\boldsymbol{D}_{13}\boldsymbol{C}_{\mathrm{K}}\right],\\
\bar{\boldsymbol{D}} &= \tilde{\boldsymbol{D}}^{-1}.
\end{aligned}$$

Finally, plugging the above realizations of $T_{\Delta,12}$, $(I-T_{\Delta,11}\Delta)^{-1}$ and $T_{\Delta,21}$ into (4.27) and a number of straightforward but tedious simplifications yield the result (4.31). ■

B.3 Proof of Lemma 4.1

Proof: In order to obtain a robustly stable loop, the stability condition on (Q, S) from Prop. 2.8 is not sufficient yet because it only refers to the controlled channel; in Fig. 4.5, however, the pre-stabilized, yet uncertain loop $T(S)$ is subject to exogenous inputs $\boldsymbol{w}$. Hence, to ensure robust closed-loop internal stability, it is required that the pair $\left(\begin{bmatrix} 0 & 0 \\ 0 & Q \end{bmatrix}, T(S)\right)$ be stable [231, p. 80] for all allowed perturbations. To this end, it is sufficient to show [231, p. 32] that $T_{11}(S), T_{12}(S), T_{21}(S) \in \mathcal{RH}_\infty$ as $S \in \mathcal{RH}_\infty$, and $(Q, T_{22}(S)) = (Q, S)$ should be a stabilizing loop by construction. Denoting with $P(S)$ the transfer matrix between $\mathrm{col}(\boldsymbol{w}, \boldsymbol{u})$ and $\mathrm{col}(\boldsymbol{z}, \boldsymbol{y})$ for the plant determined by S, the uncertain operator $T(S)$ is given by [231, p. 79]

$$T(S) = \begin{bmatrix} T_{11}(S) & T_{12}(S) \\ T_{21}(S) & T_{22}(S) \end{bmatrix} = \begin{bmatrix} P_{11}(S) + P_{12}(S)U\tilde{M}(S)P_{21}(S) & P_{12}(S)M(S) \\ \tilde{M}(S)P_{21}(S) & S \end{bmatrix}. \tag{B.8}$$

By the first condition (i), consider that (4.33) is necessary and sufficient for the stability of the nominal uncertain loop. Then, for all $\Delta \in \mathcal{D}_\Delta$, there exists a stable dual Youla operator $S \in \mathcal{RH}_\infty$ whose coprime factors satisfy $\tilde{M}(S), M(S) \in \mathcal{RH}_\infty$ by construction and consequently $\mathcal{S}_\Delta \subset \mathcal{RH}_\infty$. Using (4.27) to calculate S, by the factorization, the plant under control $P(S)$ in (B.8) is precisely that of the uncertain plant $\mathcal{F}_u(G, \Delta)$ from the controller's point of view. Thus, with (4.33) also $P_{11}(S), P_{12}(S), P_{21}(S) \in \mathcal{RH}_\infty$ and consequently $T(S) \in \mathcal{RH}_\infty, \forall \Delta \in \mathcal{D}_\Delta$. Alternatively, the requirement (i) could also be proved by coprimeness of the factors involved [237, App. B]. The requirement (ii) follows directly from the small-gain condition of Prop. 2.8: S is an uncertain stable time-varying operator and the only assumption about the plug-in controller Q is that $\|Q\|_\infty \leq \gamma_\mathrm{Q}$ for some finite γ_Q. The small-gain theorem is therefore necessary [269, Th. 9.1] to ensure that all $S \in \mathcal{S}_\Delta$ are stabilized by all $Q \in \mathcal{Q}$. ∎

B.4 Proof of Theorem 3.1

Proof: The steps in the derivation are similar to the gain scheduling literature exploiting the general Youla parameterization [185]. Consider coprime factorizations of the nominal plant $G_0 = \tilde{M}_0^{-1}\tilde{N}_0 = N_0 M_0^{-1}$, respectively of the nominal controller $K_0 = \tilde{V}_0^{-1}\tilde{U}_0 = U_0 V_0^{-1}$ such that the double Bezout identity (2.4) holds, which is repeated here for convenience:

$$\begin{bmatrix} \tilde{V}_0 & -\tilde{U}_0 \\ -\tilde{N}_0 & \tilde{M}_0 \end{bmatrix} \begin{bmatrix} M_0 & U_0 \\ N_0 & V_0 \end{bmatrix} = \begin{bmatrix} M_0 & U_0 \\ N_0 & V_0 \end{bmatrix} \begin{bmatrix} \tilde{V}_0 & -\tilde{U}_0 \\ -\tilde{N}_0 & \tilde{M}_0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{B.9}$$

According to Prop. 2.1, all internally stabilizing controllers for G_{yu} can be written in terms of a parameter system $Q \in \mathcal{RH}_\infty$ in a right stable fractional form as

$$K(Q) = (U_0 + M_0 Q)(V_0 + N_0 Q)^{-1}. \tag{B.10}$$

In the controlled channel of (3.3), all states are measurable without direct feedthrough, *i. e.*, $G_{yu} : \delta\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_2\boldsymbol{u},\ \boldsymbol{y} = \boldsymbol{x}$. Therefore, by $\boldsymbol{C}_2 = \boldsymbol{I}$ and $\boldsymbol{D}_{22} = \boldsymbol{0}$ and for static state feedback $\boldsymbol{u} = \boldsymbol{D}_{\mathrm{K},i}\,\boldsymbol{x}$, the state-space realizations of the general coprime factorization (A.21) specialize to

$$\begin{bmatrix} M_i & U_i \\ N_i & V_i \end{bmatrix} : \left[\begin{array}{c|cc} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{B}_2 & \boldsymbol{0} \\ \hline \boldsymbol{F} & \boldsymbol{I} & \boldsymbol{D}_{\mathrm{K},i} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I} \end{array}\right], \quad \begin{bmatrix} \tilde{V}_i & -\tilde{U}_i \\ -\tilde{N}_i & \tilde{M}_i \end{bmatrix} : \left[\begin{array}{c|cc} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{D}_{\mathrm{K},i} & -\boldsymbol{B}_2 & \boldsymbol{B}_2\boldsymbol{D}_{\mathrm{K},i} \\ \hline \boldsymbol{F}-\boldsymbol{D}_{\mathrm{K},i} & \boldsymbol{I} & -\boldsymbol{D}_{\mathrm{K},i} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I} \end{array}\right]. \tag{B.11}$$

The parameterized controller (B.10) can be reformulated as lower fractional transformation $K = \mathcal{F}_\ell\left(J, Q\right)$ with the central system from (2.8), yielding $J = \begin{bmatrix} U_0 V_0^{-1} & \tilde{V}_0^{-1} \\ V_0^{-1} & -V_0^{-1} N_0 \end{bmatrix}$. Employing the realizations of the coprime factors from (B.11) and reducing to a minimal realization, the central system (3.7) is obtained. In order to calculate the set $\mathcal{Q}$ corresponding to $\mathcal{K}$, both the nominal controller K_0 as well as controllers $K_i \in \mathcal{K}$ are expressed by means of coprime factors (B.11). Thus, coprime factorizations are constructed for all the plant/controller interconnection pairs. One may then work with the coprime factors in order to consider the differences between the loops. The Q_i corresponding to controller K_i, given a factorization of the central controller K_0, can be calculated by $Q_i = (-\tilde{U}_0 + \tilde{V}_0 K_i)(\tilde{M}_0 - \tilde{N}_0 K_i)^{-1}$ [231, Ch. 8.3]. One can use

$$Q_i = \tilde{U}_i V_0 - \tilde{V}_i U_0 = \tilde{V}_i\left(K_i - K_0\right) V_0 \tag{B.12}$$

from [161, Th.1] alternatively. Plugging in the factors (B.11), the state-space construction of parameters Q_i follows as (3.10) after removal of one unobservable state. Recovery of the local controllers can now be shown by interconnecting (3.7) with (3.10). Removing 5 uncontrollable and 1 unobservable states, it follows indeed $\mathcal{F}_\ell\left(J, Q_i\right) = K_i$. Stability under arbitrary interpolation, however, is not yet ensured by (3.7) and (3.10). To see this, consider the closed loop $\mathcal{F}_\ell\left(G, \mathcal{F}_\ell\left(J, Q\right)\right)$ that can be reduced to dynamics with state matrix

$$\boldsymbol{A}_{\mathrm{cl}} = \begin{bmatrix} \boldsymbol{A} + \boldsymbol{B}_2 \boldsymbol{D}_{\mathrm{K},0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A} + \boldsymbol{B}_2 \boldsymbol{F} & \boldsymbol{B}_2 \boldsymbol{C}_{\mathrm{Q}}(\boldsymbol{\alpha}) & \boldsymbol{B}_2 \boldsymbol{D}_{\mathrm{Q}}(\boldsymbol{\alpha}) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{\mathrm{Q}}(\boldsymbol{\alpha}) & \boldsymbol{B}_{\mathrm{Q}}(\boldsymbol{\alpha}) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A} + \boldsymbol{B}_2 \boldsymbol{D}_{\mathrm{K},0} \end{bmatrix},$$

where $\boldsymbol{A}_{\mathrm{Q}}(\boldsymbol{\alpha}), \boldsymbol{B}_{\mathrm{Q}}(\boldsymbol{\alpha}), \boldsymbol{C}_{\mathrm{Q}}(\boldsymbol{\alpha}), \boldsymbol{D}_{\mathrm{Q}}(\boldsymbol{\alpha})$ define the realization of the plug-in filter $Q(\boldsymbol{\alpha})$. Given the block-diagonal respectively block-triangular structure of $\boldsymbol{A}_{\mathrm{cl}}$ and invoking Lem. A.1 three times, there exists a CQLF if there exists one for each sub-block on the diagonal. Hence, (3.9) is enforced for $\boldsymbol{A}_{\mathrm{Q}}$, the matrices $\boldsymbol{A} + \boldsymbol{B}_2 \boldsymbol{D}_{\mathrm{K},0}$ and $\boldsymbol{A} + \boldsymbol{B}_2 \boldsymbol{F}$ are stable by construction. As pointed out in [80], there is always a transformation to realize $\mathcal{Q}$ such that a CQLF (3.8) exists. ■

B.5 Proof of Theorem 3.2

Proof: Connecting the controller to the switching plant results in the closed-loop dynamics

$$\begin{bmatrix} \boldsymbol{x}(t+1) \\ \boldsymbol{x}_{\mathrm{Q}}(t+1) \\ \tilde{\boldsymbol{x}}(t+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{\sigma(t)} + \boldsymbol{B}_{2,\sigma(t)} \boldsymbol{F}_{\sigma(t)} & \boldsymbol{B}_{2,\sigma(t)} \boldsymbol{\Xi}_{\sigma(t)}(t) & \boldsymbol{B}_{2,\sigma(t)} \boldsymbol{F}_{\sigma(t)} \\ \boldsymbol{0} & \boldsymbol{A}_{\mathrm{Q},\sigma(t)} & \boldsymbol{B}_{\mathrm{Q},\sigma(t)} \boldsymbol{C}_{2,\sigma(t)} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{\sigma(t)} + \boldsymbol{L}_{\sigma(t)} \boldsymbol{C}_{2,\sigma(t)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{x}_{\mathrm{Q}}(t) \\ \tilde{\boldsymbol{x}}(t) \end{bmatrix}, \tag{B.13}$$

where $\tilde{\boldsymbol{x}}(t) = \hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)$ denotes the state estimation error. Given the solvability of (2.24), the blocks $\boldsymbol{A}_{\sigma(t)} + \boldsymbol{B}_{2,\sigma(t)} \boldsymbol{F}_{\sigma(t)}$ and $\boldsymbol{A}_{\sigma(t)} + \boldsymbol{L}_{\sigma(t)} \boldsymbol{C}_{2,\sigma(t)}$ are quadratically stable with a common quadratic Lyapunov matrix each. Furthermore, the elements $\boldsymbol{A}_{\mathrm{Q},\sigma(t)}$ also share a common quadratic Lyapunov function by construction (3.18). Thus, since (B.13) is in upper block-triangular form the overall closed loop system is switching stable as long as all $\Xi_i(t)$ are bounded. ■

C

Details on Robot Manipulators

C.1 Detailed Derivation of Manipulator Norm Bounds

Details how the bounding function $\Phi(\|\boldsymbol{x}\|)$ in (4.11) is obtained for a specific manipulator are usually omitted in the literature and some authors assume a constant, *i. e.*, very conservative, bound [210]. It is also common [13] to use the bound quadratic in $\|\boldsymbol{x}\|$ to account for the velocity cross-product terms. In this appendix, the technicalities how to obtain a bound of the form (4.11) are spelled out. This should help the reader by clarifying the meaning of the single components of the uncertainty in the derivation given in Appendix B.1. For details on the computational procedures to obtain numeric values, the interested reader is additionally referred to [191, 69].

First, let us write out the neglected nonlinearities (4.4b) as a sum of their components $\tilde{\boldsymbol{n}} = \tilde{\boldsymbol{n}}_\mathrm{C} + \tilde{\boldsymbol{n}}_\mathrm{f} + \tilde{\boldsymbol{n}}_\mathrm{g}$ and recall the available overestimates from Assumptions 4.1 and 4.2. In order to obtain a bound on Coriolis/centrifugal mismatch

$$\tilde{\boldsymbol{n}}_\mathrm{C} = \left(\hat{\boldsymbol{C}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}) - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right)\dot{\boldsymbol{q}} + \hat{\boldsymbol{C}}(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}})\boldsymbol{w}_2,$$

let the worst-case $\|\hat{\boldsymbol{C}} - \boldsymbol{C}\|$ be described by some $\tilde{a}_\mathrm{c}\|\hat{\dot{\boldsymbol{q}}}\|$, $\tilde{a}_\mathrm{c} \in \mathbb{R}_0^+$ (*e. g.* obtained by simulation), then by the triangle inequality

$$\|\tilde{\boldsymbol{n}}_\mathrm{C}\| \leq \tilde{a}_\mathrm{c}\|\hat{\dot{\boldsymbol{q}}}\| \cdot \|\dot{\boldsymbol{q}}\| + \hat{a}_\mathrm{c}\|\hat{\dot{\boldsymbol{q}}}\|\|\boldsymbol{w}_2\| \leq \tilde{a}_\mathrm{c}\|\dot{\boldsymbol{q}}\|^2 + \tilde{a}_\mathrm{c}\|\boldsymbol{w}_2\|\|\dot{\boldsymbol{q}}\| + \hat{a}_\mathrm{c}\|\dot{\boldsymbol{q}}\|\|\boldsymbol{w}_2\| + \hat{a}_\mathrm{c}\|\boldsymbol{w}_2\|^2.$$

Using Ass. 4.1 and Ass. 4.2, we obtain[1]

$$\|\tilde{\boldsymbol{n}}_\mathrm{C}\| \leq \underbrace{\left(\tilde{a}_\mathrm{c}(v_\mathrm{max} + W_2) + \hat{a}_\mathrm{c}W_2\right)}_{\triangleq \alpha_{1,\mathrm{c}}}\|\dot{\boldsymbol{q}}\| + \underbrace{\hat{a}_\mathrm{c}W_2^2}_{\triangleq \alpha_{0,\mathrm{c}}}.$$

Assuming viscous and dynamic[2] friction terms,

$$\tilde{\boldsymbol{n}}_\mathrm{f} = \hat{F}_\mathrm{v}\hat{\dot{\boldsymbol{q}}} + \hat{F}_\mathrm{d}(\hat{\dot{\boldsymbol{q}}}) - F_\mathrm{v}\dot{\boldsymbol{q}} - F_\mathrm{d}(\dot{\boldsymbol{q}})$$

[1]Given the realistic assumption on bounded workspace and bounded velocities, one can introduce $\alpha_2\|\boldsymbol{x}\|^2 \leq \alpha_2^{'}\|\boldsymbol{x}\|, \alpha_2^{'} = \alpha_2 v_\mathrm{max}$ at the expense of some conservatism [127, Ch. 17].

[2]Here, *dynamic* friction refers to the naming convention in robotics, but it is assumed to be non-dynamic in the sense that there is no memory (state) in the friction model.

and by $\|F_\text{v}\dot{\boldsymbol{q}} + F_\text{d}(\dot{\boldsymbol{q}})\| \leq a_\text{f,v}\|\dot{\boldsymbol{q}}\| + a_\text{f,d}$ as used in [132, Ch. 3.3],

$$\|\tilde{\boldsymbol{n}}_\text{f}\| \leq \underbrace{\tilde{a}_\text{f,v}}_{\triangleq \alpha_{1,\text{f}}} \|\dot{\boldsymbol{q}}\| + \underbrace{\hat{a}_\text{f,v} W_2 + \tilde{a}_\text{f,d}}_{\triangleq \alpha_{0,\text{f}}},$$

where again it was assumed that the worst-case difference can be described by some $\tilde{a}_\text{f,v}$. Similarly, the gravity error is bounded by $\|\tilde{\boldsymbol{n}}_\text{g}\| \leq \|\hat{\boldsymbol{g}}(\hat{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q})\| = \tilde{a}_\text{g} \triangleq \alpha_{0,\text{g}}$, in case of only revolute joints, or by some $\|\tilde{\boldsymbol{n}}_\text{g}\| \leq \alpha_{1,\text{g}}\|\boldsymbol{q}\| + \alpha_{0,\text{g}}$ in case of prismatic joints. Hence, in sum the bound is given by

$$\|\tilde{\boldsymbol{n}}\| \leq \alpha_{1,\text{g}}\|\boldsymbol{q}\| + (\alpha_{1,\text{c}} + \alpha_{1,\text{f}})\, \|\dot{\boldsymbol{q}}\| + (\alpha_{0,\text{c}} + \alpha_{0,\text{f}} + \alpha_{0,\text{g}})\,.$$

Using (B.1), one obtains a bound of the form (4.11) by

$$\|\tilde{\boldsymbol{n}}\| \leq \underbrace{\|[\alpha_{1,\text{g}}, \alpha_{1,\text{c}} + \alpha_{1,\text{f}}]\|}_{\alpha_1}\|\boldsymbol{x}\| + \underbrace{\alpha_{0,\text{c}} + \alpha_{0,\text{f}} + \alpha_{0,\text{g}}}_{\alpha_0}\,. \tag{C.1}$$

C.2 Models of Chapter 4

Vertical planar elbow manipulator example. For the description and detailed physical parameters of the particular manipulator, the reader is referred to [203]. Normalizing w. r. t. SI units, the resulting model is given by

$$\boldsymbol{M} = \begin{bmatrix} 0.162\cos(q_2) + 0.655 & 0.0809\cos(q_2) + 0.142 \\ 0.0809\cos(q_2) + 0.142 & 0.356 \end{bmatrix}, \tag{C.2a}$$

$$\boldsymbol{C} = \begin{bmatrix} -0.0809\dot{q}_2\sin(q_2) & -0.0809\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ 0.0809 q_1\sin(q_2) & 0 \end{bmatrix}, \tag{C.2b}$$

$$\boldsymbol{g} = \begin{bmatrix} 3.60\cos(q_1 + q_2) + 9.35\cos(q_1) \\ 3.60\cos(q_1 + q_2) \end{bmatrix}, \tag{C.2c}$$

$$\boldsymbol{f} = \text{blkdiag}(3.00, 3.00)\,\dot{\boldsymbol{q}}. \tag{C.2d}$$

To simulate an inaccurate AID control example, a modified dynamical model is used:

$$\hat{\boldsymbol{M}} = \begin{bmatrix} 0.17\cos(q_2) + 0.702 & 0.0852\cos(q_2) + 0.178 \\ 0.0852\cos(q_2) + 0.178 & 0.437 \end{bmatrix}, \tag{C.3a}$$

$$\hat{\boldsymbol{C}} = \begin{bmatrix} -0.0852\dot{q}_2\sin(q_2) & -0.0852\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ 0.0852 q_1\sin(q_2) & 0 \end{bmatrix}, \tag{C.3b}$$

$$\hat{\boldsymbol{g}} = \begin{bmatrix} 4.07\cos(q_1 + q_2) + 8.87\cos(q_1) \\ 4.07\cos(q_1 + q_2) \end{bmatrix}, \tag{C.3c}$$

$$\hat{\boldsymbol{f}} = \text{blkdiag}(4.50, 4.50)\,\dot{\boldsymbol{q}}. \tag{C.3d}$$

Table C.1: Common cases of control laws employed in the inner approximate inverse dynamics loop (4.3) and illustrative numeric parameters used in the simulation study, with the values of the uncertainty bounds according to Thm. 4.1.

Inner Controller		**Inertia Model**	**Nonlinearities**	**Bounds**	
Type	**Abbr.**			α_{M}	α_{Ψ}
Perfect feedback linearization	PFL	$\boldsymbol{M}$ from (C.2)	$\hat{\boldsymbol{C}} = \boldsymbol{C}, \quad \hat{\boldsymbol{f}} = \boldsymbol{f}, \quad \hat{\boldsymbol{g}} = \boldsymbol{g}$	0	0
Approximate inverse dynamics	AID	$\hat{\boldsymbol{M}}$ from (C.3)	$\hat{\boldsymbol{C}}, \hat{\boldsymbol{f}}, \hat{\boldsymbol{g}}$ from (C.3)	0.2523	9.092
Simplified inverse dynamics	SID	$\mathrm{diag}(0.655, 0.356)$	$\hat{\boldsymbol{C}} = \boldsymbol{0}, \quad \hat{\boldsymbol{f}} = \boldsymbol{0}, \quad \hat{\boldsymbol{g}} = \boldsymbol{g}$	0.6457	59.37
Diagonal scaling	DS	$\mathrm{diag}(0.655, 0.356)$	$\hat{\boldsymbol{C}} = \boldsymbol{0}, \quad \hat{\boldsymbol{f}} = \boldsymbol{0}, \quad \hat{\boldsymbol{g}} = \boldsymbol{0}$	0.6457	77.99
Gravity compensation	GC	$\boldsymbol{I}$	$\hat{\boldsymbol{C}} = \boldsymbol{0}, \quad \hat{\boldsymbol{f}} = \boldsymbol{0}, \quad \hat{\boldsymbol{g}} = \boldsymbol{g}$	2.765	59.37
No inverse dynamics	NID	$\boldsymbol{I}$	$\hat{\boldsymbol{C}} = \boldsymbol{0}, \quad \hat{\boldsymbol{f}} = \boldsymbol{0}, \quad \hat{\boldsymbol{g}} = \boldsymbol{0}$	2.765	77.99

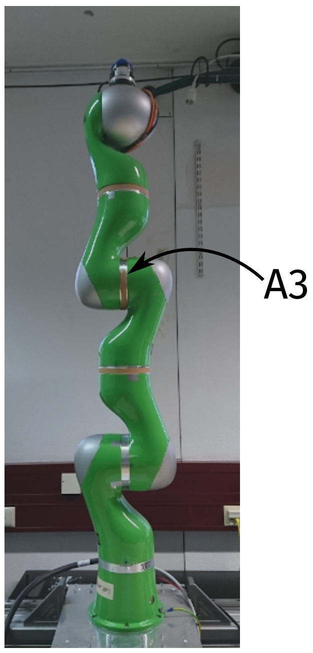


Figure C.1: The KUKA LWR IV+ robot manipulator used in the control experiments with variable impedance implementation exploiting a suitably realized Q-parameterization approach.

C.3 Experimental Setup of Chapter 7

KUKA LWR IV+. For an overview of the robotic system, the reader is referred to [26]. The manipulator shown in Fig. C.1 is connected to a control box, supplied by the manufacturer, which provides a communication interface based on the UDP protocol— the so-called fast research interface (FRI) [207]. Thereby, a soft real-time capable control interface is provided working with a fixed rate of data exchange at 1 kHz. At the other side, a standard personal computer is used running a real-time capable Linux operating system with a PREEMPT_RT patched kernel. The control algorithms were implemented in MATLAB/Simulink and translated into executable code by means of code generation.

The FRI does not provide measured joint velocities $\dot{\boldsymbol{q}}_{\mathrm{msr}}$. Therefore, the joint velocities are estimated [132, p. 226] from the measured joint positions $\boldsymbol{q}_{\mathrm{msr}}$ by calculating a discrete approximate derivative and low-pass filtering with a Butterworth filter of second order with a cut-off frequency of $10\,\mathrm{rad\,s^{-1}}$. The widespread quasi-continuous approach of modeling the plant/controller in continuous-time and subsequent implicit discretization by means of the solver resulted in useless control behavior on the experimental hardware: it is crucial to work with a discrete-time formulation of the Q-parameterization in the robotic hardware experiments, utilizing discretized plant models and the discrete-time equations in the controller design.

The limits of the signal that can be superimposed on the commanded torque before going into saturation were identified experimentally as

$$\boldsymbol{\tau}_{\mathrm{FRI,sat}} = [175 \quad 175 \quad 100 \quad 100 \quad 100 \quad 35 \quad 35]^{\top}.$$

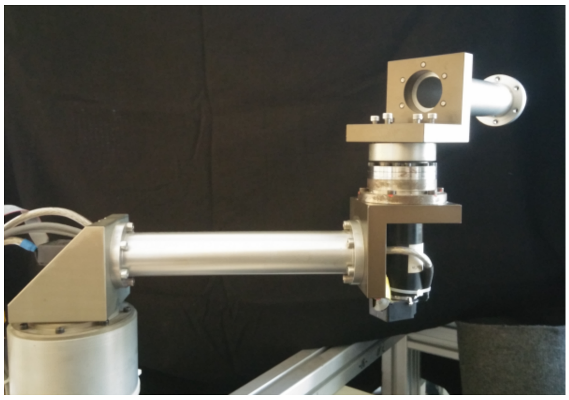


Figure C.2: The two DoF robot manipulator used in the hardware experiment of Chap. 8.

C.4 Experimental Setup of Chapter 8

Horizontal planar elbow manipulator. The parameters of the robot manipulator with two rotational degrees of freedom are provided in Tab. C.2. The values are normalized w. r. t. SI units unless stated differently. The dynamic behavior of the robot manipulator model was simulated using

$$\boldsymbol{M} = \begin{bmatrix} 0.0286\cos(q_2) + 0.442 & 0.0143\cos(q_2) + 0.00880 \\ 0.0143\cos(q_2) + 0.00880 & 0.223 \end{bmatrix}, \tag{C.4a}$$

$$\boldsymbol{n} = \begin{bmatrix} -0.0140\sin(q_2)\dot{q}_2^2 - 0.0290\sin(q_2)\dot{q}_1\dot{q}_2 + 0.00260\dot{q}_1 \\ 0.0140\dot{q}_1^2\sin(q_2) + 0.00260\dot{q}_2 \end{bmatrix}. \tag{C.4b}$$

In the hardware experimental setup depicted in Fig. C.2, Maxon motors were used in each joint with incremental encoders on the motor shaft for position measurement as well as Harmonic Drive gears with ratio 1:100. The motors were driven by a current source in pulse width modulation mode, amplifying a voltage supplied by a D/A pin on a Sensoray 626 multifunction analog/digital I/O PCI board. The interested reader is referred to [74] for more details of this manipulator.

Table C.2: Parameters of the example robot manipulator and the nominal PD controller.

Physical Parameter	Value
Length of link 1	0.3 m
Length of link 2	0.2 m
Angular range $\lvert q_1\rvert$	$\pi/2$
Angular range $\lvert q_2\rvert$	π

Controller Parameter	Value
T_s	1 ms
P-gain	diag(500, 500)
D-gain	diag(22.4, 22.4)

List of Figures

List of Tables

Bibliography

[1] C. Abdallah, D. M. Dawson, P. Dorato, and M. Jamshidi, “Survey of robust control for rigid robots,” *IEEE Control Syst.*, vol. 11, no. 2, pp. 24–30, Feb. 1991.

[2] D. Alazard and P. Apkarian, “Exact Observer-Based Structures for Arbitrary Compensators,” *Int. J. Robust Nonlinear Control*, vol. 9, no. 2, pp. 101–118, 1999.

[3] E. Alibekov, J. Kubalík, and R. Babuška, “Policy derivation methods for critic-only reinforcement learning in continuous spaces,” *Eng. Appl. Art. Intell.*, vol. 69, pp. 178–187, 2018.

[4] M. D. S. Aliyu, *Nonlinear H_∞ control, Hamiltonian systems and Hamilton-Jacobi equations.* Boca Raton, FL: CRC Press, 2011.

[5] E. Anderlini, D. I. M. Forehand, E. Bannon, and M. Abusara, “Control of a realistic wave energy converter model using least-squares policy iteration,” *IEEE Trans. Sustain. Energy*, vol. 8, no. 4, pp. 1618–1628, Oct. 2017.

[6] B. D. O. Anderson, “From Youla-Kucera to Identification, Adaptive and Nonlinear Control,” *Automatica*, vol. 34, no. 12, pp. 1485–1506, 1998.

[7] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1990.

[8] C. W. Anderson, P. M. Young, M. R. Buehner, J. N. Knight, K. A. Bush, and D. C. Hittle, “Robust reinforcement learning control using integral quadratic constraints for recurrent neural networks,” *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 993–1002, Jul. 2007.

[9] J. Anderson, J. C. Doyle, S. H. Low, and N. Matni, “System level synthesis,” *Annual Rev. Control*, vol. 47, pp. 364–393, 2019.

[10] K. Aström and R. Murray, *Feedback Systems: An Introduction for Scientists and Engineers.* Princeton, NJ, USA: Princeton University Press, 2010.

[11] K. Åström and B. Wittenmark, *Adaptive Control*, 2nd ed. Reading, MA: Dover Publications, 2008.

[12] G. J. Balas, A. K. Packard, M. G. Safonov, and R. Y. Chiang, “Next generation of tools for robust control,” in *Proc. 2004 IEEE Amer. Control Conf. (ACC)*, vol. 6, Jun. 2004, pp. 5612–5615.

[13] L. Bascetta and P. Rocco, “Revising the Robust-Control Design for Rigid Robot Manipulators,” *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 180–187, Feb. 2010.

[14] G. Battistelli, D. Selvi, and A. Tesi, "Robust Switching Control: Stability Analysis and Application to Active Disturbance Attenuation," *IEEE Trans. Autom. Control*, vol. 62, no. 12, pp. 6369–6376, Dec. 2017.

[15] R. Bellman, *Dynamic Programming.* Princeton University Press, NJ, USA, 1957.

[16] J. Bendtsen, K. Trangbaek, and J. Stoustrup, "Plug-and-Play Control – Modifying Control Systems Online," *IEEE Trans. Control Syst. Techn.*, vol. 21, no. 1, pp. 79–93, Jan. 2013.

[17] J. Bendtsen and K. Trangbaek, "Closed-Loop Identification for Control of Linear Parameter Varying Systems," *Asian J. Control*, vol. 16, no. 1, pp. 40–49, 2014.

[18] J. Bendtsen and K. Trangbaek, "Multiple Model Adaptive Control Using Dual Youla-Kucera Factorisation," *IFAC Proc. Vol.*, vol. 45, no. 13, pp. 63–68, 2012, 7th IFAC Symp. Robust Control Design.

[19] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *Proc. European Control Conf. (ECC)*, Jul. 2015, pp. 2501–2506.

[20] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Int. Conf. Neural Inform. Proc. Syst. (NIPS)*, Long Beach, CA, USA, 2017, pp. 909–919.

[21] D. P. Bertsekas, *Dynamic programming and optimal control*, 2. Athena Scientific Belmont, MA, 1995, vol. 1.

[22] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*, ser. Optimization and neural computation series. Athena Scientific, 1996, vol. 3.

[23] F. D. Bianchi and R. S. Sánchez-Peña, "Interpolation for gain-scheduled control with guarantees," *Automatica*, vol. 47, no. 1, pp. 239–243, 2011.

[24] F. D. Bianchi and R. S. S. Peña, "A novel design approach for switched LPV controllers," *Int. J. Control*, vol. 83, no. 8, pp. 1710–1717, 2010.

[25] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1371–1394.

[26] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR lightweight robot arm – a new reference platform for robotics research and manufacturing," in *41st International Symposium on Robotics and ROBOTIK 2010 (6th German Conference on Robotics)*, Jun. 2010, pp. 1–8.

[27] F. Blanchini, S. Miani, and F. Mesquine, "A Separation Principle for Linear Switching Systems and Parametrization of All Stabilizing Controllers," *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 279–292, Feb. 2009.

[28] S. P. Boyd and C. H. Barratt, *Linear Controller Design: Limits of Performance.* Englewood Cliffs, NJ: Prentice Hall, 1991.

[29] S. P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, ser. SIAM studies in applied mathematics. Philadelphia: Society for Industrial and Applied Mathematics, 1994, vol. 15.

[30] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Syst. Mag.*, vol. 26, no. 3, pp. 96–114, 2006.

[31] M. Brown and C. J. Harris, *Neurofuzzy adaptive modelling and control.* Prentice Hall, 1994.

[32] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 820–833, 2011.

[33] L. Buşoniu, D. Ernst, B. D. Schutter, and R. Babuška, "Online least-squares policy iteration for reinforcement learning control," in *Proc. 2010 Amer. Control Conf. (ACC)*, Jun. 2010, pp. 486–491.

[34] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators.* CRC press, 2010, vol. 39.

[35] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement Learning for Control: Performance, Stability, and Deep Approximators," *Annu. Rev. Control*, vol. 46, pp. 8–28, 2018.

[36] L. Buşoniu, A. Lazaric, M. Ghavamzadeh, R. Munos, R. Babuška, and B. De Schutter, "Least-squares methods for policy iteration," in *Reinforcement Learning: state-of-the-art*, ser. Adaptation, Learning, and Optimization, M. Wiering, M. van Otterlo, M. Wiering, and M. van Otterlo, Eds., vol. 12, Heidelberg, Germany: Springer, 2012, pp. 75–109.

[37] S. Calinon, "Learning from demonstration (programming by demonstration)," in *Encyclopedia of Robotics*, M. H. Ang, O. Khatib, and B. Siciliano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 1–8.

[38] S. Calinon, *Robot programming by demonstration.* Lausanne, Switzerland: EPFL Press, 2009.

[39] S. Calinon, P. Kormushev, and D. G. Caldwell, "Compliant skills acquisition and multi-optima policy search with EM-based reinforcement learning," *Robot. Auton. Syst.*, vol. 61, no. 4, pp. 369–379, 2013.

[40] D. Carnevale, S. Galeani, M. Sassano, and A. Astolfi, "Robust hybrid estimation and rejection of multi-frequency signals," *Int. J. Adapt. Control Sig. Process.*, vol. 30, no. 12, pp. 1649–1673, 2016.

[41] M. Cole, P. Keogh, and C. Burrows, "Adaptive-Q control of vibration due to unknown disturbances in rotor/magnetic bearing systems," in *Proc. IEEE Int. Conf. Control Applic.*, 2000, pp. 965–970.

[42] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in MATLAB.* Berlin Heidelberg: Springer, 2011.

[43] Y. Cui, T. Matsubara, and K. Sugimoto, "Kernel dynamic policy programming: Applicable reinforcement learning to robot systems with high dimensional states," *Neural Netw.*, vol. 94, pp. 13–23, 2017.

[44] D. M. Dawson, Z. Qu, F. L. Lewis, and J. F. Dorsey, "Robust control for the tracking of robot motion," *Int. J. Control*, vol. 52, no. 3, pp. 581–595, 1990.

[45] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, Feb. 2015.

[46] M. P. Deisenroth, G. Neumann, and J. Peters, “A survey on policy search for robotics,” *Found. Trends Robot.*, vol. 2, no. 1–2, pp. 1–142, 2013.

[47] M. Deniša, A. Gams, A. Ude, and T. Petrič, “Learning compliant movement primitives through demonstration and statistical generalization,” *IEEE/ASME Trans. Mech.*, vol. 21, no. 5, pp. 2581–2594, Oct. 2016.

[48] C. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties.* New York, NY: Academic Press, Inc., 1975.

[49] P. B. genannt Dohmann, “Implementation of a Trajectory Learning and Flexible Roll-out Module Based on Dynamic Movement Primitives on the KUKA LWR 4+,” Ingenieurspraxis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2016.

[50] S. G. Douma, P. M. J. V. den Hof, and O. H. Bosgra, “Controller tuning freedom under plant identification uncertainty: double Youla beats gap in robust stability,” *Automatica*, vol. 39, no. 2, pp. 325–333, 2003.

[51] G. Dullerud and F. Paganini, *A Course in Robust Control Theory: A Convex Approach.* New York, NY: Springer-Verlag, 2000.

[52] A. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor, “Regularized policy iteration with nonparametric function spaces,” *J. Mach. Learn. Res.*, vol. 17, no. 139, pp. 1–66, 2016.

[53] J. A. Farrell and M. M. Polycarpou, *Adaptive approximation based control: Unifying neural, fuzzy and traditional adaptive approximation approaches.* Hoboken, NJ: John Wiley & Sons, 2006, vol. 48.

[54] F. Ferraguti, C. Secchi, and C. Fantuzzi, “A tank-based approach to impedance control with variable stiffness,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 4948–4953.

[55] N. Figueroa and A. Billard, “A physically-consistent bayesian non-parametric mixture model for dynamical system learning,” in *Proc. 2nd Conf. Robot Learn. (CoRL)*, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87, Oct. 2018, pp. 927–946.

[56] S. A. Fjerdingen and E. Kyrkjebø, “Safe reinforcement learning for continuous spaces through lyapunov-constrained behavior,” *11th Scandinavian Conf. Artificial Intell.*, pp. 70–79, 2011.

[57] S. R. Friedrich and M. Buss, “A robust stability approach to robot reinforcement learning based on a parameterization of stabilizing controllers,” in *2017 IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3365–3372.

[58] S. R. Friedrich and M. Buss, “A Simple Architecture for Arbitrary Interpolation of State Feedback,” *IEEE Contr. Syst. Lett.*, vol. 3, no. 2, pp. 469–474, Apr. 2019.

[59] S. R. Friedrich and M. Buss, “Direct adaptive-Q control for online performance enhancement of switching linear systems,” in *Proc. 55th IEEE Conf. Decision Control (CDC)*, Dec. 2016, pp. 6375–6381.

[60] S. R. Friedrich and M. Buss, “Parameterizing robust manipulator controllers under approximate inverse dynamics: A double-Youla approach,” *Int. J. Robust Nonlinear Control*, vol. 29, pp. 5137–5163, 15 2019.

[61] S. R. Friedrich, M. Schreibauer, and M. Buss, “Least-squares policy iteration algorithms for robotics: Online, continuous, and automatic,” *Eng. Appl. Art. Intell.*, vol. 83, pp. 72–84, Aug. 2019.

[62] L. C. Fu, “Robust adaptive decentralized control of robot manipulators,” *IEEE Trans. Autom. Control*, vol. 37, no. 1, pp. 106–110, Jan. 1992.

[63] L. Furieri, Y. Zheng, A. Papachristodoulou, and M. Kamgarpour, “An Input–Output Parametrization of Stabilizing Controllers: Amidst Youla and System Level Synthesis,” *IEEE Control Syst. Lett.*, vol. 3, no. 4, pp. 1014–1019, Oct. 2019.

[64] A. Gams, B. Nemec, A. Ijspeert, and A. Ude, “Coupling movement primitives: Interaction with the environment and bimanual tasks,” *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 816–830, Aug. 2014.

[65] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. P. How, “A tutorial on linear function approximators for dynamic programming and reinforcement learning,” *Found. Trends Mach. Learn.*, vol. 6, no. 4, pp. 375–451, 2013.

[66] J. H. Gillula and C. Tomlin, “Guaranteed safe online learning via reachability: Tracking a ground target using a quadrotor,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 2723–2730.

[67] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 2.1*, `http://cvxr.com/cvx`, Mar. 2014.

[68] E. Gribovskaya, A. Kheddar, and A. Billard, “Motion learning and adaptive impedance for robot control during physical interaction with humans,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 4326–4332.

[69] W. M. Grimm, “Robot non-linearity bounds evaluation techniques for robust control,” *Int. J. Adaptive Contr. Signal Proc.*, vol. 4, no. 6, pp. 501–522, 1990.

[70] V. Gullapalli, J. A. Franklin, and H. Benbrahim, “Acquiring robot skills via reinforcement learning,” *IEEE Control Syst. Mag.*, vol. 14, no. 1, pp. 13–24, Feb. 1994.

[71] F. Hansen, G. Franklin, and R. Kosut, “Closed-loop identification via the fractional representation: Experiment design,” in *1989 Amer. Control Conf. (ACC)*, Jun. 1989, pp. 1422–1427.

[72] R. Hayat and M. Buss, “Model identification for robot manipulators using regressor-free adaptive control,” in *2016 UKACC 11th Int. Conf. on Control*, Aug. 2016, pp. 1–7.

[73] R. Hayat, M. Leibold, and M. Buss, “Robust-Adaptive Controller Design for Robot Manipulators Using the $\mathcal{H}_\infty$ Approach,” *IEEE Access*, vol. 6, pp. 51 626–51 639, 2018.

[74] R. Hayat, “Model-Free Robust-Adaptive Controller Design and Identification for Robot Manipulators,” Dissertation, Technische Universität München, München, 2019.

[75] S. S. Haykin, *Adaptive Filter Theory*, 5th. Pearson Education Limited, 2014.

[76] M. K. Helwa, A. Heins, and A. P. Schoellig, “Provably Robust Learning-Based Approach for High-Accuracy Tracking Control of Lagrangian Systems,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1587–1594, Apr. 2019.

[77] B. Hencey and A. G. Alleyne, “A Robust Controller Interpolation Design Technique,” *IEEE Trans. Control Syst. Tech.*, vol. 18, no. 1, pp. 1–10, Jan. 2010.

[78] B. Hencey and A. Alleyne, “Robust controller interpolation via parameterization,” in *ASME 2008 Dynamic Syst. Control Conf.*, Amer. Soc. Mech. Eng., 2008, pp. 1237–1244.

[79] J. Hespanha, P. Santesso, and G. Stewart, “Optimal controller initialization for switching between stabilizing controllers,” in *Proc. 46th IEEE Conf. Decision Control (CDC)*, Dec. 2007, pp. 5634–5639.

[80] J. P. Hespanha and A. S. Morse, “Switching between stabilizing controllers,” *Automatica*, vol. 38, no. 11, pp. 1905–1917, 2002.

[81] J. P. Hespanha, *Linear Systems Theory.* Princeton, NJ, USA: Princeton University Press, 2009.

[82] J. P. Hespanha, D. Liberzon, and A. Morse, “Overcoming the limitations of adaptive control by means of logic-based switching,” *Syst. Control Lett.*, vol. 49, no. 1, pp. 49–65, 2003, Adaptive Control.

[83] H. Hirsch-Kreinsen, E. Kubach, R. Stark, G. von Wichert, S. Hornung, L. Hubrecht, J. Sedlmeir, and S. Steglich, *Key themes of Industrie 4.0 – Research and development needs for successful implementation of Industrie 4.0*, Plattform Industrie 4.0, Ed., `https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/acatech-keythemes-industrie-4-0.pdf?__blob=publicationFile&v=6`, Accessed: 2019-12-23, Sep. 2019.

[84] N. Hogan, “Impedance control: An approach to manipulation: Parts I–III,” *ASME. J. Dyn. Sys., Meas., Control*, vol. 107, no. 1, Mar. 1985.

[85] P. Honeine, “Approximation errors of online sparsification criteria,” *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4700–4709, Sep. 2015.

[86] F. Hourfar, H. J. Bidgoly, B. Moshiri, K. Salahshoor, and A. Elkamel, “A reinforcement learning approach for waterflooding optimization in petroleum reservoirs,” *Eng. Appl. Art. Intell.*, vol. 77, pp. 98–116, 2019.

[87] R. Hu and P. C. Müller, “Position Control of Robots by Nonlinearity Estimation and Compensation: Theory and Experiments,” *J. Intell. Robot. Syst.*, vol. 20, no. 2, pp. 195–209, 1997.

[88] Z. Huang, X. Xu, H. He, J. Tan, and Z. Sun, “Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. PP, no. 99, pp. 1–12, 2017.

[89] K. Hunt, D. Sbarbaro, R. Żbikowski, and P. Gawthrop, “Neural networks for control systems–a survey,” *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[90] J. Hwangbo, *ROCK* example implementation*, `https://bitbucket.org/adrlab/c_rockstar`, Accessed: 2015-11-16.

[91] J. Hwangbo, C. Gehring, H. Sommer, R. Siegwart, and J. Buchli, "ROCK* – efficient black-box optimization for policy learning," in *IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2014, pp. 535–540.

[92] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. 2002 IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, May 2002, pp. 1398–1403.

[93] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.

[94] J. Imae, L. Irlicht, G. Obinata, and J. B. Moore, "Enhancing optimal controllers via techniques from robust and adaptive control," *Int. J. Adaptive Contr. Signal Proc.*, vol. 6, no. 5, pp. 413–429, 1992.

[95] L. Irlicht and J. Moore, "Functional Learning in Optimal Non-linear Control," in *Proc. 1991 IEEE Amer. Control Conf. (ACC)*, Jun. 1991, pp. 2137–2142.

[96] J. Y. Ishihara and R. M. Sales, "Doubly coprime factorizations related to any stabilizing controllers in state space," *Automatica*, vol. 35, no. 9, pp. 1573–1577, Sep. 1999.

[97] H. S. Jakab and L. Csató, "Sparse approximations to value functions in reinforcement learning," in *Artificial Neural Networks*, P. Koprinkova-Hristova, V. Mladenov, and N. K. Kasabov, Eds., Cham: Springer, 2015, pp. 295–314.

[98] M. Jin, S. H. Kang, P. H. Chang, and J. Lee, "Robust Control of Robot Manipulators Using Inclusive and Enhanced Time Delay Control," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 5, pp. 2141–2152, Oct. 2017.

[99] M. Jin and J. Lavaei, "Control-theoretic analysis of smoothness for stability-certified reinforcement learning," in *Proc. 57th IEEE Conf. Decision Control (CDC)*, Miami Beach, FL, Dec. 2018, pp. 6840–6847.

[100] T. Jung and D. Polani, "Kernelizing LSPE(λ)," in *2007 IEEE Int. Symp. ADPRL*, Apr. 2007, pp. 338–345.

[101] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artificial Intell. Res.*, vol. 4, pp. 237–285, 1996.

[102] K. Kalyanam and T.-C. Tsao, "Experimental study of adaptive-Q control for disk drive track-following servo problem," *IEEE/ASME Trans. Mechatronics*, vol. 15, no. 3, pp. 480–491, Jun. 2010.

[103] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Pearson Education Limited, 2002.

[104] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish, "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annual Rev. Control*, vol. 36, no. 1, pp. 42–59, 2012.

[105] S. M. Khansari-Zadeh, K. Kronander, and A. Billard, "Modeling robot discrete movements with state-varying stiffness and damping: A framework for integrated motion generation and impedance control," in *Proc. Robotics: Sci. Syst.*, Berkeley, USA, Jul. 2014.

[106] S. M. Khansari-Zadeh and A. Billard, “Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions,” *Robot. Autonom. Syst.*, vol. 62, no. 6, pp. 752–765, 2014.

[107] S. Khansari-Zadeh and A. Billard, “Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models,” *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[108] S. M. Khansari-Zadeh and O. Khatib, “Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors,” *Auton. Robots*, vol. 41, no. 1, pp. 45–69, Jan. 2017.

[109] P. P. Khargonekar and M. A. Dahleh, “Advancing systems and control research in the era of ML and AI,” *Annu. Rev. Control*, vol. 45, pp. 1–4, 2018.

[110] P. P. Khargonekar and D. Shim, “Maximally robust state-feedback controllers for stabilization of plants with normalized right coprime factor uncertainty,” *Syst. Control Lett.*, vol. 22, no. 1, pp. 1–4, 1994.

[111] B. K. Kim, H.-T. Choi, W. K. Chung, and I. H. Suh, “Analysis and design of robust motion controllers in the unified framework,” *ASME J. Dyn. Syst., Meas., Control*, vol. 124, no. 2, pp. 313–320, 2002.

[112] J. H. Kim, S. Hur, and Y. Oh, “L_1 Robustness of Computed Torque Method for Robot Manipulators,” in *Proc. 2018 IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 330–335.

[113] J. H. Kim, S.-M. Hur, and Y. Oh, “Performance analysis for bounded persistent disturbances in PD/PID-controlled robotic systems with its experimental demonstrations,” *Int. J. Control*, vol. 91, no. 3, pp. 688–705, 2018.

[114] M. J. Kim, Y. Choi, and W. K. Chung, “Bringing Nonlinear $\mathcal{H}_\infty$ Optimality to Robot Controllers,” *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 682–698, Jun. 2015.

[115] J. Kober, D. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *Int. J. Robot. Res.*, no. 11, pp. 1238–1274, 2013.

[116] J. P. Kolhe, M. Shaheed, T. Chandar, and S. Talole, “Robust control of robot manipulators based on uncertainty and disturbance estimation,” *Int. J. Robust Nonlin. Control*, vol. 23, no. 1, pp. 104–122, Oct. 2013.

[117] P. Kormushev, S. Calinon, and D. G. Caldwell, “Reinforcement learning in robotics: Applications and real-world challenges,” *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

[118] R. M. Kretchmar, P. M. Young, C. W. Anderson, D. C. Hittle, M. L. Anderson, and C. C. Delnero, “Robust reinforcement learning control with static and dynamic stability,” *Int. J. Robust Nonlinear Control*, vol. 11, no. 15, pp. 1469–1500, 2001.

[119] F. Kreutmayr, “Learning-Q Control with Reinforcement Learning,” Bachelor’s Thesis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2016.

[120] K. Kristinsson and G. A. Dumont, “System identification and control using genetic algorithms,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 22, no. 5, pp. 1033–1046, Sep. 1992.

[121] K. Kronander and A. Billard, “Learning Compliant Manipulation through Kinesthetic and Tactile Human-Robot Interaction,” *IEEE Trans. Haptics*, vol. 7, no. 3, pp. 367–380, 2013.

[122] K. Kronander and A. Billard, “Passive interaction control with dynamical systems,” *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 106–113, Jan. 2016.

[123] K. Kronander and A. Billard, “Stability Considerations for Variable Impedance Control,” *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1298–1305, Oct. 2016.

[124] V. Kučera, “Stability of Discrete Linear Feedback Systems,” *IFAC Proc. Vol.*, vol. 8, no. 1, pp. 573–578, Aug. 1975.

[125] V. Kučera, *Discrete Linear Control: The Polynomial Equation Approach.* John Wiley & Sons, New York, London, Sydney, 1979.

[126] M. Kuipers and P. Ioannou, “Multiple Model Adaptive Control With Mixing,” *IEEE Trans. Autom. Control*, vol. 55, no. 8, pp. 1822–1836, Aug. 2010.

[127] T. R. Kurfess, Ed., *Robotics and Automation Handbook.* Boca Raton, FL: CRC press, 2005.

[128] M. G. Lagoudakis and R. Parr, “Least-squares policy iteration,” *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, 2003.

[129] E. Lavretsky and K. Wise, *Robust and Adaptive Control With Aerospace Applications*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag, 2013.

[130] W. S. Lee, B. D. O. Anderson, R. L. Kosut, and I. M. Y. Mareels, “A new approach to adaptive robust control,” *Int. J. Adaptive Contr. Signal Proc.*, vol. 7, no. 3, pp. 183–211, 1993.

[131] D. J. Leith and W. E. Leithead, “Survey of gain-scheduling analysis and design,” *Int. J. Control*, vol. 73, no. 11, pp. 1001–1025, 2000.

[132] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot Manipulator Control: Theory and Practice.* Boca Raton, FL: CRC Press, 2003.

[133] F. L. Lewis and D. Vrabie, “Reinforcement learning and adaptive dynamic programming for feedback control,” *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, Mar. 2009.

[134] F. Lewis, D. Vrabie, and K. Vamvoudakis, “Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers,” *IEEE Control Syst. Mag.*, vol. 32, no. 6, pp. 76–105, Dec. 2012.

[135] F. L. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, ser. IEEE Press Ser. Comput. Intell. 1. John Wiley & Sons, Inc., 2013, vol. 17.

[136] Y. Li, K. P. Tee, R. Yan, W. L. Chan, and Y. Wu, “A Framework of Human–Robot Coordination Based on Game Theory and Policy Iteration,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1408–1418, Dec. 2016.

[137] D. Liberzon, *Switching in Systems and Control.* Boston, MA, USA: Birkhäuser, 2003.

[138] H. Lin and P. Antsaklis, “Stability and Stabilizability of Switched Linear Systems: A Survey of Recent Results,” *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 308–322, Feb. 2009.

[139] W. Liu, J. C. Principe, and S. Haykin, *Kernel adaptive filtering: a comprehensive introduction.* John Wiley & Sons, 2011, vol. 57.

[140] Z. Liu, J. Liu, and W. He, “Dynamic modeling and vibration control for a nonlinear 3-dimensional flexible manipulator,” *Int. J. Robust Nonlinear Control*, vol. 28, no. 13, pp. 3927–3945, 2018.

[141] J. Löfberg, “YALMIP: A Toolbox for Modeling and Optimization in MATLAB,” in *Proc. CACSD Conference*, Taipei, Taiwan, 2004.

[142] H. Luo, M. Krueger, T. Koenings, S. X. Ding, S. Dominic, and X. Yang, “Real-time optimization of automatic control systems with application to BLDC motor test rig,” *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4306–4314, May 2017.

[143] J. Ma and W. B. Powell, “A convergent recursive least squares approximate policy iteration algorithm for multi-dimensional markov decision process with continuous state and action spaces,” in *2009 IEEE Int. Symp. ADPRL*, Mar. 2009, pp. 66–73.

[144] J. Ma and W. B. Powell, “Convergence analysis of kernel-based on-policy approximate policy iteration algorithms for markov decision processes with continuous, multidimensional states and actions,” in *Dept. Oper. Res. Financial Eng., Princeton Univ.*, 2010.

[145] K. Maier, “Reset Map Investigation for Switching Adaptive-Q Control,” Bachelor’s Thesis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2016.

[146] N. Matni, A. Proutiere, R. A., and S. Tu, *From self-tuning regulators to reinforcement learning and back again*, 2019 IEEE 58th Conf. Decision Control (CDC), Tutorial Session and Paper, Dec. 2019.

[147] M. A. McEver, D. G. Cole, and R. L. Clark, “Adaptive feedback control of optical jitter using Q-parameterization,” *Optical Eng.*, vol. 43, no. 4, pp. 904–910, 2004.

[148] D. McFarlane and K. Glover, *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions*, ser. Lecture Notes in Control and Information Sciences (LNCIS). Berlin, Germany: Springer-Verlag, 1990, vol. 138.

[149] A. Megretski and A. Rantzer, “System analysis via integral quadratic constraints,” *IEEE Trans. Autom. Control*, vol. 42, no. 6, pp. 819–830, Jun. 1997.

[150] L. Mirkin, “Intermittent Redesign of Analog Controllers via the Youla Parameter,” *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1838–1851, Apr. 2017.

[151] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2016, pp. 1928–1937.

[152] J. B. Moore, K. Glover, and A. Telford, “All stabilizing controllers as frequency-shaped state estimate feedback,” *IEEE Trans. Autom. Control*, vol. 35, no. 2, pp. 203–208, Feb. 1990.

[153] J. B. Moore and T. T. Tay, “Adaptive control within the class of stabilizing controllers for a time-varying nominal plant,” *Int. J. Control*, vol. 50, no. 1, pp. 33–53, 1989.

[154] T. Mori, M. Howard, and S. Vijayakumar, “Model-free apprenticeship learning for transfer of human impedance behaviour,” in *2011 11th IEEE-RAS Int. Conf. Humanoid Robots*, Oct. 2011, pp. 239–246.

[155] J. Nakanishi, J. A. Farrell, and S. Schaal, “Composite adaptive control with locally weighted statistical learning,” *Neural Netw.*, vol. 18, no. 1, pp. 71–90, 2005.

[156] F. Navas, I. Mahtout, V. Milanés, and F. Nashashibi, “Youla-Kucera control structures for switching,” in *IEEE Conf. Control Techn. Appl. (CCTA)*, Copenhagen, Denmark, Aug. 2018.

[157] B. Nemec, N. Likar, A. Gams, and A. Ude, “Human robot cooperation with compliance adaptation along the motion trajectory,” *Auton. Robots*, vol. 42, no. 5, pp. 1023–1035, Jun. 2018.

[158] A. Y. Ng and H. J. Kim, “Stable adaptive control with online learning,” in *Proc. 17th Int. Conf. Neural Inform. Process. Syst. (NIPS)*, Vancouver, British Columbia, Canada: MIT Press, 2004, pp. 977–984.

[159] D. Nguyen-Tuong and J. Peters, “Model learning for robot control: A survey,” *Cognitive Process.*, vol. 12, no. 4, pp. 319–340, 2011.

[160] H. Niemann, “Dual Youla parameterisation,” *IEE Proc. – Control Theory Appl.*, vol. 150, 493–497(4), 5 Sep. 2003.

[161] H. Niemann and J. Stoustrup, “An architecture for implementation of multivariable controllers,” in *Proc. 1999 IEEE Amer. Control Conf. (ACC)*, vol. 6, 1999, pp. 4029–4033.

[162] H. Niemann, “A controller architecture with anti-windup,” *IEEE Contr. Syst. Lett.*, vol. 4, pp. 139–144, 1 2020.

[163] H. Niemann, “An architecture for controller parameterization,” in *Proc. 2018 Amer. Control Conf. (ACC)*, IEEE, 2018, pp. 418–423.

[164] H. Niemann, J. Stoustrup, and R. B. Abrahamsen, “Switching between multivariable controllers,” *Optim. Control Appl. Meth.*, vol. 25, no. 2, pp. 51–66, 2004.

[165] M. Noll, “Experimental Evaluation of an Active Variable Impedance Control Architecture,” Bachelor’s Thesis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2017.

[166] M. Noll, “Interfacing Robot Hardware with a Matlab/Simulink Controller Design Toolbox,” Ingenieurspraxis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2016.

[167] M. Olbrich, “Low-Complexity Linear Parameter-Varying Identification and Control of a Robot Manipulator,” Bachelor’s Thesis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2016.

[168] T. Oomen, “Advanced Motion Control for Precision Mechatronics: Control, Identification, and Learning of Complex Systems,” *IEEJ J. Ind. Appl.*, vol. 7, no. 2, pp. 127–140, 2018.

[169] D. Ormoneit and Ś. Sen, “Kernel-based reinforcement learning,” *Mach. Learn.*, vol. 49, no. 2, pp. 161–178, Nov. 2002.

[170] A. Packard, G. Balas, R. Liu, and J.-Y. Shin, “Results on worst-case performance assessment,” in *Proc. 2000 IEEE Amer. Control Conf. (ACC)*, vol. 4, 2000, pp. 2425–2427.

[171] I. Palunko, P. Donner, M. Buss, and S. Hirche, “Cooperative suspended object manipulation using reinforcement learning and energy-based control,” in *2014 IEEE/RSJ Int. Conf. Intell. Robot Syst. (IROS)*, Sep. 2014, pp. 885–891.

[172] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro, “A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots,” in *2013 IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 4896–4901.

[173] Z. Pan, H. Chen, and X. You, “Support vector machine with orthogonal legendre kernel,” in *2012 Int. Conf. Wavelet Anal. Pattern Recogn.*, Jul. 2012, pp. 125–130.

[174] Y. P. Pane, S. P. Nageshrao, J. Kober, and R. Babuška, “Reinforcement learning based compensation methods for robot manipulators,” *Eng. Appl. Art. Intell.*, vol. 78, pp. 236–247, 2019.

[175] S. S. Pchelkin, A. S. Shiriaev, A. Robertsson, L. B. Freidovich, S. A. Kolyubin, L. V. Paramonov, and S. V. Gusev, “On Orbital Stabilization for Industrial Manipulators: Case Study in Evaluating Performances of Modified PD+ and Inverse Dynamics Controllers,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 1, pp. 101–117, Jan. 2017.

[176] T. J. Perkins and A. G. Barto, “Lyapunov design for safe reinforcement learning,” *J. Mach. Learn. Res.*, vol. 3, pp. 803–832, Mar. 2003.

[177] I. R. Petersen and R. Tempo, “Robust control of uncertain systems: Classical results and recent developments,” *Automatica*, vol. 50, no. 5, pp. 1315–1335, 2014.

[178] G. Pillonetto, F. Dinuzzo, T. Chen, G. D. Nicolao, and L. Ljung, “Kernel methods in system identification, machine learning and function estimation: A survey,” *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[179] A. S. Polydoros and L. Nalpantidis, “Survey of model-based reinforcement learning: Applications on robotics,” *J. Intell. Robot. Syst.*, vol. 86, no. 2, pp. 153–173, May 2017.

[180] W. B. Powell and J. Ma, “A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multidimensional continuous applications,” *J. Control Theory Appl.*, vol. 9, no. 3, pp. 336–352, 2011.

[181] W. B. Powell, “Perspectives of approximate dynamic programming,” *Ann. Oper. Res.*, vol. 241, no. 1, pp. 319–356, 2016.

[182] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, 1994.

[183] Z. Qu and J. Dorsey, “Robust tracking control of robots by a linear feedback law,” *IEEE Trans. Autom. Control*, vol. 36, no. 9, pp. 1081–1084, Sep. 1991.

[184] B. Rasmussen and A. Alleyne, “Stable gain-scheduling on endogenous signals,” in *Proc. 2005 Amer. Control Conf. (ACC)*, Portland, OR, USA: IEEE, 2005, pp. 1895–1900.

[185] B. P. Rasmussen and Y. J. Chang, “Stable controller interpolation and controller switching for LPV systems,” *ASME. J. Dyn. Sys., Meas., Control.*, vol. 132, no. 1, 2010.

[186] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, ser. Adaptive computation and machine learning. MIT Press, 2006.

[187] B. Recht, “A tour of reinforcement learning: The view from continuous control,” *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 2, no. 1, pp. 253–279, 2019.

[188] J. Rey, K. Kronander, F. Farshidian, J. Buchli, and A. Billard, “Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies,” *Auton. Robots*, May 2017.

[189] C. Richard, J. C. M. Bermudez, and P. Honeine, “Online prediction of time series data with kernels,” *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, 2009.

[190] J. W. Roberts, I. R. Manchester, and R. Tedrake, “Feedback controller parameterizations for reinforcement learning,” in *2011 IEEE Symp. Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, IEEE, 2011, pp. 310–317.

[191] P. Rocco, “Stability of PID control for industrial robot arms,” *IEEE Trans Robot. Autom.*, vol. 12, no. 4, pp. 606–614, Aug. 1996.

[192] M. T. Rosenstein and A. G. Barto, “Reinforcement learning with supervision by a stable controller,” in *Proc. 2004 Amer. Control Conf. (ACC)*, vol. 5, Jun. 2004, pp. 4517–4522.

[193] M. Rösler, “Nonlinear Adaptive Filtering for Learning-Q Control,” Forschungspraxis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2017.

[194] M. Rösler, “Variable Impedance Feedback Motion Planning Skills by Internal Model Parameterization,” M.S. thesis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2019.

[195] M. A. Rotea and P. P. Khargonekar, “H^2-optimal control with an H^∞-constraint: the state feedback case,” *Automatica*, vol. 27, no. 2, pp. 307–316, 1991.

[196] M. C. Rotkowitz, “Parametrization of all stabilizing controllers subject to any structural constraint,” in *Proc. 49th IEEE Conf. Decision Control (CDC)*, 2010, pp. 108–113.

[197] W. J. Rugh and J. S. Shamma, “Research on Gain Scheduling,” *Automatica*, vol. 36, no. 10, pp. 1401–1425, Oct. 2000.

[198] F. R. P. Safaei, J. P. Hespanha, and G. Stewart, “On controller initialization in multivariable switching systems,” *Automatica*, vol. 48, no. 12, pp. 3157–3165, 2012.

[199] H. G. Sage, M. F. De Mathelin, and E. Ostertag, “Robust control of robot manipulators: A survey,” *Int. J. Control*, vol. 72, no. 16, pp. 1498–1522, 1999.

[200] E. Sariyildiz, H. Sekiguchi, T. Nozaki, B. Ugurlu, and K. Ohnishi, “A Stability Analysis for the Acceleration-Based Robust Position Control of Robot Manipulators via Disturbance Observer,” *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 5, pp. 2369–2378, Oct. 2018.

[201] S. Schaal and C. G. Atkeson, “Learning control in robotics,” *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 20–29, Jun. 2010.

[202] C. Scherer and S. Weiland, *Linear Matrix Inequalities in Control*, ser. Lecture Notes 2. Delft, The Netherlands: Dutch Institute for Systems and Control, 2000, vol. 3.

[203] M. M. Schill and M. Buss, “Kinematic Trajectory Planning for Dynamically Unconstrained Nonprehensile Joints,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 728–734, 2018.

[204] B. Schölkopf and A. J. Smola, *Learning with Kernels: support vector machines, regularization, optimization, and beyond*, ser. Adaptive computation and machine learning series. MIT Press, 2002.

[205] M. Schreibauer, “Performance Enhancement of Stabilizing Controllers by Reinforcement Learning,” M.S. thesis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2018.

[206] M. Schreibauer, “Towards Reinforcement Learning in Q-Parametrizations,” Forschungspraxis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2016.

[207] G. Schreiber, A. Stemmer, and R. Bischoff, “The Fast Research Interface for the KUKA Lightweight Robot,” in *ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*, IEEE, 2010, pp. 15–21.

[208] D. Schröder and M. Buss, *Intelligente Verfahren, Identifikation und Regelung nichtlinearer Systeme*, 2nd ed. Berlin, Heidelberg: Springer Vieweg, 2017.

[209] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[210] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*. London, UK: Springer Science & Business Media, 2012.

[211] H. Seraji, “Decentralized adaptive control of manipulators: theory, simulation, and experimentation,” *IEEE Trans. Robot. Autom.*, vol. 5, no. 2, pp. 183–201, Apr. 1989.

[212] J. S. Shamma and M. Athans, “Gain scheduling: potential hazards and possible remedies,” *IEEE Control Syst.*, vol. 12, no. 3, pp. 101–107, Jun. 1992.

[213] J. S. Shamma, “An overview of LPV systems,” in *Control of Linear Parameter Varying Systems with Applications*, J. Mohammadpour and C. W. Scherer, Eds. Boston, MA: Springer, 2012, pp. 3–26.

[214] Y. Shavit, N. Figueroa, S. S. M. Salehian, and A. Billard, “Learning augmented joint-space task-oriented dynamical systems: A linear parameter varying and synergetic control approach,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2718–2725, Jul. 2018.

[215] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, UK: Springer, 2009.

[216] O. Sigaud, C. Salaün, and V. Padois, “On-line regression algorithms for learning mechanical models of robots: A survey,” *Robot. Auton. Syst.*, vol. 59, no. 12, pp. 1115–1129, 2011.

[217] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design.* New York, NY, USA: John Wiley & Sons, Inc., 2005.

[218] M. Spong and M. Vidyasagar, “Robust linear compensator design for nonlinear robotic control,” *IEEE J. Robot. Autom.*, vol. 3, no. 4, pp. 345–351, Aug. 1987.

[219] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control.* New York, NY: John Wiley and Sons Inc., 2006.

[220] G. E. Stewart and G. A. Dumont, “Finite horizon based switching between stabilizing controllers,” in *2006 Amer. Control Conf. (ACC)*, Jun. 2006, pp. 1550–1556.

[221] D. J. Stilwell and W. J. Rugh, “Stability preserving interpolation methods for the synthesis of gain scheduled controllers,” *Automatica*, vol. 36, no. 5, pp. 665–671, 2000.

[222] D. J. Stilwell, “J-Q interpolation for gain scheduled controllers,” in *Proc. 38th IEEE Conf. Decision Control (CDC)*, vol. 1, Dec. 1999, pp. 749–754.

[223] J. Stoustrup and H. Niemann, “Active fault diagnosis by controller modification,” *Int. J. Syst. Sci.*, vol. 41, no. 8, pp. 925–936, 2010.

[224] F. Stulp and O. Sigaud, “Many regression algorithms, one unified model: A review,” *Neural Netw.*, vol. 69, pp. 60–79, 2015.

[225] F. Stulp and O. Sigaud, “Robot Skill Learning: From Reinforcement Learning to Evolution Strategies,” *Paladyn, J. Behavioral Robot.*, vol. 4, Issue 1, pp. 49–61, Sep. 2013.

[226] T. Sugie, T. Yoshikawa, and T. Ono, “Robust controller design for robot manipulators,” *J. Dyn. Syst., Meas., Control*, vol. 110, no. 1, pp. 94–96, 1988.

[227] R. S. Sutton, A. G. Barto, and R. J. Williams, “Reinforcement learning is direct adaptive optimal control,” *IEEE Control Syst.*, vol. 12, no. 2, pp. 19–22, Apr. 1992.

[228] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT press, Cambridge, 1998.

[229] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Proc. 12th Int. Conf. Neural Inform. Proc. Syst. (NIPS)*, 2000, pp. 1057–1063.

[230] J. A. K. Suykens, J. Vandewalle, and B. L. R. D. Moor, “NLq theory: Checking and imposing stability of recurrent neural networks for nonlinear modeling,” *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2682–2691, Nov. 1997.

[231] T.-T. Tay, I. Mareels, and J. B. Moore, *High Performance Control.* Boston, MA, USA: Birkhäuser, 1998.

[232] T.-T. Tay and J. B. Moore, “Performance enhancement of two-degree-of-freedom controllers via adaptive techniques,” *Int. J. Adapt. Control Signal Process.*, vol. 4, no. 1, pp. 69–84, 1990.

[233] T.-T. Tay and J. Moore, “Enhancement of fixed controllers via adaptive-Q disturbance estimate feedback,” *Automatica*, vol. 27, no. 1, pp. 39–53, 1991.

[234] T. Tay, J. Moore, and R. Horowitz, “Indirect adaptive techniques for fixed controller performance enhancement,” *Int. J. Control*, vol. 50, no. 5, pp. 1941–1959, 1989.

[235] G. Taylor and R. Parr, “Kernelized value function approximation for reinforcement learning,” in *Proc. 26th Int. Conf. Mach. Learn. (ICML)*, ACM, 2009, pp. 1017–1024.

[236] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems.* London, UK: Springer, 2004.

[237] S. C. Thomsen, H. Niemann, and N. K. Poulsen, “Robust stability in constrained predictive control through the Youla parameterisations,” *Int. J. Control*, vol. 84, no. 4, pp. 653–664, 2011.

[238] D. Tolić and I. Palunko, “Learning suboptimal broadcasting intervals in multi-agent systems,” in *Proc. 17th IFAC World Congress*, Seoul, Korea, vol. 50, 2017, pp. 4144–4149.

[239] K. Trangbaek, J. Stoustrup, and J. Bendtsen, “Stable Controller Reconfiguration through Terminal Connections,” in *Proc. 17th IFAC World Congress*, Seoul, Korea, vol. 41, 2008, pp. 331–335.

[240] A. Tremba, G. Calafiore, F. Dabbene, E. Gryazina, B. Polyak, P. Shcherbakov, and R. Tempo, “RACT: Randomized Algorithms Control Toolbox for MATLAB,” in *Proc. 17th IFAC World Congress*, Seoul, Korea, vol. 41, 2008, pp. 390–395.

[241] M. C. Turner, G. Herrmann, and I. Postlethwaite, “Discrete-time anti-windup: Part 1 – Stability and performance,” in *European Control Conf. (ECC)*, Sep. 2003, pp. 473–478.

[242] K. Tziortziotis, K. Vlachos, and K. Blekas, “Reinforcement learning-based motion planning of a triangular floating platform under environmental disturbances,” in *2016 24th Mediterranean Conf. Control Autom. (MED)*, Jun. 2016, pp. 1014–1019.

[243] A. Valibeygi and R. A. de Callafon, “Adaptive regulation for disturbance rejection with uncertain parameters: Stability analysis,” in *2017 Amer. Control Conf. (ACC)*, May 2017, pp. 1493–1498.

[244] B. Vanderborght, “Humans and robots working together,” *IEEE Robot. Autom. Mag.*, vol. 25, no. 2, pp. 4–4, Jun. 2018.

[245] M. B. Vankadari, K. Das, C. Shinde, and S. Kumar, “A reinforcement learning approach for autonomous control and landing of a quadrotor,” in *2018 Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, Jun. 2018, pp. 676–683.

[246] M. Vidyasagar, *Control Systems Synthesis: A Factorization Approach.* Cambride, MA: MIT Press, 1985.

[247] J. Vinogradska, B. Bischoff, and J. Peters, “Approximate value iteration based on numerical quadrature,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1330–1337, Apr. 2018.

[248] P. Vlachas, “Development of a Learning Algorithm for Convergent Systems,” Forschungspraxis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2015.

[249] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*, ser. Control Engineering Series. London, UK: IET, 2013, vol. 81.

[250] J. Wang, X. Xu, D. Liu, Z. Sun, and Q. Chen, "Self-learning cruise control using kernel-based least squares policy iteration," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 3, pp. 1078–1087, May 2014.

[251] Z. Wang, I. Mareels, and J. Moore, "Adaptive disturbance rejection," in *Proc. 30th IEEE Conf. Decision Control (CDC)*, Dec. 1991, pp. 2836–2841.

[252] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.

[253] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-the-Art*, ser. Adaptation, Learning, and Optimization. Springer, 2012, vol. 12.

[254] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 229–256, 1992.

[255] C. C. de Wit, S. B, and B. G, Eds., *Theory of Robot Control.* London, UK: Springer, 1996.

[256] W. Xie and T. Eisaka, "Design of LPV control systems based on Youla parameterisation," *IEE Proc. Control Theory Appl.*, vol. 151, no. 4, pp. 465–472, Jul. 2004.

[257] W. Xie, "H_∞ performance controller parameterisation of linear switching plants under uncontrolled switching," *Int. J. Control*, vol. 89, no. 5, pp. 871–878, 2016.

[258] X. Xu, Z. Hou, C. Lian, and H. He, "Online learning control using adaptive critic designs with sparse kernel machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 5, pp. 762–775, May 2013.

[259] X. Xu, C. Lian, L. Zuo, and H. He, "Kernel-based approximate dynamic programming for real-time online learning control: An experimental study," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 1, pp. 146–156, Jan. 2014.

[260] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 973–992, 2007.

[261] X. Xu, C. Lian, J. Wang, H.-G. He, and D. Hu, "Actor–critic reinforcement learning for autonomous control of unmanned ground vehicles," *in Science Robotics*, p. 42, 2016.

[262] S. Yahyaa and B. Manderick, "Knowledge gradient for online reinforcement learning," in *Agents and Artificial Intelligence. ICAART 2014. LNCS, vol 8946.*, B. Duval, J. van den Herik, S. Loiseau, and J. Filipe, Eds., Cham: Springer, 2015, pp. 103–118.

[263] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, "Human-like adaptation of force and impedance in stable and unstable interactions," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 918–930, Oct. 2011.

[264] C. Yang, C. Zeng, C. Fang, W. He, and Z. Li, "A DMPs-based framework for robot learning and generalization of humanlike variable impedance skills," *IEEE/ASME Trans. Mech.*, vol. 23, no. 3, pp. 1193–1203, Jun. 2018.

[265] D. C. Youla, H. Jabr, and J. J. Bongiorno, "Modern wiener-hopf design of optimal controllers–part ii: The multivariable case," *IEEE Trans. Autom. Control*, vol. 21, no. 3, pp. 319–338, Jun. 1976.

[266] P. M. Young, M. P. Newlin, and J. C. Doyle, "Mu analysis with real parametric uncertainty," in *Proc. 30th IEEE Conf. Decision Control (CDC)*, Dec. 1991, 1251–1256 vol.2.

[267] C. Zeng, C. Yang, J. Zhong, and J. Zhang, “Encoding multiple sensor data for robotic learning skills from multimodal demonstration,” *IEEE Access*, vol. 7, pp. 145 604–145 613, 2019.

[268] Z. Zhang, “Development of a Kernel Adaptive Filter with Bounded Energy Gain,” Forschungspraxis, Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, 2018.

[269] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control.* Upper Saddle River, NJ, USA: Prentice-Hall, 1996.