Optimal Co-Design of Hardware Architecture and Control Configuration for Interacting Systems

Michiel Haemers

Doctoral dissertation submitted to obtain the academic degrees of Doctor of Electromechanical Engineering Technology (UGent) and Doctor of Applied Engineering (UAntwerpen)



Supervisors

Prof. Kurt Stockman, PhD* - Prof. Stijn Derammelaere, PhD** - Prof. Clara-Mihaela Ionescu, PhD*

- * Department of Electromechanical, Systems and Metal Engineering Faculty of Engineering and Architecture, Ghent University
- ** Department of Electromechanics Faculty of Applied Engineering, University of Antwerp

June 2021





Optimal Co-Design of Hardware Architecture and Control Configuration for Interacting Systems

Michiel Haemers

Doctoral dissertation submitted to obtain the academic degrees of

Doctor of Electromechanical Engineering Technology (UGent) and Doctor of Applied Engineering (UAntwerpen)

Supervisors

Prof. Kurt Stockman, PhD* - Prof. Stijn Derammelaere, PhD** - Prof. Clara-Mihaela Ionescu, PhD*

- * Department of Electromechanical, Systems and Metal Engineering Faculty of Engineering and Architecture, Ghent University
- ** Department of Electromechanics
- Faculty of Applied Engineering, University of Antwerp June 2021





ISBN 978-94-6355-491-6 NUR 978 Wettelijk depot: D/2021/10.500/39

Members of the Examination Board

Chair

Prof. Em. Hendrik Van Landeghem, PhD, Ghent University

Other members entitled to vote

Prof. Jeroen De Kooning, PhD, Ghent University Wilm Decré, PhD, KU Leuven Bruno Depraetere, PhD, Flanders Make Niko Nevaranta, PhD, LUT University, Finland Prof. Jan Steckel, PhD, University of Antwerp

Supervisors

Prof. Kurt Stockman, PhD, Ghent University Prof. Stijn Derammelaere, PhD, University of Antwerp Prof. Clara-Mihaela Ionescu, PhD, Ghent University

Preface

Met het indienen en verdedigen van dit werk komt een einde aan bijna vijf jaar doctoraatsonderzoek aan UGent campus Kortrijk. Het verliep zeker niet altijd van een leien dakje, maar ik heb doorheen deze periode enorm veel bijgeleerd en ik ben dan ook zeer blij deze uitdaging tot een goed einde te kunnen brengen. Ik wil daarbij enkele mensen bedanken die dit werk mede mogelijk gemaakt hebben.

In eerste instantie mijn promotoren: prof. dr. ing. Stijn Derammelaere, prof. dr. ing. Kurt Stockman en prof. dr. ir. Clara Ionescu. Zij waren inhoudelijk mijn grootste inspiratiebronnen en ik wil hen dan ook uitgebreid bedanken voor hun tijd die in mij geïnvesteerd werd om bruikbare inzichten en feedback te voorzien. Na elk overleg kreeg ik weer de nodige 'drive' om verder te gaan en dit werk te vervolledigen. In het bijzonder wil ik Stijn bedanken. Sinds de lessenreeksen die ik als student van hem kreeg, wist hij zijn passie voor regeltechniek op mij over te brengen. Als begeleider van de legendarische Duitsland- en China-reizen leerde ik hem ook op persoonlijk vlak kennen en kwam ik te weten dat hij mijn mening deelt rond flauwe woordspelingen: "Beter een paar mindere te veel dan één goeie te weinig." Bovendien was hij ook de eerste die indertijd in mij een potentiële doctor zag. Bedankt voor je wijze raad, je politieke duidingen en je Antwerpse gidsbeurten die ons leidden van schrale café's tot heuse kasteelfeesten.

Daarnaast wil ik ook alle collega's van UGent campus Kortrijk en UAntwerpen campus Groenenborger bedanken voor de aangename werksfeer. De bewijzen van deze uitstekende collegialiteit zijn onder meer onze deelnames aan internationale bedrijfssportspelen, teamdagen en de vele aangename (lunch)pauzes. Ik wil daarbij een speciaal dankwoord richten aan Japser, die na vier jaar samen gestudeerd te hebben ook mijn collega bij UGent werd en ongeveer gelijktijdig zijn doctoraat uitvoerde. We hebben vele leuke momenten beleefd: van conferenties, studiedagen, summer schools en Doctoral Schools opleidingen tot evenementen van de studentenclub Moeder EM die we gezamenlijk erfden van Stijn. Ook bedankt aan Simon Houwen voor de gesprekken over onze gezamenlijke interesse voor muziek en om mij grafische skills bij te brengen. Bedankt aan Simon De Boever voor de verbouwtips en de 'maker' praatjes. Ik kom met veel plezier klinken op je afgewerkte watermolen ergens rond het jaar 2070. Ook bedankt aan David voor de zwemsessies, Stan om mijn Heulse bureaumaatje te zijn, Foeke voor de vele GA-tips en Florian voor de regeltechnische inzichten. Ook bedankt aan collega's Bart, Pieter, José, Simon en Heinz voor hun praktische hulp rond het construeren van de labo-opstelling. Much appreciated!

Daarenboven wil ook de leden van de examencommissie bedanken voor hun tijd en moeite om mijn thesis grondig te lezen en mij van bruikbare feedback te voorzien. Ook veel dank aan de KU Leuven en Flanders Make medewerkers van het ROCSIS SBO-project waar mijn doctoraatsonderzoek onderdeel van was.

Op persoonlijk vlak wil ik uiteraard mijn vrouw Kimberly bedanken voor de steun doorheen mijn doctoraat. Vooral naar het einde toe zat ik vaak buiten de 'kantooruren' nog aan mijn bureau en kwamen de huishoudelijke taken vooral op haar schouders terecht. Bedankt voor je begrip, je steun en om er steeds voor mij te zijn.

Ook bedankt aan mijn zoontjes Mats en Kamiel voor de nodige afleidingen. Het wandelen naar school, het spelen in de tuin of het bouwen van een autobaan waren welgekomen ontspanningsmomenten tijdens het drukke schrijfwerk. Ze zijn nog te jong om het te beseffen, maar zij waren voor mij een grote bron van motivatie om dit werk te volbrengen. Daarnaast wil ik ook mijn ouders bedanken voor de steun en studiekansen, mijn zus omdat ze steeds oprecht vroeg: "Maar wat heb je nu eigenlijk écht gedaan vandaag?", en de rest van mijn familie en vriendengroep voor de sociale ondersteuning.

Ruim een jaar geleden begon de druk gevoelig te stijgen om mijn doctoraatsonderzoek af te ronden en deze thesis op te stellen. Rond datzelfde moment werden we geteisterd door een wereldwijde COVID-19 pandemie. Het wegvallen van de sociale contacten met collega's en vrienden en het verplichte thuiswerk vergemakkelijkten het schrijfproces niet. Op dit moment van schrijven zijn we in België drie golven en bijna 25.000 betreurenswaardige COVID-19 sterfgevallen verder, maar gaan de cijfers wel de goede kant uit door de doeltreffende vaccinatiecampagne. Ik hoop dan ook dat de parallel zich verder zet en dat het einde van mijn doctoraatsonderzoek ook het einde van de pandemie en geldende maatregelen mag betekenen.

Nogmaals een welgemeende bedankt en veel leesplezier!

Michiel Haemers, mei 2021

Contents

Pr	eface		iii
Co	Contents ix		
Su	mma	ry	xi
Sa	menv	atting	XV
Lis	st of A	Abbreviations	xix
Lis	st of S	Symbols	xxi
1	Intr	oduction	1
	1.1	Context	1
	1.2	Conventional Development Methodologies	8
	1.3	Motivation	11
	1.4	Research Questions	14
	1.5	Dissertation Outline	15
	1.6	Publications and Acknowledgment	16
2 Background on Co-Design		kground on Co-Design	19
	2.1	Multi-Domain Optimization	19
	2.2	Background on Optimizations	21
	2.3	Co-Design Strategies	24
	2.4	State-of-the-art and Applications	26
	2.5	Chapter Conclusions	38
3	Mul	ti-Domain Optimization Workflow	39
	3.1	Optimization Workflow Overview	40
	3.2	Optimization Objectives	43
	3.3	Optimization Algorithm Requirements	43
	3.4	Optimization Algorithm Selection	44
		3.4.1 Gradient-Based Optimization Approach	44

		3.4.2 Derivative-Free Optimization Approach	47 5 6
	25	5.4.5 Conclusions on the Optimization Algorithm Selection)0 50
	3.3 2.6	Pareto Front)8 50
	5.0	2.6.1 Continuous LTI State Space Modeling)9 50
		3.0.1 Continuous L11 State-Space Modeling)9 (1
		3.0.2 Continuous State-Space Observer Modeling)U ()
		3.6.3 Continuous LPV and LIV State-Space Modeling)2 < 4
	27	3.6.4 Illustrative L11 case: Mechanical Synchronization Setup .	۶4 دە
	3.7		<u>)</u> 8
4	Har	dware Architecture Optimization 6	59
	4.1	Actuator & Sensor Placement	70
	4.2	Actuator & Sensor Selection	73
		4.2.1 Mechanical Synchronization Setup	74
	4.3	Open-Loop Analysis	76
		4.3.1 Mechanical Synchronization Setup	79
	4.4	Chapter Conclusions	34
5	Con	trol Configuration Optimization	35
	5.1	Control Architecture Optimization	36
		5.1.1 Background on Control Architecture Topologies 8	36
		5.1.2 Decentralized/Distributed Control Architecture Features .	39
		5.1.3 Implementation in Co-Design Methodology) 4
		5.1.4 Mechanical Synchronization Setup) 4
	5.2	Controller Tuning Optimization) 7
		5.2.1 Mechanical Synchronization Setup) 9
	5.3	Closed-Loop State-Space Methodology 10)1
		5.3.1 Background and Motivation)1
		5.3.2 Discrete State-Space Process and Observer Representation 10)3
		5.3.3 Discrete State-Space PID Representation)4
		5.3.4 Extensive Control Structure Possibilities)7
		5.3.5 SS_{CL} Methodology Workflow)9
		5.3.6 Mechanical Synchronization Setup 11	18
	5.4	Chapter Conclusions	22
6	Har	dware and Control Co-Design 12	25
	6.1	Co-Design Optimization Properties	26
		6.1.1 Objective Function	26
		6.1.2 Design Parameters	26
		6.1.3 Constraints	27
	6.2	Genetic Algorithm Implementation	28
	6.3	LTI Case: Mechanical Synchronization Setup	31
		6.3.1 Objective Function	31

С	Acti	ve Car	Suspension Setup Validation Measurements	217
B	Mec	hanical	Synchronization Case Optimization Responses	201
A	Resu	ulting S	S _{CL} Algorithms	195
8	Gen 8.1 8.2	eral Co Genera Recon	al Conclusions and Future Work al Conclusions	191 191 193
0	r.5	Chapte		170
	75	7.4.5 Chapte	Comparison to Existing Controller Tuning Methods	100
		7.4.2	Comparison to Existing Controllor Tuning Matheds	182
		7.4.1	Model-Based Pareto Optimizations	180
	7.4	Result	s and Measurements	180
	7 4	7.3.4	Genetic Algorithm Implementation	179
		7.3.3	Constraints	178
		7.3.2	Design Parameters	175
	-	7.3.1	Objective Function	175
	7.3	Hardw	vare and Control Co-Design	175
		7.2.3	Road Profile	173
		7.2.1 7.2.2	State-Space Model Identification	170
	1.2		Cal Suspension Lao Setup	10/
	7 2	7.1.2	Active Suspension Lab Satur	165
		7.1.1	Semi-Active and Adaptive Suspension Systems	164
	7.1	Introd	uction on Active Suspension Systems	163
7	Case	e: Activ	e Car Suspension Setup	163
				1
	6.5	Chapte	er Conclusions	161
		6.4.7	Case Conclusions	159
		0.4.3 646	Genetic Algorithm Implementation and Results	150
		6.4.4	Constraints	156
		6.4.3	Design Parameters	155
		6.4.2	Objective Function	155
		6.4.1	Model Properties	154
	6.4	LPV C	Case: Composite Plate	153
		6.3.7	Objective Function Surface Plots	149
		6.3.6	Comparison to Existing Controller Tuning Methods	141
		6.3.5	Optimization Results	137
		634	Genetic Algorithm Implementation	132
		0.3.2 6.3.3	Constraints	132
		632	Design Parameters	132

Summary

Today, consumer goods have become a combination of components originating from different technological sub-domains empowering a common purpose. A typical omnipresent example is a passenger car. At the beginning of the twentieth century, a car still consisted of rudimentary, isolated, and mainly mechanical parts. In contrast, today's cars can be labeled as highly interconnected systems in which many different parts from varying technologies cooperate. Moreover, this car evolution is not yet complete because, in the future, probably fully automated cars will roam the streets that will bring the passengers to the desired destination based on immense data streams from a vast range of sensors.

This evolution can also be found in industrial machines, for which it is also essential that the various components work together to achieve optimum system performance. Usually, there are several (teams of) engineers that take care of a specific aspect of the system design. The problem with this approach is that these different engineers look at the problem from a different perspective and also have different objectives in mind. For example, a mechanical engineer will be less concerned with the optimal control parameters that the control engineer must determine. Nevertheless, the mechanical engineer's choices can have a significant impact on what performance the control engineer can achieve with the control settings. In other words, there is a lack of understanding of the influence of the individual choices regarding mechanical design and control design on the maximum achievable system performance. This is the main research question to which this work provides an answer.

In this dissertation, a co-design methodology is presented that allows the simultaneous optimization of both the hardware architecture and the control configuration for systems consisting of multiple interacting subsystems. With the optimization of the hardware architecture, both the optimal type and location of actuators and sensors are determined. As part of the control architecture configuration, the optimal architecture of the control loops and the optimal controller tuning settings are determined. An extensive literature review shows that the current most advanced co-design only considers the hardware architecture and the tuning of the controllers. Therefore, simultaneously considering the control loop architecture and different types of actuators and sensors is a substantial addition to the current state-of-the-art. The problem must first be transformed into a computationally tractable formulation in order to perform the optimization within a reasonable time. This work describes how the different design options of the co-design can be linked to several design parameters to be optimized. It shows that the optimization problem has a discontinuous character with both continuous and discrete design parameters and non-linear objectives and constraints. Multiple (possibly conflicting) objectives, such as following a reference trajectory, energy consumption, or vibration reduction, can be taken into account simultaneously. The applicable constraints can be related to, e.g., total implementation cost, maximum output of the actuator, or tolerances on mechanical movements. Due to the nature of the optimization problem, the constraints, and the discontinuities that are present, only a limited number of optimization algorithms are applicable, of which a Genetic Algorithm (GA) is chosen as the most suitable. However, the optimization only becomes feasible if the time required to execute the algorithm remains limited.

This work proposes some extensions to enable the computationally efficient optimization and to solve the complex optimization problem within a reasonable time. In the first place, by applying an open-loop analysis. During this analysis, the feasibility of different hardware configurations can be determined efficiently without the need to calculate a closed-loop response. This information then ensures that no time is lost in calculating the system response of these predetermined infeasible hardware configurations during the main optimization routine. A second extension is to describe the system dynamics with various interconnected control loops as one single state-space system representation. As a result, the system response based on external inputs and disturbances can be computed much faster compared to existing methods used to describe extensive control loops. These additions significantly reduce the computation time required to obtain a solution. The result of the proposed co-design method is a Pareto analysis that provides a clear understanding of the inevitable trade-off between the total implementation cost and the maximum system performance to be achieved.

The proposed method is not tailored for one particular application or a specific optimization problem but is instead a generic method that can be used on a wide range of applications (e.g., on mechatronic, electrical, or thermal systems). Some practical cases are also considered in this work. For example, the hardware and control co-design methodology is performed on a mechanical synchronization example in which coupled inertias must follow a reference trajectory as closely as possible under the influence of varying load torques. A second application involves a composite plate for which the optimal location of collocated actuator and sensor pairs and the associated control must be determined to reduce occurring vibrations. Finally, the hardware and control co-design methodology is implemented on a setup representing a car with an active suspension. A downscaled lab setup is built to validate the model-based optimization results. The setup is based on a well-known general active car suspension model and allows to emulate the dynamic behavior of a car with an active suspension system that gives the possibility to test different hardware architectures and control configurations. This also involves applying disturbance signals that act on the central platform and correspond to the forces encountered when driving over a given realistic road profile. The aim is to design the hardware and control aspects in such a way that the active elements in the suspension reduce the arising accelerations of the central platform as much as possible to maximize the driver's comfort.

These applications highlight the flexibility and effectiveness of the proposed co-design method. Moreover, the results are also validated by comparing the obtained controller settings with existing controller tuning methods. In addition, it is also verified through objective function surface plots that none of the surrounding solutions shows a better performance than the obtained optimization result. The model-based active car suspension optimization results are also validated with measurements on the lab setup.

The results show that with the proposed methodology, a design engineer is able to successfully perform the co-design of the hardware architecture and the controller configuration. In doing so, the resulting Pareto analysis provides a very valuable insight into the relationship between the maximum achievable system performance and the total implementation cost to obtain an optimal system design.

Samenvatting

Tegenwoordig zijn consumptiegoederen meer en meer een samenstelling van onderdelen die uit verschillende technologische subdomeinen afkomstig zijn maar die een gezamenlijk doel bekrachtigen. Een typisch voorbeeld daarvan is een auto. Aan het begin van de twintigste eeuw bestond een auto nog uit rudimentaire, geïsoleerde en hoofdzakelijk mechanische onderdelen. Daarentegen kunnen auto's tegenwoordig bestempeld worden als sterk geïnterconnecteerde systemen waarbij heel wat verschillende onderdelen uit variërende technologieën samenwerken. Bovendien is deze evolutie van de auto nog niet volbracht, want in de toekomst zullen er waarschijnlijk volledig geautomatiseerde auto's in het straatbeeld opduiken die de passagiers tot de gewenste bestemming zullen brengen op basis van immense datastromen uit een heel resem sensoren.

Deze evolutie is ook terug te vinden in industriële machines waarbij het voor dergelijke systemen belangrijk is dat de verschillende onderdelen samenwerken om een optimale systeemprestatie te behalen. Doorgaans zijn er verschillende (teams van) ingenieurs die een bepaald aspect van het systeemontwerp voor zich nemen. Het probleem daarbij is dat deze verschillende ingenieurs het probleem vanuit een andere invalshoek bekijken en daarbij ook andere doelstellingen voor ogen hebben. Zo zal bijvoorbeeld een mechanisch ingenieur in mindere mate wakker liggen van de optimale regelparameters die de regeltechnisch ingenieur moet bepalen. Maar de keuzes van de mechanisch ingenieur kunnen wel degelijk een grote invloed hebben op welke performantie de regeltechnisch ingenieur kan behalen met de regeltechnische instellingen. Er is dus met andere woorden een tekort aan inzicht in de invloed van de individuele keuzes inzake mechanisch ontwerp en regeltechnisch ontwerp op de maximaal te behalen systeemprestatie. Dat is dan ook de voornaamste onderzoeksvraag waar dit werk een antwoord op biedt.

In deze thesis wordt een methode toegelicht die het mogelijk maakt om gelijktijdig het co-ontwerp van zowel de hardware architectuur als de regeltechnische configuratie te optimaliseren voor systemen die bestaan uit meerdere interagerende subsystemen. Met de optimalisatie van de hardware architectuur worden zowel het optimale type als de optimale locatie van actuatoren en sensoren bepaald. Als onderdeel van de regeltechnische configuratie worden de optimale architectuur van de regellus alsook de optimale instellingen van de regelaars bepaald. Uit een uitgebreide literatuurstudie blijkt dat het huidige

Samenvatting

meest geavanceerde co-ontwerp enkel de hardware architectuur en de afstelling van de regelaars beschouwt. Het gelijktijdig overwegen van de architectuur van de regellus en verschillende types hardware betekenen dan ook substantiële toevoegingen aan de huidige state-of-the-art.

Het optimalisatieprobleem moet eerst getransformeerd worden naar een wiskundig te optimaliseren formulering om de effectieve optimalisatie binnen een redelijke tijd te kunnen uitvoeren. Dit werk beschrijft hoe de verschillende ontwerpmogelijkheden van het co-ontwerp aan verscheidene te optimaliseren ontwerpparameters gekoppeld kunnen worden. Hieruit blijkt dat het optimalisatieprobleem een discontinu karakter heeft met continue en discrete ontwerpparameters en niet-lineaire objectieven en restricties. Hierbij kunnen meerdere (conflicterende) objectieven zoals het volgen van een referentietraject, energieconsumptie, optredende vibraties, enz. gelijktijdig in rekening gebracht worden. De daarbij geldende restricties kunnen gerelateerd zijn aan bijvoorbeeld totale implementatiekost, maximale uitgang van de actuator of toleranties op mechanische bewegingen. Door de aard van het optimalisatieprobleem zijn slechts een beperkt aantal optimalisatie-algoritmes toepasbaar, waarvan een Genetisch Algoritme (GA) als het best passende gekozen wordt. De optimalisatie wordt echter pas haalbaar als de benodigde rekentijd om het algoritme uit te voeren beperkt blijft.

In dit werk worden enkele uitbreidingen toegelicht om een rekenkundig efficiënte optimalisatie mogelijk te maken en het complexe optimalisatieprobleem binnen een afzienbare tijd te volbrengen. Een eerste uitbreiding is een open-lus analyse. Tijdens deze analyse kan de werkbaarheid van de verschillende hardware configuraties op een efficiënte manier bepaald worden zonder dat een gesloten-lus responsie berekend hoeft te worden. Deze informatie zorgt er vervolgens voor dat er tijdens het optimaliseren geen tijd verloren gaat aan het uitrekenen van de systeemresponsie van deze vooraf bepaalde onhaalbare hardware configuraties. Een tweede uitbreiding bestaat erin om de systeemdynamiek met verschillende geïnterconnecteerde regellussen als één state-space systeemrepresentatie te beschrijven. De systeemresponsie op basis van externe ingangen en storingen kan hierdoor veel sneller uitgerekend worden in vergelijking met bestaande methodes die het toelaten om uitgebreide regellussen toe te passen. Deze toevoegingen zorgen voor een significante vermindering van de benodigde rekentijd. Het resultaat van de voorgestelde co-ontwerp werkwijze is een Pareto-analyse die een duidelijk inzicht verschaft in de onafwendbare afweging tussen de totale implementatiekost en de maximaal te behalen systeemprestatie.

De voorgestelde werkwijze is niet gericht op één bepaalde toepassing of een specifiek optimalisatieprobleem, maar is een algemeen toepasbare methode en kan worden gebruikt op een breed scala aan applicaties (bijvoorbeeld op mechatronische, elektrische of thermische systemen). In dit werk komen ook enkele praktische toepassingen aan bod. Zo wordt de werkwijze voor het hardware en controle co-ontwerp uitgevoerd op een mechanisch synchronisatievoorbeeld waarbij gekoppelde inerties een referentietraject zo goed mogelijk moeten volgen onder invloed van een variërend lastkoppel. Een tweede toepassing houdt een composiet plaat in waarbij de optimale locatie van actuator- en sensorparen en de daarbij horende controle bepaald moeten worden om optredende trillingen te reduceren. Finaal wordt de werkwijze voor het optimaal co-ontwerp van de hardware en controle uitgevoerd op een opstelling die een actieve auto-ophanging voorstelt. Om de modelgebaseerde optimalisatieresultaten te valideren, wordt voor deze applicatie een geschaalde labo-opstelling gebouwd die gebaseerd is op een welgekend algemeen model van een auto met een actieve ophanging. De laboopstelling laat toe om het dynamisch gedrag na te bootsen van een auto met een actief ophangingssysteem waarbij heel wat verschillende hardware architecturen en controle configuraties getest kunnen worden. Daarbij worden storingssignalen aangelegd die inwerken op het centraal platform en die overeenkomstig zijn met de krachten die optreden bij het rijden over een realistisch wegprofiel. Het is daarbij de bedoeling dat de hardware- en controle-aspecten zodanig ontworpen worden dat de actieve elementen in de ophanging de optredende acceleraties van het centrale platform zo veel mogelijk reduceren om het comfort van de chauffeur te maximaliseren.

Deze toepassingen onderstrepen de flexibiliteit en de effectiviteit van de voorgestelde co-ontwerp werkwijze. Bovendien worden de resultaten ook gevalideerd door de bekomen regelaarsafstellingen te vergelijken met bestaande methodes om een regelaar in te stellen. Daarnaast wordt door middel van een oppervlaktegrafiek ook gecontroleerd dat geen van de omliggende oplossingen een betere prestatie vertoont dan het bekomen optimalisatieresultaat. De optimalisatie-resultaten van de actieve auto-ophanging worden ook gevalideerd met metingen op de labo-opstelling.

De resultaten tonen aan dat een ontwerpingenieur met de voorgestelde werkwijze in staat is om het co-ontwerp van de hardware architectuur en de regeltechnische configuratie met succes uit te voeren. De resulterende Paretoanalyse voorziet daarbij een heel waardevol inzicht in hoe de maximaal te behalen systeemprestaties zich verhoudt tot de totale implementatiekost om een optimaal systeemontwerp te bekomen. xviii

List of Abbreviations

The following is a list of abbreviations that are used in this dissertation:

ACD	Adaptive Coordinate Descent
CBD	Causal Block Diagrams
CD	Coordinate Descent
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CO	Convex Optimization
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
DC	Direct Current
DIRECT	Dividing Rectangles
DMS	Direct Multisearch
DOF	Degrees Of Freedom
DS	Direct Search
EA	Evolutionary Algorithms
ES	Exhaustive Search
FRF	Frequency Response Function
FV	Fitness Value
GA	Genetic Algorithm
GPS	Global Positioning System
HEV	Hybrid Electric Vehicle
IID	Iterative and Incremental Development
IMC	Internal Model Control
IPM	Initial Population Matrix
ISE	Integral of Square Error
IT	Information Technology
LB	Lower Bounds
LP	Linear Programming
LPV	Linear Parameter-Varying
LQR	Linear-Quadratic Regulator
LSITS	Large Space Intelligent Truss Structure
LTI	Linear Time-Invariant
LTV	Linear Time-Varying

MEMS	Microelectromechanical Systems
MFC	Macro-Fiber Composite
MIMO	Multiple-Input Multiple-Output
MPC	Model Predictive Control
NASA	National Aeronautics and Space Administration
NCS	Networked Control System
NLP	Non-Linear Programming
PEM	Prediction Error Minimization
PID	Proportional, Integral, Derivative
PLC	Programmable Logic Controller
PM	Polynomial Models
PS	Pattern Search
PSO	Particle Swarm Optimization
QP	Quadratic Programming
RAM	Random Access Memory
RMS	Root Mean Square
SA	Simulated Annealing
SISO	Single-Input Single-Output
SO	Surrogate Optimization
SQP	Quadratic Sequential Programming
SRTM	Shuttle Radar Topography Mission
SS	State-Space
TF	Transfer Function
UAV	Unmanned Aerial Vehicle
UB	Upper bounds
VIA	Variable Impedance Actuation
XP	Extreme Programming

List of Symbols

The following is a list of the most important symbols that are used in this dissertation:

$\mathbb B$	Set of all binary number types
\mathbb{Z}	Set of all integer number types
\mathbb{R}	Set of all real number types
b_{act}	Actuator placement binaries
b_{sen}	Sensor placement binaries
i_{act}	Actuator selection integer
i_{sen}	Sensor selection integer
\mathbb{B}_{Θ}	Diagonal matrix grouping the actuator placement binaries
\mathbb{B}_{Γ}	Diagonal matrix grouping the sensor placement binaries
\mathbb{B} r	Vector grouping the control architecture binaries
\mathbb{Z}_{act}	Vector grouping the actuator selection integers
\mathbb{Z}_{sen}	Vector grouping the sensor selection integers
\mathbb{R}_{tuning}	Vector grouping the controller tuning parameters
I	Identity matrix
∇	Gradient
∇^2	Hessian
SS	State-space system representation
\mathbf{A}	State-space system matrix
в	State-space input matrix
С	State-space output matrix
D	State-space feedthrough/feedforward matrix
\mathbf{L}	State-space observer gain matrix
x	State-space state vector
У	State-space output vector
u	State-space input vector
e	Controller input
l	Number of available actuator inputs from input vector u
m	Number of outputs in the output vector y
n	Number of states in the state vector x
0	Number of PID controllers in the control structure

p	Number of possible actuator types
q	Number of possible sensor types
r	Number of reference trajectories
w	Number of possible different open-loop hardware configurations
Dist	External disturbance signal
Ref	External reference signal
K_p	PID control: proportional gain
T_i	PID control: integral gain
T_d	PID control: derivative gain
$c_1, c_2, \text{ and } c_3$	Discrete PID controller parameters
T_s	Sample time [s]
T_0	Simulation starting time $[s]$
T_m	Total simulation time $[s]$
F_s	Sampling frequency $[Hz]$
K	Number of simulation samples
k	Simulation sample number
θ	Angular displacement [rad]
θ	Pitch angle [rad]
z	Translational position $[m]$
\dot{z}	Translational velocity $[m/s]$
\ddot{z}	Translational acceleration $[m/s^2]$
J	Inertia $\left[kgm^2\right]$
f	Force $[N]$
T	Torque [Nm]
k	Torsional spring constant $[Nm/rad]$
k	Translational spring constant $[N/m]$
b	Torsional damping constant $[Nms/rad]$
b	Translational damping constant $[Ns/m]$
d	Disturbance signal
$\mathbf{C}_{\mathbf{M}}$	Controllability matrix
$\mathbf{O}_{\mathbf{M}}$	Observability matrix
\mathbf{Q}	LQR weights on state
\mathbf{R}	LQR weights on actuator effort
γ	Quantification of the closed-loop H_{∞} performance
ϕ_{PM}	Phase margin
ω	Frequency [Hz]
\angle	Angle
j	Imaginary unit satisfying $j^2 = -1$

The following is a list of sub- and superscripts that are commonly used in this dissertation and what they refer to:

x_{OL}	Open-loop
x_{CL}	Closed-loop
x_{pro}	Process
x_{obs}	Observer
\hat{x}	Estimated value
x	Amplitude
x_c	Continuous time-domain
x_d	Discrete time-domain
x_L	Load
x_1, x_2, \dots	Feedback loop to system input
x_{dec}	Decentralized control
x_{dis}	Distributed control
$x_a, x_b,$	Cascade control level
x_{ff}	Feedforward control
x_{sc}	Synchronizing control
x_{pre}	Predefined matrices remaining constant during calculations
x_{Act}	Actuator
x_{Sen}	Sensor
x_{fr}	Front-right
x_{fl}	Front-left
x_{rr}	Rear-right
x_{rl}	Rear-left

Chapter 1 Introduction

This first chapter starts with a brief overview of the history of control design. The evolution from primitive mechanical machines to complex, interconnected systems is mentioned. This is followed by an explanation of some common development methodologies to indicate what a conventional design cycle looks like for complex (industrial) systems. Next, some shortcomings in the current design of complex systems are pointed out while motivating the need for what is done in this PhD dissertation. Subsequently, the research questions of this work are briefly yet concisely addressed. After that, an overview is given of the different PhD dissertation chapters and how they relate to each other. Finally, a list is given of the scientific publications of the author of this PhD dissertation.

1.1 Context

Throughout the last centuries, society has undergone a real metamorphosis under the influence of four major industrial revolutions. The first industrial revolution started about 270 years ago in England, after which it spread to the rest of Europe at the beginning of the nineteenth century. This revolution started with the invention of a steam engine that ensured that people no longer depended on human resources, horsepower, watermills, or windmills to do labor. From 1765 onward, the steam engine was further improved by James Watt, causing significant changes in the textile industry. In 1788, he applied the Watts flyball governor (see Fig. 1.1), which is considered the first automatic control system. If the speed of the connected shaft increases, the balls swing out, which closes the valve until a balance is achieved between desired and the proportional gain of the linkage and valve. Although a centrifugal regulator was invented by Christiaan Huygens to be used in windmills, the adaptation of Watts was so important for the steam engine that Watts is often mistakenly labeled as the inventor of the centrifugal regulator. This centrifugal regulator controls the power transmission from a driving steam engine to a driven machine by controlling the admission of steam into the cylinders [1].



Figure 1.1: Overview of a centrifugal flyball governor (from [2])



Figure 1.2: Model of 'Spinning Jenny', the first mechanical multi-spindle spinning frame, as presented in the Museum of Early Industrialism, Wuppertal, Germany (from [3])

Fig. 1.2 shows a model of 'Spinning Jenny', the first machine-driven spinning frame invented by James Hargreaves [3]. A little later, Joseph-Marie Jacquard developed a fully automatic silk looming machine [4], shown in Fig. 1.3. By actuating these machines with a steam engine, textiles could be produced on a large scale. Small workshops were replaced by large factories, and labor-intensive craftsmanship was eventually changed for mass-producing machines. The first industrial revolution is also characterized by the introduction of cast iron. Under the influence of these new techniques, it was possible to produce more cheaply, which meant that more people could access more consumer goods and that the general standard of living gradually improved [5].



Figure 1.3: Overview of a Jacquard looming machine (from [4])

The second industrial revolution (also known as the technological revolution) took place from the second half of the 19th century to the First World War and was mainly characterized by the development of the Bessemer process in the 1860s, developed by Sir Henry Bessemer. This process is distinguished by a new kind of furnace that could convert molten crude iron into steel (see Fig. 1.4). This technique further evolved into the introduction of new technologies, particularly the internal combustion engine, the oil industry, new materials and substances (including alloys and chemicals), and communication technologies such as telegraph and radio. While the first industrial revolution focused on iron, steam technologies, and textile production, the second industrial revolution revolved around steel, railroads, electric machines, and chemicals [6]. During the second industrial revolution in the 20th century, the automotive industry took off, which has become one of the world's largest manufacturing industries by revenue. One of the best-known examples of the first cars produced on a large scale is Henry Ford's Model T, with Fig. 1.5 showing an example of a Model T from 1910. Henry Ford was also a pioneer in applying the principle of an assembly line [7].

From the 1950s, several new forms of communication were introduced, mainly due to the rise of the computer. This made it possible to consult information almost anywhere in the world and made it possible for companies to globalize. Also, significant steps were taken to further automate the production processes, including the introduction of the Programmable Logic Controller (PLC) [10]. As a result, the control of an industrial machine shifted from mainly mechanical control that was

Introduction



Figure 1.4: Sir Henry Bessemer's Bessemer converter, located in Sheffield (from [8])



Figure 1.5: Ford Model T automobile from 1910 (from [9])

hard to adapt to flexible electronic controllers, but the focus was still mainly on single-input single-output (SISO) control systems. From then on, coordination of production and logistics could be carried out worldwide, enabling large-scale economies. The switch from analog, electronic, and mechanical technology to digital forms mark this third industrial revolution.

The rise of the internet indicates the start of the digital revolution. As a result, the 'information society' era has dawned, with the acquisition and processing of information becoming more important than pure production. The processing of data is generating increasingly large capital flows worldwide. For some, a fourth industrial revolution is currently in full swing. This fourth industrial revolution is referred to as the 'smart industry' or 'industry 4.0' and stands for new automation trends and profound data acquisition, storage and exchange in industrial manufacturing techniques [11, 12].

Nowadays, consumer goods and industrial machines are more and more comprised of co-operating parts from different technological subdomains. Therefore, the design process of complex systems is rapidly shifting from small scale development of isolated systems to large-scale development of integrated systems [13]. Regarding large-scale control, particular attention is paid to multiple-input multipleoutput (MIMO) controllers. The different parts of a system consisting of multiple subsystems need to exchange data efficiently to obtain a high-performing assembly. This is why the term 'cyber-physical systems' (CPS) is used, in which the term 'cyber' refers to the computer-based algorithms that monitor and control (different processes of) the complete system. This controller will usually be implemented as software and monitors the system activity by inputting sensor data and then using actuators to control the physical parts based on appropriate decisions. This procedure results in the traditional view of a physical process controlled by a software component, also known as a feedback control system [14, 15]. The generic and most basic representation of a feedback control system (with 'cyber' and 'physical' parts) is shown in Fig. 1.6.



Figure 1.6: Generic representation of a feedback control system, showing the interaction between the 'cyber' and 'physical' parts (reproduced from [14])

A practical example of a feedback control system is the cruise control of a car, shown in Fig. 1.7. The car itself represents the physical parts that are actuated by the control inputs generated by a cruise controller. The sensor of this cruise controller is a tachometer that converts the wheel speed to the speed of the car. The actuator is the throttle, quantifying the power going to the car engine. Based on the current speed, the cruise controller software will determine to what extent the throttle needs to be excited to maintain the desired speed, regardless of weight, wind, slope, and other external factors [14].



Figure 1.7: Graphical representation of a car cruise controller feedback system (reproduced from [14])

The term 'embedded systems' is used for systems that can largely be regarded as standalone entities and that incorporate elements of control logic and real-world applications. Embedded systems are typically a single device, specifically designed to perform a limited number of tasks and often with limited resources. In contrast, cyber-physical systems (CPS) include many constituent systems and operates at a much larger scale, potentially including many embedded systems or other cyberphysical system elements. It is well-accepted that cyber-physical systems consist of a large number of interacting components and display some recurring characteristics that distinguish them from classic control systems, such as extensive 'cyber' components, a vast scale of operation, a hybrid discrete-continuous nature, high adaptability, and integration with multiple external systems [14].

These cyber-physical systems are already widespread within our society. Consider the example of a car. At the beginning of the 20th century, a car consisted of rudimentary, isolated, and mainly mechanical parts. However, this changed drastically with the industrial and Information Technology (IT) revolutions. Nowadays, these old-fashioned cars have evolved into versatile and inter-connected cyberphysical systems with features like parking assist, gesture and voice control, and regular updates of the operating system via the internet. In the field of car suspensions, major innovations have been implemented in recent decades, whereby passive spring-damper systems have been replaced by (semi) active components that can substantially improve the road handling and the driver's comfort. In Chapter 7, the findings of this PhD are validated on an active car suspension lab setup. The introduction of that chapter provides an overview of recent developments in the field of different types of (semi) active car suspensions.

The development of complex cyber-physical systems is far from finished. For example, the first forms of self-driving cars can already be seen in the public streetscape, where the car drives autonomously based on data coming from GPS, cameras, radars, lidars, and other sensors [16–18]. Fig. 1.8 gives an overview of the various sensors that apply to modern-day self-driving cars. It is believed that these self-driving cars will evolve even further in their autonomy into an extensive network of cars that continually exchange information to control the traffic flow and reduce congestions. For example, cars that approach an intersection may

1.1 Context

communicate to negotiate which one crosses first. The complexity of designing and operating such novel systems is evident, especially in the face of the necessary compatibility between the range of cars fielded by all the different car manufacturers [19].



Figure 1.8: Graphical overview of the typical sensors used in autonomous vehicles (from [20])

Other common examples of cyber-physical systems appear in telecommunication, healthcare, energy distribution, climate control, robotics, aerospace, etc. [21] The strong interaction between the physical parts and the controlling cyber units pose new challenges for the optimal design of fully integrated cyber-physical systems, as different interdisciplinary interests need to be covered through this interaction. That is why cyber-physical systems (CPS) have emerged in recent years as a prominent research topic with practical relevance and has been drawing increasingly more attention from different communities.

1.2 Conventional Development Methodologies

The exponential rate at which CPSs evolve and their increasing complexity pose new challenges to their design. To tackle the increasing complexity, engineers use a model-based systems engineering methodology [22]. With this method, a model of a (part of a) machine is formulated to closely mimic its behavior. In this way, requirements relating to engineering, design, verification, and validation can be carried out (at least partially) on a simulation [21]. This results in a substantial reduction of the time needed to go through an entire design cycle from concept to the later phases of the product life cycle.

There exist several development methodologies. The simplest is the so-called 'waterfall' method, in which the development is broken down into five consecutive steps: requirements, design, implementation, verification, and maintenance (see Fig. 1.9) [23]. With this method, all development phases are completed sequentially, so there is no simultaneous co-design of hardware and control.



Figure 1.9: The waterfall development methodology (reproduced from [23])
1.2 Conventional Development Methodologies

The V-model works similarly to the waterfall model but differs in that each step after the implementation is tested against the corresponding step prior to the implementation phase. A graphical overview of the V-model is shown below in Fig. 1.10. The main drawback of the V-model is that the total system design is established at the beginning of the development, so little or no variation is possible. Likewise, it is impossible to establish complex systems where components or interactions are not known in advance [24]. The control design in the shown V-cycle is applied in the implementation phase. Since interactions between different disciplines cannot be considered, a co-design of hardware and control cannot be applied efficiently using the V-cycle development methodology.



Figure 1.10: The V-model development methodology (reproduced from [25])

The spiral methodology uses mainly the same components as the waterfall method: requirements, design, implementation, and testing. The difference is that these are carried out cyclically, usually shifting the focus in one cycle to a particular part of the system design. In the case of one completed cycle, the requirements are re-examined, and it is decided whether the system design can be labeled as finished. For every completed cycle, additional features are added to the system design [26]. Fig. 1.11 shows the graphical representation of the spiral methodology. This development methodology can be used to perform the simultaneous co-design of hardware and control, initially taking into account the full interactions between the different subsystems of a system and then performing a co-design according to the specified objectives and constraints.



Figure 1.11: The spiral development methodology (reproduced from [25])

Other examples of model-based design methodologies are Iterative and Incremental Development (IID) [27], Scrum [28], and Extreme Programming (XP) [28]. These originally arose from software development and can only be applied to a limited extent in the model-based design of complex industrial machines. For more information, please refer to the accompanying references. The choice of the development methodology depends on factors such as size, complexity, flexibility, and need for innovation. It is important to remember that applying model-based methods allows the design cycle to be completed much quicker since testing and validating a model is nearly always faster than having to do the same on physical prototypes.

1.3 Motivation

The complete system design consists of many different interacting hardware and control design possibilities that each have an (unknown) influence on the overall system performance. In conventional development strategies, the hardware and control designs are treated sequentially and are therefore approached entirely separated [29]. This results in a so-called 'industrial design gap' between hardware engineering on the one hand and control engineering on the other hand. This industrial design gap is graphically displayed in Fig. 1.12.



Figure 1.12: Graphical representation of the industrial design gap between hardware design and control design, with properties typical for each design aspect

In conventional development strategies, mechanical engineers start with the design of a physical setup, corresponding with the structural objectives on, for instance, weight, inertia, and strength. Next, the system is divided into subsystems and the positions of the actuators and sensors in the system are determined. Correspondingly, the actuator and sensor types are selected. Subsequently, control engineers design a control system for the fixed physical structure based on the given inputs and outputs of actuators and sensors, satisfying a different set of objectives, such as settling time, reference tracking properties or robustness to disturbances. They also determine which control strategy to be used (e.g., PID, MPC,

LQR, H_{∞}) and which features the control scheme should contain. Based on all this, satisfactory controller parameters (= controller tuning) are then determined. This sequential design approach is intuitive to implement but suffers from several problems when applied to more sophisticated applications consisting of several subsystems.

First, there is usually a tight inter-dependency between the control configuration and the hardware architecture that is almost totally neglected when applying sequential design. For example, the ability to achieve acceptable control performance is affected by the location and type of sensors and actuators, but the control engineer cannot change it. Simply stated, the hardware can limit the controller design space and, hence, the optimal achievable control performance. In many cases, it is hard to predict the distinct impact of these design properties on the overall system performance. For example, it is often unclear what the effect is on the overall system performance when a specific actuator or sensor is left out or sized differently on a subsystem level. Therefore, the hardware engineers have to make assumptions regarding the control configuration, after which the control engineers have to stick to that hardware architecture. This sequential approach might not lead to the most efficient or optimal design. Since modern systems are becoming increasingly more complex and the resources are always constrained by cost and space, a resource-efficient design has become an increasingly important issue [30].

Second, with sequential design, incorporating the needs of both hardware and control engineers becomes even more complicated as the overall size of the system and the number of interconnected subsystems increase. This results in sub-optimal designs and considerable integration, test, and validation efforts. Therefore, there is a rising demand for systematic and comprehensive co-design methods to accomplish both the optimal hardware architecture and control configuration [30]. Regarding the system's hardware optimization, most literature only considers the mechanical design or geometry parameters (e.g., [31–35]). Choices on the location and types of actuators and sensors have, so far, been mainly dictated by practical experience rather than by an actual optimization procedure.

The idea of looking at process equipment design and control system design as an integrated problem definition is not new, as is clear from the following quote from a paper by Ziegler and Nichols from 1943: "In the application of automatic controllers, it is important to realize that controller and process form a unit; credit or discredit for results obtained are attributable to one as much as the other. A poor controller is often able to perform acceptably on a process that is easily controlled. The finest controller made, when applied to a miserably designed process, may not deliver the desired performance. True, on badly designed processes, advanced controllers are able to eke out better results than older models, but on these processes, there is a definite end point which can be approached by instrumentation and it falls short of perfection" [36]. Although considerable research has been done on the control/architecture co-design problem in recent years, it is far from being complete. For the co-design methods to really become powerful and practical for industrial applications, further research efforts are needed to extend and generalize current techniques.

This is also the case for the Flemish industry, dealing with systems consisting of interacting subsystems that exhibit complex dynamic behavior. The current industrial control approach for these systems is often a decentralized PID-like control that focuses on controlling each subsystem separately. This is mainly to manage the controller tuning complexity, which involves significant human interaction. Moreover, engineers prefer diagnosis on PID controllers directly linked to subsystems rather than PID controllers in which there is no direct physical connection. For individual subsystems, PID controllers are relatively easy and intuitive to tune. However, for systems consisting of interacting subsystems, PID controller tuning is tedious and time-consuming even for an experienced operator. Due to its limited degrees of freedom, decentralized PID control alone fails to address the complex behavior of systems consisting of multiple subsystems as a whole. Consequently, the achievable performance is limited and can no longer meet the continuously increasing demands.

This leads to a need for a methodology to optimize the total system, combining the dimensioning and selection of sensors, actuators, and corresponding control. More complicated controllers and additional sensors and actuators do not necessarily lead to an economic profit. The balance between the enhanced performance and the elevated costs (more sensors, actuators, and a more complex control architecture) must be right. Hence, when considering a more complicated machine design, a systematic analysis and optimization of this interplay must be performed. Additionally, a way must be found to visualize the optimization results to provide the end-user with insights into the interplay between different design aspects.

The interplay between enhanced performance and elevated costs turns the control configuration design into a nontrivial task. Despite the strong industrial relevance and interest, there are currently no tools to simultaneously optimize the hardware and control for systems of interacting subsystems. Control design tools assume a prefixed control architecture: they assume given sensors and actuators and tune or optimize the controller parameters for a selected control and hardware architecture. The only option for a control engineer in co-designing the hardware and control is to try different configurations exhaustively and carefully analyze the resulting performances. How to address, in a (more) automatic way, multiple topologies with a large variety in the component types and numbers remains an open question [37, 38]. However, the lack of tools that support hardware and controller co-design is a critical problem that hinders the practical application of co-design methodologies. Therefore, there is a need to develop a toolchain that integrates the co-design framework and also bridges the gap between the separate controller and hardware architecture design tools [30].

Therefore, this PhD research aims to establish a generally applicable methodology to optimize both the control design and hardware design simultaneously. In doing so, multiple (conflicting) objectives and various constraints have to be taken into account. One difficulty of the control and hardware architecture codesign is integrating both entities in a manageable form for optimization purposes. Therefore, the interplay between the controller and the hardware architecture design spaces needs to be studied and identified, and specific characteristics on both sides should be exploited to formulate the problem in a mathematical way. This mathematical representation should allow the designer to efficiently apply a suitable optimization algorithm to obtain the results in a reasonable time. Additionally, it should be investigated how system-level analysis techniques can be used so that a priori insights can be obtained to effectively exclude pre-determined infeasible combinations of hardware and control components. Finally, a visualization strategy must be developed to present the resulting trade-off between conflicting objectives in a clear and comprehensible way.

1.4 Research Questions

Based on the aforementioned issues, the following research questions can be formulated:

- The complete design of complex machines consists of many different interacting hardware and control design possibilities that each have an (unknown) influence on the overall machine performance. How can the simultaneous co-design of both hardware and control be systematically formulated in a mathematical form, manageable for optimization purposes?
- How can the trade-off between (potentially) conflicting objectives be presented in a clear way?
- The application of iterative (evolutionary) optimization algorithms requires a large number of simulations. How can the mathematical system description be adapted so that the complete setup can be simulated efficiently and optimization results can be obtained in a reasonable time?
- Can system level analysis techniques be applied so that a priori insights can be obtained to efficiently exclude pre-determined infeasible combinations of hardware and control combinations?

1.5 Dissertation Outline

Chapter 2 provides a thorough overview of the co-design concept and what it means in the scope of this PhD dissertation. It first addresses a multi-objective optimization problem and explains the general properties of optimization problems. Next, an overview is given of existing co-design strategies, and extensive analysis is given of the current state-of-the-art in hardware and control co-design. This will show that none of the existing methods reaches the level of co-design applied in this work.

Chapter 3 starts with an overview of the novel workflow developed in this work to accomplish the hardware and control co-design. Next, the different objectives that apply to this co-design problem are given. Subsequently, more background information is given about the different types of optimization algorithms, and a well-founded choice is made for the algorithm used to execute the co-design problem. As a final part of this chapter, a way to clearly and graphically represent the optimization results is described.

After that, more detail is given on how the different parts of the workflow are dealt with. Chapter 4 starts with how the hardware architecture optimization can be mathematically represented and in this manner optimized using the suggested optimization algorithm. In addition, this chapter also presents an open-loop analysis that ensures that a priori information from the open-loop system can be used to speed up the workflow and thus achieve a more efficient optimization.

Next, Chapter 5 provides an overview of the meaningful control architecture features to obtain a satisfactory system performance. After that, the integration of these industrially relevant different control configurations in the optimization problem is discussed, which is a unique feature in the hardware and control codesign. The second element of the control configuration optimization mentioned in this chapter is the tuning of the corresponding controller parameters.

Furthermore, a novel methodology is explained to reformulate an open-loop state-space system and the corresponding control architecture as one closed-loop state-space system. One of the advantages of this method is that the closed-loop system response can be calculated much faster, which in turn means that the general co-design methodology can be performed more efficiently.

Thereafter, Chapter 6 describes how the approach of the different sub-problems from the previous chapters can be combined to implement the innovative co-design methodology. This chapter generically describes how the hardware and control co-design is performed, and provides some examples of applying this co-design methodology to models of practical cases.

Subsequently, Chapter 7 discusses in detail how the general hardware and control co-design methodology is accomplished on an active car suspension setup. The co-design methodology results are also validated on the existing physical setup. Finally, general conclusions and suggestions for future work are formulated in Chapter 8.

1.6 Publications and Acknowledgment

Articles in International SCI Journals

An overview of peer-reviewed journal papers that were published in the scope of this PhD:

- M. Haemers, S. Derammelaere, C. Ionescu, and K. Stockman, "Optimal Hardware and Control Co-Design applied to an Active Car Suspension Setup", *Machines*, Vol.9, issue 3, pp. 55, 2021
- M. Haemers, S. Derammelaere, A. Rosich, C. Ionescu, and K. Stockman, "Towards a generic optimal co-design of hardware architecture and control configuration for interacting subsystems", *Mechatronics*, vol. 63, pp. 102275, 2019

An overview of peer-reviewed journal papers that were published outside the scope of this PhD:

• S. Derammelaere, M. Haemers, C. Copot, F. Verbelen, C. Ionescu, and K. Stockman, "Realtime locomotion control of a snakeboard robot based on a novel model, enabling better physical insights", *European Journal of Control*, vol. 63, pp. 102275, 2018

Articles in conference proceedings

An overview of peer-reviewed conference papers that were published in the scope of this PhD:

- M. Haemers, C. Ionescu, B. Depraetere, K. Stockman, and S. Derammelaere, "Hardware and Control Co-Design enabled by a State-Space Formulation of Cascaded, Interconnected PID Controlled Systems", 7th International Conference on Optimization and Applications (ICOA2021), Wolfenbüttel, Germany, 19-20 May 2020
- M. Haemers, S. Derammelaere, C. Ionescu, K. Stockman, J. De Viaene, and F. Verbelen, "Proportional-Integral State-Feedback Controller Optimization for a Full-Car Active Suspension Setup Using a Genetic Algorithm", *3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control (PID18)*, Ghent, Belgium, 9-11 May 2018
- M. Haemers, S. Derammelaere, and K. Stockman, "Co-design of Controller and Setup Configuration using Genetic Algorithm", 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 13-15 September 2017

An overview of peer-reviewed conference papers that were published outside the scope of this PhD:

- C. Ionescu, M. Haemers, A. Maxim, C. Copot, S. Derammelaere and K. Stockman, "Automatic tuning of predictive control in a hydrostatic drive train system in nominal operation", 23rd International Conference on System Theory, Control and Computing (ICSTCC), 2019
- F. Verbelen, M. Haemers, J. De Viaene, S. Derammelaere, K. Stockman, and P. Sergeant, "Adaptive PI Controller for Slip Controlled Belt Continuously Variable Transmission", *3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control (PID18)*, 2018
- J. De Viaene, M. Haemers, F. Verbelen, S. Derammelaere, and K. Stockman, "Current Reduction in Stepping Motor Applications Using an Adaptive PI Controller Based on Linearized Dynamics", *3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control (PID18)*, 2018
- S. Derammelaere, M. Haemers, J. De Viaene, F. Verbelen, and K. Stockman, "A quantitative comparison between BLDC, PMSM, Brushed DC and Stepping Motor Technologies", *19th International Conference on Electrical Machines and Systems (ICEMS)*, 2016
- J. De Viaene, F. Verbelen, M. Haemers, S. Derammelaere, and K. Stockman, "Quantifying the commutation error of a BLDC machine using sensorless load angle estimation", *19th International Conference on Electrical Machines and Systems (ICEMS)*, 2016

Acknowledgment

This research was supported by Flanders Make, the strategic research centre for the manufacturing industry, and is part of a Flanders Make SBO project: '*ROCSIS: Robust and Optimal Control for Systems of Interacting Subsystems*.'

Chapter 2 Background on Co-Design

In this chapter, the concept of the simultaneous design of a system's hardware and control is discussed. An overview is given of the nomenclature that is used to indicate the different parts of this co-design. Next, some general properties of op-timizations are mentioned, after which different existing co-design strategies are explained. The chapter ends with an overview of which parts of the co-design can be found in the current literature and why previous works cannot achieve the profound level of co-design aspired in this PhD. Although in this chapter several properties of optimization algorithms are mentioned that are necessary to correctly indicate the state-of-the-art in co-design, an in-depth overview of the different algorithms to perform multi-objective optimizations will be explained in following Chapter 3.

2.1 Multi-Domain Optimization

In industrial applications, several objectives from different engineering domains are of importance. The simultaneous optimization of multiple, possibly conflicting, objectives is labeled as 'multi-objective' optimization [39]. One example is the selection of a new car by a customer. Of course, the car's comfort is crucial and can be quantified under different aspects. For example, the engine capacity will be essential to indicate how fast the car can accelerate. Other aspects such as road handling, sound insulation, automatic gearbox or trunk space will also have a particular influence on the perceived comfort. An unavoidable objective in this problem (and by extension in almost all problems) is the total cost. Every component that contributes to comfort will have a specific cost. Therefore, it is up to the end-user to make a good trade-off between the multiple objectives and choose a car that suits both the cost and the minimum desired level of comfort. Another example is a motor control problem in which the tracking error and the power consumption should be minimized. A minimal tracking error could be obtained at the expense of high power consumption, or a large tracking error is allowed while using very little power. In the extreme case, the motor could be turned off to achieve zero power consumption, but then the tracking error would be underwhelming [40].

Many industrial systems can be represented as a set of subsystems, connected through physical interactions. The overall system design optimization involves two main challenges (hence the term 'combined design' or 'co-design'), where both parts can have different objectives to take into account. These challenges are the decision on the optimal hardware architecture and the optimal control configuration. The hardware architecture is further subdivided into the actuator and sensor selection on the one hand and the actuator and sensor placement on the other hand. The control configuration consists of the control architecture and controller tuning. Fig. 2.1 shows the subdivision of the different parts of the general co-design. This co-design terminology is widely used in literature (e.g., [21,41–44]) and is consistently used throughout this dissertation. The term co-design is sometimes referred to in the literature as 'integrated design'.



Figure 2.1: Overview of the nomenclature used for the different aspects of the total system composition optimization

The hardware architecture optimization consists on the one hand of the actuator and sensor placement optimization. This means optimizing the physical location of the hardware within the entire system. The hardware architecture optimization also consists of the actuator and sensor selection, by which the choice of the type and size of actuators or sensors is understood. In this way, a distinction can be made between actuators or sensors with different cost, dynamics, output saturation, resolution, etc.

The control configuration optimization consists on the one hand of the control architecture optimization, in which the structure of the controller is adjusted and thus optimized. In this way, the consideration of different control features can also be considered as part of the system composition optimization. On the other hand, the control configuration optimization also consists of the controller tuning optimization. This refers to setting the controller values (= controller parameters).

A generally applicable method for combined architecture design, component sizing, and control configuration optimization of the overall system design is still an open research question [38], as mentioned in detail in Section 1.3. This PhD research will provide an answer by developing a generically applicable procedure in which both the placement and selection of the actuators and sensors are carried out,

as well as the simultaneous determination of the control architecture and associated controller tuning. The overall system optimization aimed at in this work has a design space that is unprecedented in current control literature, as will be shown in Section 2.4. This general co-design methodology will enable the end-user to gain a better understanding of the inevitable trade-off between cost and performance.

2.2 Background on Optimizations

This section highlights some general principles of optimization problems that are important in the scope of this dissertation. The complexity of the optimization algorithm depends on the nature of the optimization problem (i.e., the nature of its constraints and objectives) and its size (i.e., the number of design parameters and constraints). The most basic and general form of a continuous optimization problem is to minimize a scalar objective function f of the variable x, as seen below in (2.1) where $x \in \mathbb{R}^n$ is a real vector with $n \ge 1$ components and f(x) is a smooth function.

$$\min_{x} f(x) \tag{2.1}$$

This can be extended for multiple-objective optimization problems with k objectives, as shown in (2.2) [40].

$$\min_{x} f(x) = \min_{x} \left[f_1(x), f_2(x), \dots, f_k(x) \right]$$
(2.2)

A function is said to be smooth when second derivatives exist and are continuous. On the contrary, a function can be non-smooth and even discontinuous [45]. The difference between these three is shown in Fig. 2.2, in which a smooth function $f_s(x)$ is shown in blue, a non-smooth function $f_{ns}(x)$ is shown in yellow, and a discontinuous function $f_d(x)$ is shown in green.



Figure 2.2: General example of a smooth function $f_s(x)$ in blue, nonsmooth function $f_{ns}(x)$ in yellow and a discontinuous function $f_d(x)$ in green

A distinction is made between local and global minima. A local minimum is a point at which the objective function is smaller than at all the other (feasible) points in its vicinity, but possibly greater than at a distant point. Instead, a global minimum is a point where the function value is smaller than or equal to the value of all other feasible points. Fig. 2.3 shows a graph of an objective function with multiple local minima and one global minimum.



Figure 2.3: General example of a function (blue) with two local minima and one global minimum within the range of feasible x values (yellow)

2.2 Background on Optimizations

Another important characteristic in optimizations is the concept of convexity, which is a prerequisite for optimization algorithms that quickly and efficiently reach a global minimum. A function f(x) is convex if for any two points x_1 and x_2 , the graph of f(x) lies below the straight line connecting $(x_1, f(x_1))$ to $(x_2, f(x_2))$.

Fig. 2.4 shows the difference between a convex function $f_c(x)$ and a nonconvex function $f_{nc}(x)$. The graph of $f_{nc}(x)$ lies above the dashed straight line connecting two points on the graph, making it a non-convex function. In convex optimizations, each local minimum is also a global minimum. Finding a local minimum in convex optimization is sufficient to conclude that a global minimum is found [45].



Figure 2.4: Example of a convex function $f_c(x)$ in yellow and a non-convex function $f_{nc}(x)$ in blue

Another classification within optimization problems is related to the use of constraints. In practical applications, there will always be certain constraints in effect. Constraints can be ignored if it is assumed that they have no effect on the optimization solution. Unconstrained problems also arise from reformulations of constrained problems, in which the constraints are replaced by penalization terms in the fitness function which discourage constraint violations. Constrained optimization problems emerge when there are explicit constraints on the design parameters. These constraints can vary from simple bounds ($0 \le x_i \le 100$) to more general linear constraints ($\sum_i x_i \le 1$), or non-linear inequalities that represent complex relationships among the variables [45]. Below in (2.3), the general optimization example of (2.1) is extended with an example of an inequality constraint.

$$\min_{x} f(x)$$
subject to $5x_3 \le 42$
(2.3)

Some optimization problems have a particular type of constraints in which design parameters can only attain integer values. An example of this is when an optimization problem can decide whether a specific controller feature is active or not. In this example, the controller feature presence or absence is represented by a binary number, implemented as an integer constraint with limits [0, 1]. The strategy of ignoring the integer requirement, solving the problem with real values, and then rounding all the components to the nearest integer cannot only lead to illogical solutions, but it can also by no means guarantee to give solutions that are even close to optimal. Problems with both continuous and discrete design parameters are denoted as mixed-integer problems and should be handled using discrete optimization tools [45, 46]. One possibility is applying a branch-and-bound optimization method, in which a set of possible integer values is presented as a tree structure with different branches. The algorithm will check the branches of the tree that represent subsets of the solution. A branch (and thus subset of possible solutions) will not be further considered if it cannot produce a better solution than the best one found so far by the algorithm [47]. To incorporate mixed-integer problem requirements in the mathematical formulation, (2.3) is changed by adding a constraint on the design parameters, as described in (2.4) where the design parameters x partly consist of real values x_r and integer values x_i .

$$\begin{array}{ll}
\min_{x} & f(x_r, x_i) \\
\text{subject to} & x_r \in \mathbb{R} \\
& x_i \in \mathbb{Z}
\end{array}$$
(2.4)

2.3 Co-Design Strategies

In literature, the co-design optimization strategies are grouped by [48] into sequential, simultaneous, iterative and nested ones, see Fig. 2.5.

In the (traditional) sequential strategy, the hardware and controller optimizations are considered separately and therefore optimized independently. Since both the hardware architecture and the control configuration have a significant influence on the entire system performance, this strategy cannot guarantee an optimal situation [49].

Iterative strategies optimize a hardware architecture for a fixed control configuration, then optimize the control, fixing the hardware, and so on until convergence. Iterative strategies often reduce co-design optimization problems to sequences of convex hardware and control optimization problems [50]. The sequential and iterative strategies fail to guarantee system-level optimality because they do not necessarily converge to an optimal solution. Also, they require large disparate teams to solve complex multidisciplinary optimization problems collaboratively. [31]. An interesting example of a scenario in which iterative hardware architecture and linear-quadratic regulator (LQR) controller tuning optimization fails to generate



Figure 2.5: Co-design optimization strategies

an optimal system composition is the case in which the initial guess for the iterative optimization is the output of a sequential system composition optimization. Given this initial guess, the iterative process will not change the controller tuning since the linear-quadratic regulator (LQR) method already includes some form of optimization (see [36] for details). Correspondingly, the iterative process will not change the hardware architecture for a given linear-quadratic regulator (LQR) controller, as it assumes that the current hardware architecture is already optimal for the control, and hence no better hardware architecture can be found. For that reason, the output of the iterative process will be the sequential optimum, which is not necessarily equal to the global optimum [49].

Nested strategies contain two optimization loops: an inner loop that completes the controller optimization and an outer loop that completes an iteration of the hardware optimization. For each iteration of the outer loop, an optimization of the inner loop is performed. For example, a nested strategy optimization starts with a proposal of the hardware configuration, after which an (iterative) optimization algorithm is executed on the controller in the inner loop until an optimum is reached for the current hardware configuration. In turn, the outer loop also iterates until a solution is obtained. Nested strategies can guarantee system optimality in some cases while maintaining the more straightforward interdisciplinary partitioning used in sequential ones [51].

With the simultaneous strategies, a single optimization routine is performed in which both the hardware and controller parameters are optimized at the same time. With this strategy, only one optimization routine is conducted.

Previous work [38] provides a comprehensive overview of how the different optimization strategies are applied to co-design the hardware and control of a hvbrid electric vehicle (HEV). Conclusions of that work are that the interaction between HEV components is becoming increasingly important and that neglecting it in the design step leads to loss of potential. Sequential HEV design strategies prove clear advantages but also introduce several challenges in solving the optimization problem. Nested HEV optimization poses more challenges in finding a global optimal solution at the system level and creates a shift towards multi-disciplinary design. Even so, the paper shows that HEV designs with significantly lower fuel consumption and emissions can be found using nested strategies compared to sequential strategies. Additionally in [31], the differences between sequential, iterative, and combined strategies for the hardware and controller tuning co-design of a DC motor are examined and compared. Solutions show that the combined optimization strategy outperforms the sequential and iterative strategies in terms of motor weight, speed response error, and required voltage during a test schedule. Conclusions of both cited papers also state that design and control analysts generally do not think in the same terms and that they need to integrate their tasks to a simultaneous topology, sizing, and control design optimization exploring systemlevel optima. This is precisely the main research question for this PhD dissertation.

2.4 State-of-the-art and Applications

As mentioned before, this work considers the co-design of hardware architecture and control configuration. The hardware architecture is further subdivided into actuator and sensor selection and placement, and the control configuration optimization is further separated into control architecture optimization and controller tuning optimization (see Fig. 2.1 on page 20). The following is a literature review of these different co-design aspects, indicating the extent to which they have already been optimized (simultaneously).

Determining the controller parameters for cyber-physical systems (CPS) is a very widely described problem. There are numerous methods to obtain a desirable (not necessarily optimal) feedback control design, but without considering hardware architecture design. For example, proportional-integral-derivative (PID) controller parameters can be obtained based on root locus or frequency response requirements or autotuning methods such as the Ziegler-Nichols method [52]. Other ubiquitous control algorithms and corresponding tuning methods are, for exam-

ple, Internal Model Control (IMC) [53], linear-quadratic regulator (LQR) [36], Model Predictive Control (MPC) [54], robust H_{∞} control [36], or Fractional Order PID Control [55]. Multi-domain optimization methods have been applied to obtain the controller parameters for mechatronic applications. Examples of previous work using a Genetic Algorithm (GA) to optimize feedback controller values via the linear-quadratic regulator (LQR) method for an inverted pendulum, quartercar suspension, hovercraft control, and multi-machine power systems can be found in [56–60], respectively. Furthermore, [61] and [62] used a Genetic Algorithm to optimize a robust H_{∞} controller for a vehicle suspension control, while an optimal Fractional Order PID controller tuning for an automatic voltage regulator system is accomplished in [63]. These optimizations are limited in the sense that only the control parameters are optimized.

There are also examples in which parts of the hardware architecture are optimized without considering the control optimization. For example, in [64–67] an optimization of mechanical design variables is done for a vehicle suspension system, a motor-driven four-bar system, and a passive bipedal walker, respectively. Fig. 2.6 displays the solutions using a sequential and a simultaneous optimization strategy for the hardware design variables of a passive bipedal walker. The solution found after executing the simultaneous optimization strategy shows a much better fitness according to the objective function.



Figure 2.6: Solutions after applying a sequential (left) and a simultaneous (right) optimization strategy for a passive bipedal walker (from [67])

Determining the optimal actuator placement is done in [68], where the optimal placement problem is presented as a mixed-integer problem, and is solved using a branch-and-bound procedure. Another recent example of actuator placement optimization for dynamic networks (without considering control design) can be found in [69]. The problem of selecting the optimal sensor locations is done by [70] and [71] for the flow over an airfoil and a double-link variable stiffness actuated

robot, respectively. Both examples obtain optimized sensor locations, but do not take control design and changing hardware types (= actuator and sensor selection) into account.

More recently, the simultaneous optimization of both the actuator and sensor placement for a small shell structure using a Genetic Algorithm (GA) is performed by [72]. The optimal actuator and sensor selection for a non-isothermal tubular reactor can be found in [73]. In that case, the optimization problem is converted to a mixed-integer convex problem statement, which is only possible for specific problem classes. An optimization of both the placement and the selection of actuators and sensors in shown in [74]. In that case, it concerns collocated actuator/sensor pairs where each cantilever beam mode must meet a specific peak gain. In this section's examples, only the hardware architecture is optimized (or individual parts of it), without optimizing the control design.

Up to this point, this literature overview does not provide examples of codesign, as no hardware and control parts are treated simultaneously. Table 2.1 gives an overview of the references cited so far, indicating which parts of the system composition are optimized.

		Ha	Control Configuration										
	Actuator	Actuator	Sensor	Sensor	Mechanical	Control	Controller		Feedback	H_2		Other	
Reference	Placement	Selection	Placement	Selection	design variables	Architecture	Tuning	LQR	Gains	H_{∞}	PID	methods	Applications
[56]	0	0	0	0	0	0	٠	•	0	0	0	0	Inverted pendulum
[58]	0	0	0	0	0	0	•	•	0	0	0	0	Vehicle suspension system
[57]	0	0	0	0	0	0	•	•	0	0	0	0	Inverted pendulum
[59]	0	0	0	0	0	0	•	•	0	0	0	0	Hovercraft control
[61]	0	0	0	0	0	0	•	0	0	•	0	0	Vehicle suspension system
[62]	0	0	0	0	0	0	•	0	0	•	0	0	Vehicle suspension system
[63]	0	0	0	0	0	0	•	0	•	0	0	0	Voltage regulator system
[60]	0	0	0	0	0	0	•	•	•	0	0	0	Multi-machine power system
[64]	0	0	0	0	•	0	0	0	0	0	0	0	Vehicle suspension system
[65]	0	0	0	0	•	0	0	0	0	0	0	0	Vehicle suspension system
[66]	0	0	0	0	•	0	0	0	0	0	0	0	Motor-driven four-bar system
[67]	0	0	0	0	•	0	0	0	0	0	0	0	Passive bipedal walker
[68]	•	0	0	0	0	0	0	0	0	0	0	0	Integrator chain
[69]	•	0	0	0	0	0	0	0	0	0	0	0	Set of bench-mark case-studies
[70]	0	0	•	0	0	0	0	0	0	0	0	0	Airfoil flow
[71]	0	0	•	0	0	0	0	0	0	0	0	0	Double-link VIA robot
[72]	•	0	•	0	0	0	0	0	0	0	0	0	Smart shell structure
[73]	•	0	•	0	0	0	0	0	0	0	0	0	Non-isothermal tubular reactor
[74]	•	•	•	•	0	0	0	0	0	0	0	0	Cantilever beam structure

Table 2.1: Overview of cited works indicating which parts of the system composition are optimized. Since no hardware and control parts are treated simultaneously, these examples do not represent a co-design optimization.



Figure 2.7: Structure of a two-link arm robot (from [77])

To a limited degree, efforts have been made to implement a multi-domain codesign of control configuration and hardware architecture in electromechanical systems, but in all cases, it is always assumed that the control architecture is fixed. For example, an optimization problem with a small number of design parameters describing a hard disc suspension geometry and its corresponding controller tuning was formulated in [75]. The result is an optimization of the design variables of a hard disc suspension geometry and the controller feedback gains through the use of the linear-quadratic regulator (LQR) method. Similarly, the co-design of the controller tuning through the use of a linear-quadratic regulator (LQR) method and the geometrical properties of machine tools, an active vehicle suspension system, a satellite altitude control, an angular motor controlling a mass position, and a parallel manipulator mechanism was detailed in [29, 32, 33, 49, 76], respectively.

A design method for the sequential optimization of the mechanical structure and the controller tuning for a two-link high-speed robot is developed in [77], of which the setup is shown in Fig. 2.7. First, a two-link, non-rigid arm is analyzed, and a simple dynamic model representing rapid positioning processes is attained. Next, optimal feedback gains minimizing the settling time are obtained as functions of the structural parameters describing the arm link geometry. These structural parameters are then optimized using a gradient projection method to acquire an overall optimal performance. This displays an example of the sequential co-design strategy, as mentioned above in Section 2.3.

More recently, [78] combined mechanical design properties and the PD controller tuning for an airborne wind energy system. Other controller tuning methods are also used in the co-design of the controller parameters and mechanical design variables. For example, [79] and [80] use the H_2/H_{∞} method to obtain the controller tuning parameters. PID values can also be directly optimized, as shown in [31] for a DC motor. In [81], an example is shown where the co-design of mechanical design variables and controller parameters is performed for a hybrid



Figure 2.8: Microelectromechanical System (MEMS) for a confocal scanning microscope (from [35])

power train system used in a hydraulic excavator.

Microelectromechanical systems (MEMS) devices are useful in many fields, including motion control applications, due to their small size and low power requirements. In [35], the physical dimensions of a MEMS actuator and its controller tuning are jointly optimized, both sequentially and simultaneously. This optimization does not take control architecture optimization or actuator and sensor selection into account. The particular MEMS device that is the subject of this research (see Fig. 2.8) has been proposed for a confocal scanning microscope to study biological phenomena that occur on a microsecond time-scale. This MEMS device uses four actuators to produce an out-of-plane displacement. To produce this displacement, each of the four actuators is excited with a voltage, resulting in horizontal movement of the silicon shuttles. The micro-hinges on the platform bend as shown in Fig. 2.9, and the platform moves vertically. The amount of movement resulting from the actuation depends on both the applied voltage and the actuator physical dimensions. The solutions found in the sequential problem constrained by the actuator natural frequency closely relate to the solutions generated by the simultaneous problem. Therefore, for a given application, there may be more than one design available to satisfy displacement and settling time requirements. This also indicates that for this particular case and this choice of decision variables, either co-design strategy (sequential \leftrightarrow simultaneous) will produce a set of optimal solutions. A nested hardware/controller optimization for a combined passive/active automotive suspension for a quarter-car model was executed in [34]. In that case, the controller feedback gains and the physical variables (passive stiffness and damping coefficients) were optimized for an objective incorporating sprung mass acceleration, tire deflection, suspension stroke, and maximum active control force.

The co-design of the sensor placement and the PID controller tuning (without considering control architecture optimization) was recently performed for a flexible



Figure 2.9: Microelectromechanical System (MEMS) hinge actuation (from [35])

wing of an Unmanned Aerial Vehicle (UAV) using an evolutionary algorithm [82]. The co-design approach resulted in a 45,83% reduction in generalized vibrations energy and about 52,16% reduction in wind load alleviation when compared with designs where the sensor locations are not optimum. Similarly, the sensor placement and an LQR controller tuning is performed for a mass-spring system in [83]. There are also examples where both sensor and actuator placement are optimized as part of the hardware architecture and controller tuning co-design. For example, both the actuator placement, sensor placement, and controller tuning are optimized in [84–87]. The studied applications are a vibroacoustic plate, a random state-space system, a robotic arm, and general stochastic linear systems. These examples do not consider changing controller architectures or the hardware selection of different types of actuators or sensors.

An application area where the co-design of the hardware architecture and the controller tuning also received considerable attention is that of the large space intelligent truss structures (LSITS). These are widely used in spacecrafts as the supporting truss structure in, e.g., shuttle radars or in space stations. An example of this is NASA's Shuttle Radar Topography Mission (SRTM) to measure a digital elevation model of the earth surface. The sensors are connected to the shuttle through a 60 meter large space intelligent truss structure (LSITS) [88], as shown in Fig. 2.10. Due to its properties, the vibration of LSITS will happen very quickly. If the truss structure is impacted by some particles from outer space or if the spacecraft makes an altitude maneuver, a structural vibration will manifest. As the vibration happens, it will cause many severe problems to the payloads supported on the truss structure. Hence, the issue of vibration control for LSITS is crucial.

The control of intelligent truss structures has achieved remarkable progress by using piezoelectric actuators and sensors, often applied in collocated pairs. This way, a large space intelligent truss system (LSITS) becomes a complex truss sys-



Figure 2.10: Example of a 60m long large space intelligent truss system (LSITS) supporting an outboard antenna (from [88])

tem consisting of common rods and active rods, as for example in Fig. 2.11. In each active rod, there is at least one actuator and one sensor. One of the critical points in the vibration control for LSITS is the location assignment of actuators and sensors. The placement optimization of actuators and sensors is essential for the vibration control of LSITS. In [89] the actuator placement, sensor placement, and controller tuning (in the form of Fuzzy control) are simultaneously optimized. An optimization of particular mechanical design variables is added in [90]. Both papers use a Genetic Algorithm (GA) to find a solution while coping with the existing constraints and the mixed-integer character of the optimization problem.

A hardware architecture and controller tuning co-design for controlling high frequency vibrations in a flexible cantilever beam can be found in [91]. In that work, a Genetic Algorithm (GA) is utilized for optimizing the location and types of collocated sensor and actuator pairs, while simultaneously optimizing the controller tuning using a linear-quadratic regulator (LQR) method. The cantilever beam application has 64 possible positions and four possible sizes for the actuator/sensor pairs, and by using a Genetic Algorithm (GA), a solution is found.

It is clear that a simultaneous co-design of both hardware architecture and control configuration can result in a more efficient design process, optimizing for multiple (possibly conflicting) objectives. The state-of-the-art shows that extensive research has already been done into one or more aspects of the co-design of hardware architecture and controller tuning. However, all of the co-design examples mentioned above are limited because they can only deal with fixed and rather mod-



Figure 2.11: Large space intelligent truss system (LSITS) test setup (from [89])

est control architecture. In other words, the control architecture is never included as a design parameter. Furthermore, the examples are focused on one particular application or a specific optimization problem and therefore lack universal applicability.

In contrast, the approach presented in this PhD is a generally applicable methodology, capable of simultaneously optimizing the actuator and sensor selection and placement as well as the control architecture and controller tuning. By doing so, a larger design space is explored to obtain a system-level optimum. Moreover, in this work, the actuators and sensors are not necessarily arranged in collocated pairs, which is required in specific methods mentioned above. This is why none of the above examples succeeds in achieving the profound level of co-design of control configuration and hardware architecture presented in this dissertation.

Table 2.2 depicts an overview of recent papers considering different variations of hardware architecture and controller tuning co-design. Many different controller tuning methods can be implemented, e.g., PID, LQR, or H_2/H_{∞} control, as indicated in the table. It is important to note the general co-design framework proposed in this PhD allows to carry out an advanced hardware architecture optimization and simultaneously apply a wide range of controller methods. Additionally, the proposed co-design methodology also allows to perform a control architecture optimization as part of the control configuration optimization. As can be seen in Table 2.2, there is no previous work that provides this capability.

Π

						· · · ·	<i>,</i>						
		Ha	rdware Arcl	hitecture	Control Configuration								
Reference	Actuator Placement	Actuator Selection	Sensor Placement	Sensor Selection	Mechanical design variables	Control Architecture	Controller Tuning	LQR	Feedback Gains	H_2 H_∞	PID	Other methods	Applications
[75]	0	0	0	0	•	0	•	•	0	0	0	0	Hard disc suspension
[32]	0	0	0	0	•	0	•	•	0	0	0	0	Vehicle suspension system
[34]	0	0	0	0	•	0	•	•	0	0	0	0	Vehicle suspension system
[33]	0	0	0	0	•	0	•	•	0	0	0	0	Satellite altitude control
[49]	0	0	0	0	•	0	•	•	0	0	0	0	Angular positioning motor
[35]	0	0	0	0	•	0	•	•	0	0	0	0	MEMS
[77]	0	0	0	0	•	0	•	0	0	0	0	•	High-speed two-link robot
[29]	0	0	0	0	•	0	•	0	0	0	0	•	Machine tools feed system
[76]	0	0	0	0	•	0	•	0	0	0	0	•	Parallel manipulator mechanism
[78]	0	0	0	0	•	0	•	0	0	0	0	•	Airborne wind energy system
[79]	0	0	0	0	•	0	•	0	0	•	0	0	Chip mounter machine
[80]	0	0	0	0	•	0	•	0	0	•	0	0	Flexure-linked biaxial gantry
[31]	0	0	0	0	•	0	•	0	0	0	•	0	DC motor
[81]	0	0	0	0	•	0	•	0	•	0	0	0	Hybrid power train system
[82]	0	0	•	0	0	0	•	0	0	0	•	0	Flexible UAV wing
[83]	0	0	•	0	0	0	•	•	0	•	0	0	Mass-spring system
[84]	•	0	•	0	0	0	•	•	0	0	0	0	Stochastic linear systems
[85]	•	0	•	0	0	0	•	•	0	0	0	0	Vibroacoustic plate control
[86]	•	0	•	0	0	0	•	0	0	•	0	0	Random state-space system
[87]	•	0	•	0	0	0	•	0	0	0	•	0	Robotic arm control
[89]	•	0	•	0	0	0	•	0	0	0	0	•	Intelligent truss system
[90]	•	0	•	0	•	0	•	0	•	0	0	0	Intelligent truss system
[91]	•	•	•	•	0	0	•	•	0	0	0	0	Flexible beams

 Table 2.2: Overview of the co-design elements of the cited works

The term 'co-design' is also used in the research domain of Networked Control Systems (NCS). A Networked Control System (NCS) is a control system wherein the control loops are closed through a communication network in which control and feedback signals are exchanged among the network in the form of information packages. Random time delays between sensing and actuating or data corruption during transmission (so-called data attacks) are simulated to test the system behavior when communication errors occur. Fig. 2.12 shows a general architecture for a Networked Control System with different kinds of data attacks. Within this research domain, 'co-design' is used to indicate the simultaneous determination of hardware components and controller tuning, but taking into account restrictions like changing sampling periods, delay, data bandwidth, jitter, switching rules, or scheduling schemes. [92] provides a comprehensive introduction to Networked Control Systems (NCS) and the related concepts of event-triggered and self-triggered control.



Remote control center

For example, in [94], the co-design problem of controller tuning and communication sequence is addressed for networked control systems (NCS) where the capacities of communication networks at both sides of the system are limited so that the sensor and actuator data cannot be processed simultaneously. This codesign problem is solved in three steps. First, feasible communication sequence

Figure 2.12: A general representation of a Networked Control System (NCS) under data attacks (yellow). Attack 1: data integrity attacks on the system state. Attack 2: data availability and/or integrity attack on sensor data. Attack 3: data availability and/or integrity attack on control data. (reproduced from [93])

sets for both sides are determined. Second, output feedback controllers are built on the feasible communication sequence sets. Third, switching rules are set up for the communication sequences based on the designed controller. Notably, in the first step, all the communication sequences are determined that preserve controllability and observability of the system. Then in the second step, an output feedback controller applicable for all these communication sequences is derived. Finally, a switching strategy is established for the communication sequences to switch among their feasible sets during system operation. For other recent co-design examples in the research area of Networked Control Systems (NCS), please refer to [95–99]. In addition, a detailed and recent overview can be found in [30].

Research and literature on Networked Control Systems (NCS) are more focused on optimally handling the limitations linked to the control network and distributing the needed communication forms over the different agents in the multiagent system [94, 97]. The use of Networked Control Systems (NCS) and how to cope with the associated communication restrictions are outside the scope of this dissertation.

2.5 Chapter Conclusions

Nowadays, multiple objectives from different engineering domains have to be taken into account for real-life applications since a present-day Cyber-Physical System (CPS) involves the collaboration of different components from various application areas. This also means that methods need to be developed to combine these different parts thoroughly. This is typically called 'multi-domain optimization.' Furthermore, some background on general optimization principles is specified, such as objective function smoothness, local and global minima, (objective function) convexity, and the application of (mixed-integer) constraints.

A specific example of multi-domain optimization occurs when both the hardware architecture and the control configuration of a system are simultaneously optimized. Next, the nomenclature is mentioned that is used throughout this dissertation to indicate the different parts of the system. The system composition co-design refers to the hardware architecture on the one hand and the control configuration on the other hand. In turn, the hardware architecture is subdivided into actuator and sensor selection and placement, while control configuration refers to both controller tuning and control architecture. Traditionally, these different parts are optimized sequentially, leading to a sub-optimal solution. Better results can be obtained if optimization strategies are applied that enable simultaneous optimization of the hardware architecture and control configuration.

As a final section, this chapter provides a comprehensive overview of the stateof-the-art in co-design of hardware architecture and control configuration. This shows that much research has already been done in this field, but it also exhibits that many important features have not yet been addressed in the co-design problem. First and foremost, none of the state-of-the-art works offers the possibility to optimize the control architecture as part of the control configuration optimization. This is a unique aspect of the co-design methodology presented in this PhD. The literature overview also shows that many co-design techniques are tailored towards specific system classes, and therefore lack a generic character with great freedom in defining constraints and objectives.

The simultaneous optimization of different co-design aspects also influences the necessary computing power of the optimization algorithm. In the next chapter, the different types of optimization algorithms used for multi-objective optimizations are discussed.

Chapter 3

Multi-Domain Optimization Workflow

This chapter starts with an overview of the workflow to optimize the intended codesign of hardware and control. Initially, the different steps of the workflow are introduced rather superficially, but they will be discussed in more detail in the following chapters. This chapter then explains the optimization objectives and the requirements that an optimization algorithm must meet to execute the hardware and control co-design. Next, an overview is given of generally known multi-objective optimization algorithms to arrive at a well-founded choice of the algorithm used in this PhD. Then, an explanation is given on how the optimization results can be displayed graphically. Finally, background information is provided on the state-space modeling used to represent the dynamics of the system to be optimized, including an introduction to the state-space system used throughout this dissertation to further explain specific parts of the co-design.

3.1 Optimization Workflow Overview

In this section, a general overview is given of the workflow to complete the codesign methodology, starting from an initial system model to a graphical representation of the results. The content of each step is briefly discussed without going into detail. The following chapters elaborate on the workflow components in more detail, each time also discussing the content on an illustrative example. Therefore, this section can be seen as a framework for what follows in the dissertation and can be used to see how the different parts relate to each other in function of the complete co-design methodology. Fig. 3.1 shows a graphical overview of the different steps in the workflow.



Figure 3.1: Graphical overview of the general co-design optimization workflow

At the start, a mathematical representation of a system is needed. Section 3.6 handles different kinds of state-space modeling techniques that can be used. The general co-design methodology presented in this dissertation mainly focuses on linear time-invariant (LTI) state-space systems, but the methodology is also applicable to linear parameter-varying (LPV) and linear time-varying (LTV) systems (see Section 6.4 on page 153).

Next, all possible locations of actuators and sensors in the system are defined. The possible locations indicate on which physical location in the system an actuator or sensor can be applied. In addition, the different possible types of actuators and sensors are defined. For example, different types of actuators can indicate the maximum actuator output. The types of sensors can represent which quantities of the system can be measured or what sensor resolution is active. Each type of actuator and sensor has a certain application-dependent cost and can be set by the end-user. Defining the possible locations and types of actuators and sensors will determine the hardware architecture design space. Sections 4.1 and 4.2 on pages 70 and 73 show how these design space possibilities are mathematically translated into design parameters and corresponding changes in the state-space system. The resulting (open-loop) state-space system is referred to as SS_{OL} .

After the previous step, all possible combinations are defined in the hardware architecture. Certain combinations of these hardware architectures will lead to a system composition that is illogical or even infeasible. For example, it is easy to see that a system composition without actuators or sensors will not result in a workable situation. If the system consists of multiple subsystems, a link between subsystems may be very weak or even non-existent. In these cases, it will not be possible to control the subsystem if there is no actuator acting on it. The same goes for the sensors: if a particular subsystem parameter cannot be measured or derived from measurements on other sufficiently coupled subsystems, insufficient information will be available to control this subsystem. These properties are referred to as the system controllability and observability and can be determined quickly and efficiently based on the open-loop state-space system. In Section 4.3 on page 76, an open-loop analysis is discussed in detail to determine the controllability, observability, and feasibility of all possible hardware architectures. If it turns out that a hardware architecture is not feasible, it is useless to (try to) calculate the response. Therefore, this open-loop analysis can be used later in the optimization to ensure that a non-feasible hardware architecture is detected before calculating its closed-loop response. This way, the optimization will become more efficient, and the results will be achieved faster.

The next step in the co-design workflow is to define the possibilities concerning the control architecture in order to achieve closed-loop control. Many different control loop possibilities can be applied, such as decentralized and distributed P, PI and PID controllers, cascade controls, feedforward control, synchronizing control, etc. In this way, the design space of the control architecture is defined. The presence or absence of specific control architectures can be established by linking binaries to the different control features. In this way, the optimization algorithm can choose whether to apply a control loop feature, depending on the performance and constraints. More information on the control architecture optimization can be found in Chapter 5. Then, the open-loop system and the entire control structure are transformed into a discrete time closed-loop state-space system. The transformation algorithm is discussed in Chapter 5. The result is a mathematical representation of the closed-loop system whose main advantage is that the closed-loop response can be calculated quickly and efficiently. Since the optimization algorithm has to calculate a large number of system responses, this provides a tremendous time-saving in the execution of the complete co-design methodology. After this step, the state-space system is referred to as SS_{CL} .

On this point in the workflow, the entire design space is determined in terms of hardware architecture and control configuration, and this design space is translated into a mathematical form that allows to apply a suitable optimization algorithm. Section 3.4 on page 44 provides an overview of different multi-objective optimization algorithms and an explanation why a Genetic Algorithm (GA) is used to perform the co-design in this work. By adjusting the design parameters, other hardware architectures and control configurations will be effectuated, and thus variations will be obtained in the system composition. From each system composition, the closed-loop response is calculated. From this closed-loop response, the performance is quantified by a user-defined fitness function, resulting in a fitness value. Based on this fitness value and the (non-linear) constraints, the Genetic Algorithm will adjust the design parameters generation after generation to obtain a system composition with the lowest possible fitness value and thus the highest possible performance. More information on applying a Genetic Algorithm to optimize the system composition can be found in Section 6.2 on page 128. It is important to emphasize that the general co-design methodology described in this dissertation is not restricted to using a Genetic Algorithm. If other (better) algorithms exist that meet the same requirements as the Genetic Algorithm, they can also be applied to accomplish a co-design of the hardware architecture and the control configuration.

The execution of the Genetic Algorithm is always done for a maximum system cost. In this way, the maximum achievable performance is determined for a fixed total cost of the system. In order to gain insight into the progress of this maximum achievable performance in function of a range of maximum costs, the Genetic Algorithm can be executed for several costs. In this way, the trade-off between the total implementation cost and the maximum achievable performance is determined and this trade-off can also be graphically displayed in a Pareto front (see Section 3.5 on page 58 for more background information). This Pareto front can provide a lot of insight to make a well-considered and well-informed choice on which hardware architecture and control configuration to be applied.

3.2 Optimization Objectives

Well-defined objectives are required to determine the performance of a system in order to carry out an optimization. How well a setup meets a particular objective is quantified based on a fitness value. The fitness value will change as a function of the design parameters. An optimization will always try to adjust these design parameters so that the fitness value is as low as possible.

A significant advantage of the co-design method described in this work is that it allows a high degree of freedom in defining (multiple) objectives. For example, the system performance can depend on tracking errors, settling times, vibrations, energy consumption, frequency responses, etc. Multiple objectives can be represented in one fitness value by calculating a weighted average.

The objectives related to a system cost are generally inversely proportional to the objectives indicating the performance. Thus, it seems logical that the machine's maximally achievable performance increases when it is allowed to cost more as better actuators, sensors, and control techniques can be implemented. Contrary to this, however, the economic aspect also plays an important role, and the cost of the machine should preferably be reduced as much as possible. It is clear that there will be a trade-off between cost and performance.

As mentioned before, the system cost will depend on the types and numbers of actuators and sensors. Also, the control configuration will involve a specific cost. This cost includes the controller hardware and, for example, the cost of programming controllers or the cost of training engineers or operators to tune or diagnose the controller. Therefore, determining the cost of a component can be very challenging. For example, hardware costs may depend on the number of orders, different suppliers, delivery time, etc. That is why this dissertation does not indicate the cost in a currency, but in percentages. These percentages will be chosen and set according to the proportions of actual parts and services.

3.3 Optimization Algorithm Requirements

The general co-design methodology described in this dissertation is a **multi-objective** optimization problem since an optimal solution is desired for different (conflicting) objectives, such as total system cost, reference tracking, vibration suppression, energy consumption, etc. For this purpose, a graphical representation method is desired to clearly represent the trade-off between conflicting objectives. Next to real numbers, multiple design parameters are integers or binaries since the optimization algorithm has to make choices on whether or not to apply control architecture features or on which discrete types of actuators and sensors to apply. Therefore, the chosen optimization algorithm must be able to handle **mixed-integer** problems. Moreover, these integer design parameters also cause a **discontinuous** (and thus also non-smooth) **objective function**.

In addition to the integers and binaries representing the hardware selection and control configuration, the controller tuning parameters must also be optimized simultaneously. For example, for PID controllers, this means three real number design parameters for every controller. Preferably a large number of hardware locations, hardware types and controller tuning parameters are optimized simultaneously to obtain a large system design space. The applied optimization algorithm for this co-design problem must therefore be able to handle a **large number of design parameters**.

In addition, the optimization algorithm must also be able to deal with **non-linear constraints**. These apply, for instance, because the total cost of the system depends on the chosen integer design parameters in a non-linear way.

3.4 Optimization Algorithm Selection

In general, one may distinguish between gradient-based and derivative-free algorithms as the two main approaches to deal with multi-objective optimization problems. The following is a non-exhaustive overview of several common gradientbased and derivative-free optimization algorithms with some explanation on their basic working principles.

3.4.1 Gradient-Based Optimization Approach

In gradient-based optimization, an iterative search procedure is established in which the search direction towards a local minimum of a differentiable objective function is defined by the gradient of the objective function at the current point [100]. At every point a, the (multi-variable) objective function f_x decreases fastest if the next point from a goes in the direction of the negative gradient of f at a, denoted as $-\nabla f(a)$. In an iterative way, point a converges to a local minimum. This method is described as a 'steepest descent', which is a basic first order method that is typically slow converging and scale-sensitive [101]. In the remainder of this section, more advanced derivative-based methods are described. Fig. 3.2 represents a graphical example of a gradient-based optimization that converges towards a local minimum with iterations i_0 - i_4 .


Figure 3.2: Graphical example of a gradient-based optimization approach that converges towards a local minimum with iterations i_0 - i_4

In order to choose the right gradient-based optimization algorithm for a practical problem, one should know how to classify these algorithms in terms of mathematical optimization problem forms and their accompanying optimization limitations. Examples of different gradient-based optimization classes are Non-Linear Programming (NLP) for differentiable objective functions. Linear Programming (LP) for affine objective functions (see 'simplex method' or 'interior point method'), Quadratic Programming (QP) for affine constraint functions and linear-quadratic objective functions [45]. It is believed by [102] that Sequential Quadratic Programming (SQP) is the state-of-the-art in gradient-based optimization. Concerning the problem nature, there is a great watershed between convex and non-convex optimization problems. The former can be solved effectively thanks to the absence of local optima, whereas the solution of non-convex optimization problems is complicated by the presence of local optima, non-optimal stationary points or disconnected feasible sets. An efficient method to arrive at a local minimum for convex problems is to search according to the so-called Newton direction. This direction is derived from the second-order Taylor series approximation of the objective function. Methods using the Newton direction have a fast rate of local convergence. The main drawback of the Newton direction is the need for the Hessian $\nabla^2 f(x)$. Explicit computation of this matrix of second derivatives is sometimes a cumbersome, error-prone and computationally expensive process. Quasi-Newton search directions provide an attractive alternative in that they do not require computation of the Hessian and yet still attain a linear rate of convergence. Instead of the true Hessian, an approximation is used which is updated after each step to take account of the additional knowledge gained during the previous step [45].

Articles applying a gradient-based algorithm or describing a new method include e.g. the Linear Programming (LP) 'simplex method' [103], Sequential Quadratic Programming (SQP) [104] or Convex Optimization (CO) [105]. An advantage of the gradient-based algorithms is that they often need less time to converge to an optimal solution [38]. Another advantage is that the solutions are deterministic, which means that the same solution is obtained at different algorithm runs with the same starting conditions. There are also some important limitations of these methods. First, constraints usually result in convergence in a local optimum, being an optimal point in the neighborhood around the found solution. This local optimum is not necessarily a global optimum, being the optimal point in the complete set. Converging to a local minimum instead of a global minimum also causes the optimization results to be highly dependent on the chosen starting point [102]. Second, gradient-based optimization algorithms are not applicable on non-smooth objective functions, as it is impossible to predict the behavior of the objective function near the point(s) of non-smoothness. That is, no information on the objective function obtained at one point can be used to infer anything about the objective function at neighboring points, because points of non-differentiability may intervene [45]. A convexification-based approach for non-convex problems only works for some specific cases, resulting in limited applicability [106]. Additionally, it is often not directly possible to apply integer constraints to (part of) the design parameters. In general, integer constraints can only be applied in derivative-free optimizations [79, 107].

The objective function of the general co-design methodology in this dissertation exhibits discontinuous behavior. This means that derivative or gradient information generally cannot be used to determine the direction in which the objective function is increasing (or decreasing) [108]. Dividing the objective into continuous sub-problems or convexifying the objective function to apply gradient-based algorithms would lead to a method that is too problem-specific and require deep insights by the end-user. Therefore, it can be concluded that gradient-based optimization algorithms are not preferable for the presented co-design optimization methodology.

3.4.2 Derivative-Free Optimization Approach

In contrast to the gradient-based approach, the derivative-free (also called gradientfree) algorithms are less vulnerable to black-box type problems and are generally better at handling non-linearities and discontinuities [109,110]. Another advantage is that they provide the ability to simultaneously cope with integer and real values, or so-called mixed-integer programming, without the need of relaxations in the constraints [111]. Furthermore, derivative-free methods are often the best globalsearching algorithms as they sample a large portion of the design space [112].

A drawback of these methods is that they are computationally expensive. However, due to the increase in computing power over the past decades, the use of these optimization routines becomes more appealing to obtain a solution within a reasonable time. Another drawback of certain derivative-free optimization approaches is that they are stochastic and thus non-deterministic. They may yield different solutions on different runs, even when started from the same point on the same model, depending on which points are randomly sampled [108]. In the next subsections, some examples of commonly used derivative-free algorithms are briefly explained.

Exhaustive Search (ES)

The simplest and most inefficient way to optimize a problem is to test every possible solution and choose the best one. This method is only applicable to problems with a minimal design space and quickly becomes impossible to apply to larger problems.

Pattern Search (PS)

Pattern Search (PS) (also known as Direct Search (DS)) optimization methods solve optimization problems iteratively by computing a set of points around a current point at every step. These points are called the 'coordinates' and determine in which direction an improvement of the objective function is obtained. If the algorithm finds a coordinate that lowers the objective function, than this point becomes the new current point at the next iteration step of the algorithm. If none of the coordinates show a fitness value improvement, then a global solution is sought by decreasing the mesh size [113–115].

An adaptation of a Pattern Search algorithm is Coordinate Descent (CD), in which the mesh size is determined by performing a line search [116]. In this way, it can be compared with the gradient-based methods, in which every Coordinate Descent iteration does not converge in the direction of steepest descent as in gradient-based methods, but instead in the most favorable direction according to the coordinates. This means that in the case of Coordinate Descent, no derivatives of the objective function are needed. In accordance with the gradient-based optimization methods, the coordinate descent method generally also has problems with non-smooth objective functions [117]. An example of a Coordinate Descent optimiza-



Figure 3.3: Illustrative example of a Coordinate Descent (CD) optimization method with meshes in blue applied to a Rosenbrock function with initial point x_0 (reproduced from [118])

tion method applied to a Rosenbrock function with initial point $x_0 = (-3, -4)$ is seen in Fig. 3.3. The Coordinate Descent (CD) method reaches the optimum after 22231 function evaluations.

An improved version of the Coordinate Descent (CD) is the Adaptive Coordinate Descent (ACD), in which the coordinates are not fixed, but are gradually transformed so that the new coordinates correlate as little as possible with the objective function [119]. Fig. 3.4 shows the application of an Adaptive Coordinate Descent (ACD) applied to a Rosenbrock function with the same initial point $x_0 = (-3, -4)$ as in the example with a Coordinate Descent (CD). The Adaptive Coordinate Descent (ACD) reaches the same point after only 325 function evaluations, or about 70 times faster than the Coordinate Descent (CD) method, which is comparable to gradient-based optimizations [118].

Pattern Search, Coordinate Descent and Adaptive Coordinate Descent have the disadvantage that they cannot inherently cope with mixed-integer constraints, making them unsuited algorithms for the co-design optimization problem of this dissertation.



Figure 3.4: Illustrative example of an Adaptive Coordinate Descent (ACD) optimization method with meshes in blue applied to a Rosenbrock function with initial point x_0 (reproduced from [118])

Bayesian Optimization

In Bayesian statistics, a 'prior' (short for 'prior probability distribution') of an uncertain quantity is the probability distribution that would express one's beliefs about this quantity before some evidence of this quantity is taken into account. In Bayesian Optimization, the objective function is treated as a random function and a prior is placed over it. The prior is updated after the objective function evaluations to form a posterior distribution, from which a next point can be extracted by using an acquisition function. The development of Bayesian Optimization is primarily attributable to Jonas Mockus from a series of publications in the 1970s and 1980s [120–122].

The Bayesian Optimization approach originates from statistics and appears to be used primarily for statistical and machine learning purposes. However, it has the potential to be used to a wide assortment of applications, ranging from computer graphics and visual design [123], autonomous vehicles [124] to automatic machine learning toolboxes [125]. However, Bayesian Optimization cannot guarantee to find a global optimum [126] and is practically limited to optimizing up to 20 design parameters [127, 128], which is too little for applying the co-design methodology presented in this PhD.

Dividing Rectangles (DIRECT)

DIRECT or Dividing Rectangles is a global optimization algorithm developed by Donald R. Jones [129]. The algorithm begins by scaling the design space to an n-dimensional unit hypercube. DIRECT initiates its search by evaluating the objective function at the center point of the hypercube. DIRECT then trisects this hyperrectangle and samples the center points of the other two resulting hyperrectangles. From here, DIRECT selects and further trisects the optimal hyperrectangles. This division process continues until a prespecified iteration limit is reached or until convergence is achieved. An example of the division of rectangles in the first three iterations for a two-dimensional problem is illustrated in Fig.3.5, in which the blue rectangles represent the optimal rectangles selected for division in that particular iteration. DIRECT selects larger rectangles and smaller rectangles with better objective function values to balance between the local and global search [112].

Direct Multisearch (DMS) is an technique developed by extending DI-RECT from single to multi-objective optimization, able to deal with a any type of constraints. It uses the concept of Pareto dominance to maintain a list of non-dominated points from which the new iterates or poll centers are chosen [111].



Figure 3.5: Example of the first three iterations of a DIRECT Algorithm for a two-dimensional problem in which the blue rectangles represent the optimal rectangles selected for division in that particular iteration (reproduced from [129])

Simulated Annealing (SA)

Simulated Annealing (SA) is a stochastic (thus non-deterministic) algorithm, which means that they follow a random path in finding the global optimum. Simulated Annealing algorithm is analogous to the process of annealing of metals. When metals are at a high temperature, the atoms can move relatively freely, but as the temperature is decreased slowly, the atom movements get restricted and start adopting the most stable orientation by taking the lowest possible energy state. Attaining the lowest possible energy state can be thought of as reaching the global minimum in the optimization process. The algorithm starts at a random design point. From this design point, the algorithm jumps to a new random design point and evaluates the objective function value and feasibility. If the current point is better than the previous, then the current point is accepted to be a potentially optimal point and if the current point is worse than the previous point then its acceptance or rejection depends on the Metropolis probability criterion given by equation (3.1), with P the Metropolis probability, f the objective function evaluations, and t the temperature.

$$P(f,t) = e^{\frac{f_{new} - f_{current}}{t}}$$
(3.1)

From the equation above, it can be seen that the new design point is more likely to be accepted if its function value is close to the objective function of the current design parameters. Moreover, the probability of acceptance is higher when the temperature t is high. The algorithm may accept a new design point which is worse than the current one. It is this feature that prevents the method from becoming stuck in a local minimum. The Simulated Annealing (SA) algorithm does a global search initially when the temperature is high and even worse design points are more likely to be accepted. The algorithm then switches to local search when temperature is decreased and worse design points are less likely to be selected. Thus, the switching from the global search to local search depends on the value of the temperature. This process of selection is continued as the temperature is decreased at each iteration. The stopping criterion for this non-deterministic optimization technique is defined by the number of prespecified iterations. In general, Simulated Annealing (SA) is not suitable for multi-objective optimization problems [112, 130, 131].

Surrogate Optimization

In Surrogate Optimization (SO) a surrogate (= approximation) of the objective function is used, with the advantage that the surrogate takes less time to evaluate. Therefore, Surrogate Optimization (SO) is useful when the optimization problem has a computationally expensive objective function. The objective function can be non-smooth, but the algorithm works best for a continuous objective function. The algorithm inherently makes a balance between two goals: speed and exploration. Speed is needed to obtain a good solution in few objective evaluations, while ex-

ploration is used to search for a global minimum. Surrogate Optimization (SO) converges to a global minimum, but convergence is slow [132, 133].

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is one of the many evolutionary optimization algorithms in which biological processes like mutation and selection are used as search operators. The operators are used in a loop, of which one iteration is called a generation. The iterative calculation of subsequent generations is continued until a termination criterion is met [134–136]. Particle Swarm Optimization (PSO) solves a problem by having a population of candidate solutions, also called 'particles,' and moving these particles around in the search-space according to simple mathematical formulas over the particle's position and velocity. Each particle movement is influenced by its own fitness value but is also guided towards the best-known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm towards the best solutions [137]. PSO is a heuristic algorithm that can search very large spaces of candidate solutions. However, heuristic algorithms such as PSO do not guarantee an optimal solution is ever found [138]. There are many different variants of evolutionary algorithms, each of which shows differences in the processes modeled upon biological processes. Some other examples of Evolutionary Algorithms (EA) are Cuckoo Search [139], Ant Colony Optimization [140], and Artificial Bee Colony Algorithm [141, 142].

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is another evolutionary-based optimization algorithm broadly based on the principle of biological evolution. Within this fairly new evolutionary strategy, new candidate solutions are defined according to a multivariate normal distribution. Recombination consists of selecting a new mean value for the distribution, while mutation consists of adding a random vector with zero mean. Pairwise dependencies between the variables in the distribution are represented by a covariance matrix. The covariance matrix adaptation (CMA) is a method to update the covariance matrix of this distribution [143]. Furthermore, local gradient information can be taken into account during optimization, even though the objective function is not differentiable in all locations [144]. Fig. 3.6 illustrates an example of the CMA-ES optimization procedure on a simple two-dimensional problem. The spherical optimization landscape is depicted with solid lines of equal fitness values. On this simple problem, the population concentrates over the global optimum within a few generations [145].

3.4 Optimization Algorithm Selection



Figure 3.6: Example of a CMA-ES optimization procedure on a simple twodimensional problem (reproduced from [146]). (a) Generate solutions (black). (b) Rank the solutions and keep the ones with the best fitness values. (c) Compute the evolution paths (green) and update the covariance matrix (blue) accordingly. (d) Move the center of the mutation distribution (purple) to the weighted mean. (e) Update the step size. (f) Generate solutions for the next generation.

Genetic Algorithm

Genetic Algorithms (GA) are a group of optimization algorithms based on evolutionary processes and Darwin's concept of natural selection [112]. It starts by (randomly) generating an initial population of individual solutions utilizing a creation function. Every solution consists of a combination of proposed design parameters. For every individual solution, a fitness value is calculated using the objective function (often referred to as 'fitness function'), resulting in a definite score of how well the individual performs. When assigning a fitness value, the constraints are also taken into consideration. Individual solutions that do not meet certain constraints are penalized with a much higher fitness value as a result.

Based on these fitness values, GA then applies three different operations to create a new generation of individuals, being elite selection, crossover and mutation. Elite selection implies that individuals with the best fitness have the most significant probability of being directly selected to pass on to the next generation, unchanged. Crossover (or recombination) is where (parts of) the design parameters of two individuals are exchanged to get two new individuals. The crossover step is a unique factor to distinguish a Genetic Algorithm (GA) from other evolutionary optimization strategies. Mutation is where design parameters of one individual from the initial population are randomly changed to get a new individual. The crossover fraction indicates the ratio between crossover and mutation. The above process is executed to generate a new generation of individuals and is continued to further improve the fitness of the best individual until a stopping criterion is met [112]. Possible stopping criteria are related to a maximum calculation time, a maximum number of generations or a minimum amount of change in the average fitness values over a number of successive generations. The workflow of the algorithm is schematically displayed in Figure 3.7.

An advantage of the Genetic Algorithm is that there are no strict requirements on the objective function (such as convexity, linearity, and continuity). This means that there exist a large amount of freedom in the scoring of each individual in the objective function. In this way, the individual's fitness value can be calculated according to (a weighted sum of) several objectives, such as settling time, reference tracking, energy consumption or robustness. This fitness function can be non-linear and even discontinuous. The implementation of discontinuous (and thus non-linear) constraints can easily be programmed in a constraint function. In Genetic Algorithms, there is no convexification necessary as in gradient-based optimization algorithms [147]. Moreover, there are no restrictions regarding the maximum number of design parameters in the case of a Genetic Algorithm.

A disadvantage of the Genetic Algorithm is that a lot of objective function evaluations are needed, resulting in a relatively long calculation time compared to gradient-based methods [148]. Because this is also a non-deterministic algorithm, it cannot be guaranteed that the solution found by the Genetic Algorithm is the global minimum of the optimization problem [149]. However, the randomization avoids being stuck in a local minimum as much as possible [150].



Figure 3.7: Simplified schematic overview of a Genetic Algorithm work-flow

3.4.3 Conclusions on the Optimization Algorithm Selection

Table 3.1 summarizes the various mentioned optimization algorithms and their main characteristics based on the literature review in this chapter.

Concerning the derivative-free optimization algorithms, the Exhaustive Search, Bayesian Optimization, Genetic Algorithm, and Surrogate Optimization are the only optimization algorithms that inherently support mixed-integer optimization [151]. Exhaustive Search and Bayesian Optimizations cannot be applied in this work because they can only be applied to problems with few design parameters. From the remaining Genetic Algorithm and Surrogate Optimization, the Genetic Algorithm is the only optimization algorithm that straighforwardly supports nonlinear constraints [152]. Consequently, it can be concluded that a Genetic Algorithm (GA) is the derivative-free algorithm that is most suited to administer the requirements for the optimization problem discussed in this dissertation. The main drawbacks of using a Genetic Algorithm (GA) are that it is computationally expensive and that it inherently cannot guarantee to find the global optimum. Therefore, the end-user may not know how far the final solution is from the actual global optimum. However, literature shows that even non-global solutions have better performances compared to using classical iterative methods to find the system composition. Whether the results are 'good enough' is certainly problem-dependent [153].

Optimization algorithms		Deterministic/ stochastic?	Objective type	Guaranteed convergence to global optimum?	Supports integer constraints?	Computation expense	Additional remarks
Gradient- based	Non-linear Programming (NLP)	Deterministic	Convex	•	0	Mid	For differentiable objective functions
	Linear Programming (LP)	Deterministic	Convex	•	0	Low	For affine objective functions and constraints
	Quadratic Programming (QP)	Deterministic	Convex	•	0	Low	For affine constraint functions and linear-quadratic objective functions
	Sequential Quadratic Programming (QP)	Deterministic	Convex	•	0	Mid	
Derivative- free	Exhaustive Search (ES)	Deterministic	Non-smooth	•	•	Very high	Only for problems with a very small design space
	Pattern Search (PS)	Deterministic	Non-smooth	0	0	High	Preferably smooth objective functions
	Bayesian optimization	Stochastic	Non-smooth	0	•	Very high	Limited to small-scale problems
	Dividing Rectangles (DIRECT)	Deterministic	Non-smooth	0	0	Very high	
	Simulated Annealing (SA)	Stochastic	Non-smooth	•	0	High	Converges to global optimum for bounded problems with very slow cooling schedule
	Surrogate Optimization (SO)	Stochastic	Non-smooth	•	•	Very high	For time-consuming objective functions, does not support non-linear constraints
	Particle Swarm Optimization (PSO)	Stochastic	Non-smooth	0	0	High	
	Covariance Matrix Adaptation Evolution Strategy (CMA-ES)	Stochastic	Non-smooth	0	0	High	Ability to include local second-order derivative information
	Genetic Algorithm (GA)	Stochastic	Non-smooth	0	•	High	Supports non-linear constraint and objective functions

Table 3.1: Summary of the various mentioned optimization algorithms and their main characteristics based on the literature review in this chapter

3.5 Pareto Front

For the overall co-design methodology, cost and performance are conflicting objectives without a unique optimal solution. Instead, the concept of Pareto optimality can be used to characterize the trade-off between different objectives. Pareto optimality is obtained when one objective of an individual solution cannot be improved without another objective deteriorating. Such an individual solution is then described as a Pareto point. All Pareto points collectively form a Pareto front that indicates the boundary of the Pareto optimality [154,155]. In the context of this dissertation, the main results from the optimizing algorithm can be graphically shown in a Pareto front, revealing the trade-off between the total implementation cost and the optimal achievable performance.

A general example of a Pareto front is given in Fig. 3.8. As stated before, the lower the fitness value, the better the performance. Every variation of the design parameters leads to a specific system composition with a certain cost and performance. If all these individual points are shown in a graph, than an individual point is said to be on the Pareto front if no objective can be improved without worsening another objective [156]. If, for example, a system design is operating with a suboptimal performance (red point in Fig. 3.8), two different actions can be taken to obtain Pareto optimality: performance can be increased without an extra cost (arrow 1) or the investment cost can be reduced without deterioration in performance (arrow 2).



Figure 3.8: Example of a Pareto front (in blue) showing the interplay between two conflicting objectives. To obtain Pareto optimality from a sub-optimal solution (red point), two different actions can be taken: performance can be increased without an extra cost (arrow 1), or the investment cost can be reduced without deterioration in performance (arrow 2).

In the context of this dissertation, a Pareto front shows the maximum achievable performance in function of the cost. This is a very useful tool to graphically provide insights for the design engineer into the trade-off between performance and cost, which is otherwise very complicated to predict. The Pareto front gives a set of reasonable choices, but any choice from the Pareto front still entails a trade-off between conflicting objectives. By restricting the feasible solutions to those who are at the Pareto front, a designer can make more efficient trade-offs, rather than considering the full design space of every design parameter. But it is still a question of engineering judgment to select a point along the Pareto front as the ultimate solution [40, 157].

3.6 Background on State-Space Modeling

The general co-design methodology described in this dissertation starts with a model that describes the dynamics of the system to be controlled. More and more, engineering relies on model-based design. The ever increasing computing power makes it possible to accurately simulate the machine behavior and thus design a machine without the need for expensive prototypes or time-consuming tests on physical objects [25]. Over the past decades, the models have also become more and more complex and inclusive in order to obtain a simulation of a (set of) machines as close as possible to the operation of a real system.

To arrive at a model representing the behavior of a process, an engineer can start by mapping the individual physical relations in differential equations. From these, many alternative model formats can be used to characterize dynamic processes (e.g., Frequency Response Functions (FRF), Transfer Functions (TF) or Polynomial Models (PM)). For complex systems, especially multiple-input multiple-output (MIMO) problems [158], one of the most flexible and useful structures for numerical calculations is the state-space representation.

3.6.1 Continuous LTI State-Space Modeling

In this representation, the process dynamics are described in a set of coupled firstorder differential equations, possibly linearized around an operating point [36, 52, 159, 160]. A continuous (open-loop) state-space representation $SS_{pro,c}$ is used to represent these differential equations of a linear time-invariant (LTI) process in matrix form. This structure is formulated in (3.2), with *n* being the number of states in the state vector **x**, *m* being the number of outputs in the output vector **y**, and *l* being the maximum number of available actuator inputs from input vector **u**. The $A_{pro,c}$, $B_{pro,c}$, $C_{pro,c}$ and $D_{pro,c}$ matrices are referred to as the continuous time-domain process state matrix, input matrix, output matrix and feedthrough (or direct transition) matrix, respectively [52].

$$\mathbf{SS}_{pro,c} = \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}_{pro,c} \cdot \mathbf{x}(t) + \mathbf{B}_{pro,c} \cdot \mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}_{pro,c} \cdot \mathbf{x}(t) + \mathbf{D}_{pro,c} \cdot \mathbf{u}(t) \end{cases}, \\ \text{with } \mathbf{x} \in \mathbb{R}^{n} , \mathbf{u} \in \mathbb{R}^{l} , \mathbf{y} \in \mathbb{R}^{m} \quad (3.2) \end{cases}$$

A schematic overview of a continuous LTI state-space system is depicted in Fig. 3.9.



Figure 3.9: Schematic overview of a continuous state-space process

3.6.2 Continuous State-Space Observer Modeling

In some applications, one or more states may not be directly accessible because it is too expensive to install sensors and measure them. If the state variables are not available because of system configuration or cost, it is possible to estimate the states by using an observer (or sometimes called state estimator). The estimated variables are denoted by a 'hat' as in $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ to distinguish them from the process variables \mathbf{x} and \mathbf{y} . The continuous observer state equations are depicted in (3.3).

$$\mathbf{SS}_{obs,c} = \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}_{obs,c} \cdot \mathbf{\hat{x}}(t) + \mathbf{B}_{obs,c} \cdot \mathbf{u}(t) + \mathbf{L}_{obs,c}(\mathbf{y}(t) - \mathbf{\hat{y}}(t)) \\ \mathbf{\hat{y}}(t) = \mathbf{C}_{obs,c} \cdot \mathbf{\hat{x}}(t) + \mathbf{D}_{obs,c} \cdot \mathbf{u}(t) \\ & \text{with } \mathbf{\hat{x}} \in \mathbb{R}^{n} , \ \mathbf{u} \in \mathbb{R}^{l} , \ \mathbf{\hat{y}} \in \mathbb{R}^{m} \end{cases}$$
(3.3)

Fig. 3.10 shows a state-space representation in the continuous time-domain of a process (light blue) with an observer (dark blue). An additional observer gain $\mathbf{L}_{obs,c}$ is added. The difference between the process output $\mathbf{y}(t)$ and the estimated output $\hat{\mathbf{y}}(t)$ is multiplied by this matrix $\mathbf{L}_{obs,c}$ and subsequently added to the observer state equations. In this way, a so-called Leuenberger observer is obtained [161]. The observer is asymptotically stable if the error between the estimated states $\hat{\mathbf{x}}(t)$ and the process states $\mathbf{x}(t)$ converges to zero for $t \to \infty$. This condition is fulfilled when the matrix $\mathbf{A}_{obs,c} - \mathbf{L}_{obs,c} \mathbf{C}_{obs,c}$ has stable eigenvalues.

3.6 Background on State-Space Modeling

As a rule of thumb, the poles of the observer $\mathbf{A}_{obs,c} - \mathbf{L}_{obs,c} \mathbf{C}_{obs,c}$ are chosen to converge ten times faster than the process poles, characterized by multiplying the real parts of the process poles by a factor of 10 [161]. In theory, the \mathbf{A}_{pro} , \mathbf{B}_{pro} , \mathbf{C}_{pro} and \mathbf{D}_{pro} process matrices are the same as the respective observation matrices \mathbf{A}_{obs} , \mathbf{B}_{obs} , \mathbf{C}_{obs} and \mathbf{D}_{obs} , but due to unavoidable modelling errors, there will practically always be a (preferably small) difference. A more advanced form of estimation can be obtained by applying a Kalman filter [162], but the use of this filter is not further developed in this work and is also not mandatory to demonstrate the operation of the general co-design methodology. More information on state observer design can be found in [163].



Figure 3.10: Schematic overview of a continuous state-space process (light blue) with observer (dark blue)

3.6.3 Continuous LPV and LTV State-Space Modeling

In some systems, a linear time-invariant (LTI) system with a linearization around a single operating point is insufficient to model the dynamic behavior of the system. This is the case when certain parameter variations have a drastic influence on the dynamics of the system. The variability of certain process parameters can be taken into account by formulating them explicitly as a variable in the state-space system. In this way, linear parameter-varying (LPV) systems are obtained.

A classic example of an LPV system is a hoisting crane, depicted in Fig. 3.11 below. If an external force F is applied (e.g., by the wind), the mass m will start to swing. The natural frequency f of the mass m suspended from a cable is:

$$f = \frac{1}{2\pi} \sqrt{\frac{g}{l}}.$$
(3.4)

The gravitational acceleration g is assumed to remain constant. As long as the cable length l does not change, this system can be simplified as a linear timeinvariant (LTI) state-space system. However, an essential part of the operation of this crane is that the cable length l changes. Additionally, it can be seen that the natural frequency f varies depending on the cable length l. Thus, an LTI state-space representation will not be sufficient to correctly represent the machine operation, and a more advanced form of model (and control) is needed.



Figure 3.11: Hoisting crane overview with cable length l, cable angle α , mass m and external force F (from [164])

3.6 Background on State-Space Modeling

A possible strategy to address this problem is called gain scheduling, in which the system is linearized at different operating points depending on a changing parameter. This changing parameter p(t) is called the 'scheduling parameter' or 'scheduling variable'. Next, different linear controllers are designed for each operating point, and these controllers are then scheduled as a function of the scheduling parameter p(t). In this way, a non-linear controller is obtained for a non-linear system by combining multiple linear controllers. The active controller values are determined by switching different linear controllers or by interpolation between them [165].

A linear parameter-varying (LPV) state-space representation allows to model one or more changing parameters directly into the system. In this way, the cumbersome method of defining different linear controllers in the case of gain scheduling is no longer necessary. The scheduling parameter p(t) should not be confused with the standard system inputs u(t), as this scheduling parameter p(t) affects the system input-output relationship [164]. In short, linear parameter-varying (LPV) systems are linear systems, but with state-space descriptions that are a function of time-varying parameters p(t). The time variations of the scheduling parameter p(t) are not necessarily known in advance, but are measurable (or estimated) during operation. The state-space system is shown in (3.5), which is similar to the linear time-invariant (LTI) case, but with the inclusion of the scheduling parameter p(t) [166, 167].

$$\mathbf{SS}_{LPV,c} = \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}_{pro,c}(p(t)) \cdot \mathbf{x}(t) + \mathbf{B}_{pro,c}(p(t)) \cdot \mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}_{pro,c}(p(t)) \cdot \mathbf{x}(t) + \mathbf{D}_{pro,c}(p(t)) \cdot \mathbf{u}(t) \\ & \text{with } \mathbf{x} \in \mathbb{R}^n , \ \mathbf{u} \in \mathbb{R}^l , \ \mathbf{y} \in \mathbb{R}^m \quad (3.5) \end{cases}$$

A variation of the linear parameter-varying (LPV) system is a so-called linear time-varying (LTV) system, in which the scheduling parameter p(t) only depends on time and not on other real-time measured values. It is important to note that the general co-design methodology described in this dissertation can handle linear time-invariant (LTI), linear parameter-varying (LPV), and linear time-variant (LTV) systems. As of next subsection, an LTI system is used as the process model, while in Section 6.4 on page 153 the co-design methodology applied on an LPV system is demonstrated.

3.6.4 Illustrative LTI case: Mechanical Synchronization Setup

Throughout the remainder of this dissertation, references are made to a mechanical synchronization setup to explain the proposed co-design methodology. The main purpose of this illustrative example is to position a central load inertia using an actuator (with inertia) at each side of the central inertia. Common examples of this type of application are steel ladles and overhead gantry cranes. First of all, a model is established in which all possible actuator locations are included. A graphical overview of the application is given in Fig. 3.12, with θ the angular displacement [rad], T_1 the actuator 1 torque [Nm], T_2 the actuator 2 torque [Nm], T_L the load torque [Nm], k the torsional spring constant [Nm/rad] and b the torsional damping constant [Nms/rad]. The three bodies (both motors and the load) are dynamically coupled by rotational springs and dampers (k and b), as depicted in Fig. 3.12. The values for these parameters are shown in Table 3.2.



Figure 3.12: A mechanical synchronization application in which two actuator inertias have to control the position of a central load inertia (top), with schematic representation (bottom)

Parameter		Value	Unit
Actuator 1 inertia	$= J_1$	0,173	kgm^2
Actuator 2 inertia	$= J_2$	0,173	kgm^2
Load inertia	$= J_L$	3	kgm^2
Torsional spring constant 1	$= k_1$	150	Nm/rad
Torsional spring constant 2	$= k_2$	150	Nm/rad
Angular damping constant 1	$= b_1$	0.2	Nms/rad
Angular damping constant 2	$= b_2$	0.2	Nms/rad

 Table 3.2: System parameters for the mechanical synchronization application

The differential equations are shown in (3.6).

$$\begin{cases} \ddot{\theta}_{1}J_{1} = T_{1} - k_{1}(\theta_{1} - \theta_{L}) - b_{1}(\dot{\theta}_{1} - \dot{\theta}_{L}) \\ \ddot{\theta}_{2}J_{2} = T_{2} - k_{2}(\theta_{2} - \theta_{L}) - b_{2}(\dot{\theta}_{2} - \dot{\theta}_{L}) \\ \ddot{\theta}_{L}J_{L} = -T_{L} - k_{1}(\theta_{L} - \theta_{1}) - b_{1}(\dot{\theta}_{L} - \dot{\theta}_{1}) - k_{2}(\theta_{L} - \theta_{2}) - b_{2}(\dot{\theta}_{L} - \dot{\theta}_{2}) \\ \end{cases}$$
(3.6)

After rearranging, (3.7) is obtained.

$$\begin{cases} \ddot{\theta}_{1} = \frac{T_{1}}{J_{1}} - \frac{k_{1}}{J_{1}}\theta_{1} + \frac{k_{1}}{J_{1}}\theta_{L} - \frac{b_{1}}{J_{1}}\dot{\theta}_{1} + \frac{b_{1}}{J_{1}}\dot{\theta}_{L} \\ \ddot{\theta}_{2} = \frac{T_{2}}{J_{2}} - \frac{k_{2}}{J_{2}}\theta_{2} + \frac{k_{2}}{J_{2}}\theta_{L} - \frac{b_{2}}{J_{2}}\dot{\theta}_{2} + \frac{b_{2}}{J_{2}}\dot{\theta}_{L} \\ \ddot{\theta}_{L} = -\frac{T_{L}}{J_{L}} + \frac{k_{1}}{J_{L}}\theta_{1} + \frac{k_{2}}{J_{L}}\theta_{2} - \frac{k_{2}+k_{1}}{J_{L}}\theta_{L} + \frac{b_{1}}{J_{L}}\dot{\theta}_{1} + \frac{b_{2}}{J_{L}}\dot{\theta}_{2} - \frac{b_{1}+b_{2}}{J_{L}}\dot{\theta}_{L} \end{cases}$$
(3.7)

The states, outputs and inputs for the state-space representation for this openloop LTI process are shown in (3.8), (3.9), and (3.10), respectively. The process inputs u [Nm] are controller effort signals, while *Dist* [Nm] is the external disturbance signal acting on the load inertia.

$$\mathbf{x}_{pro} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_L \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_L \end{bmatrix}$$
(3.8)
$$\mathbf{y}_{pro} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_L \end{bmatrix}$$
(3.9)

$$\mathbf{u}_{pro} = \begin{bmatrix} u_1 \\ u_2 \\ Dist_L \end{bmatrix}$$
(3.10)

The corresponding continuous linear time-invariant (LTI) state-space matrices for the open-loop process can be found in (3.11) - (3.14).

$$\mathbf{A}_{pro,c} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{-k_1}{J_1} & 0 & \frac{k_1}{J_1} & \frac{-b_1}{J_1} & 0 & \frac{b_1}{J_1} \\ 0 & \frac{-k_2}{J_2} & \frac{k_2}{J_2} & 0 & \frac{-b_2}{J_2} & \frac{b_2}{J_2} \\ \frac{k_1}{J_L} & \frac{k_2}{J_L} & \frac{-(k_1 + k_2)}{J_L} & \frac{b_1}{J_L} & \frac{b_2}{J_L} & \frac{-(b_1 + b_2)}{J_L} \end{bmatrix}$$
(3.11)
$$\mathbf{B}_{pro,c} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{J_1} & 0 & 0 \\ 0 & \frac{1}{J_2} & 0 \\ 0 & 0 & \frac{1}{-J_L} \end{bmatrix}$$
(3.12)
$$\mathbf{C}_{pro,c} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$
(3.13)

$$\mathbf{D}_{pro,c} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
(3.14)

The reference angular position trajectory Ref to be followed by the load inertia J_L can be seen in Fig. 3.13 (a) and consists of a trapezoidal movement to an angle of 1.8 radians or approximately 103 degrees. The shape of the associated load torque T_L can be seen in Fig. 3.13 (b). The shape of the reference signal and the associated disturbance signal originates from an example of a steel ladle application in which a casting angle of 1.8 radians is attained to pour out the molten metal.



Figure 3.13: (a) Reference trajectory Ref for the angular displacement to be followed by the load inertia J_L . (b) Associated load torque T_L .

3.7 Chapter Conclusions

This chapter introduces the general workflow to perform the co-design of hardware architecture and control configuration. First, a global overview is presented of the various steps to achieve a successful and efficient co-design. This overview can be seen as a framework for the next chapters, in which each chapter goes into detail on a particular part of the co-design.

The conflicting objectives of the overall co-design problem are discussed, and the necessary requirements that an optimization algorithm must meet in order to carry out the intended co-design of hardware and control are listed. This is followed by an introduction to multi-objective optimization algorithms, explaining the distinction between gradient-based and derivative-free optimization approaches. Numerous examples are discussed to provide an overview of the most common optimization techniques. Eventually, the conclusion is that a Genetic Algorithm (GA) is the only optimization technique that can cope with the requirements of the co-design optimization problem considered in this dissertation. This is followed by an explanation of how to graphically present the optimization results in a Pareto front.

A necessary condition to perform this co-design is the need for a mathematical model of the machine or system. Therefore, the chapter ends with an overview of several types of model representations, of which a state-space representation is the most appropriate one. Continuous linear time-invariant (LTI), linear parameter-varying (LPV), and linear time-varying (LTV) state-space systems are considered in more detail, all of which can be used in the overall co-design methodology. The use of an observer in state-space representations is also mentioned, and an illustrative state-space model is presented that will be used throughout this dissertation to demonstrate specific parts of the co-design.

Chapter 4

Hardware Architecture Optimization

This chapter elaborates on how the hardware architecture can be optimized as part of the system composition optimization (see Fig. 4.1). First, it is discussed how the presence or absence of actuators and sensors can be established (= actuator and sensor placement). It is then explained how different types of actuators and sensors can be specified (= actuator and sensor selection). This chapter ends with an open-loop analysis, enabling the determination of the hardware combinations' feasibility without having to calculate a closed-loop system and response.



Figure 4.1: Overview of the different aspects of the total system composition optimization with the focus of this chapter marked in blue

4.1 Actuator & Sensor Placement

In Section 3.6.1 on page 59, a state-space representation for a linear time-invariant (LTI) system is introduced. To define the actuator presence or absence, actuator placement binaries' $b_{act,...} \in [0, 1]$ are used for every possible actuator location. These binaries are grouped in a diagonal matrix \mathbb{B}_{Θ} :

$$\mathbb{B}_{\Theta} = \mathbf{I}_{l} \circ \begin{bmatrix} b_{act,1} \\ b_{act,2} \\ \vdots \\ b_{act,l} \end{bmatrix}$$

$$= \begin{bmatrix} b_{act,1} & 0 & \cdots & 0 \\ 0 & b_{act,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{act,l} \end{bmatrix},$$
(4.1)

where I_l is an identity matrix with dimensions $l \times l$, with l being the number of actuator inputs. This matrix I_l is multiplied with an array of the actuator placement binaries using an element-wise array multiplication (\circ). As a consequence, \mathbb{B}_{Θ} is a diagonal matrix with dimensions $l \times l$. Equation (4.2) shows how these binaries $b_{act,...}$ define whether an actuator will affect the LTI state-space system input or not. The actuators possibly introduce mass dynamics properties into the system, which are reflected in the system matrices. Therefore, the system matrix $\mathbf{A}_{pro,c}$, $\mathbf{B}_{pro,c}$, $\mathbf{C}_{pro,c}$ and $\mathbf{D}_{pro,c}$ are adjusted to $\mathbf{A}'_{pro,c}$, $\mathbf{B}'_{pro,c}$, $\mathbf{C}'_{pro,c}$ and $\mathbf{D}'_{pro,c}$ based on the actuator placement binaries and the accompanying change in dynamics due to the included or excluded actuators. An example in mechatronics of adjusting system matrix $\mathbf{A}_{pro,c}$ to $\mathbf{A}'_{pro,c}$ is where the influence of an actuator inertia (with possible corresponding torsional stiffness k and damping b) is removed from the system matrices depend on the state equations and cannot be presented straightforwardly in one formula. An example of adjusting the state matrices is given in Section 4.2.1.

$$\begin{split} \dot{\mathbf{x}} &= \mathbf{A}'_{pro,c} \cdot \mathbf{x} + \mathbf{B}'_{pro,c} \cdot \mathbb{B}_{\mathbf{\Theta}} \cdot \mathbf{u} \\ &= \begin{bmatrix} A'_{1,1} & A'_{1,2} & \cdots & A'_{1,n} \\ A'_{2,1} & A'_{2,2} & \cdots & A'_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A'_{n,1} & A'_{n,2} & \cdots & A'_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \dots \\ \begin{bmatrix} B'_{1,1} & B'_{1,2} & \cdots & B'_{1,l} \\ B'_{2,1} & B'_{2,2} & \cdots & B'_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ B'_{n,1} & B'_{n,2} & \cdots & B'_{n,l} \end{bmatrix} \begin{bmatrix} b_{act,1} & 0 & \cdots & 0 \\ 0 & b_{act,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{act,l} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_l \end{bmatrix} \\ &= \begin{bmatrix} A'_{1,1} & A'_{1,2} & \cdots & A'_{1,n} \\ A'_{2,1} & A'_{2,2} & \cdots & A'_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A'_{n,1} & A'_{n,2} & \cdots & A'_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} B'_{1,1} & B'_{1,2} & \cdots & B'_{1,l} \\ B'_{2,1} & B'_{2,2} & \cdots & B'_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ B'_{n,1} & B'_{n,2} & \cdots & B'_{n,l} \end{bmatrix} \begin{bmatrix} b_{act,1}u_1 \\ b_{act,2}u_2 \\ \vdots \\ b_{act,l}u_l \end{bmatrix}$$
(4.2)

Similarly, the presence or absence of a physical sensor is represented using sensor placement binaries $b_{sen,...} \in [0, 1]$. The sensor placement binaries are grouped in a diagonal matrix \mathbb{B}_{Γ} :

$$\mathbb{B}_{\Gamma} = \mathbf{I}_{m} \circ \begin{bmatrix} b_{sen,1} \\ b_{sen,2} \\ \vdots \\ b_{sen,m} \end{bmatrix}$$

$$= \begin{bmatrix} b_{sen,1} & 0 & \cdots & 0 \\ 0 & b_{sen,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{sen,m} \end{bmatrix},$$

$$(4.3)$$

where \mathbf{I}_m is an identity matrix with dimensions $m \times m$, with m being the number of sensor outputs. This matrix \mathbf{I}_m is multiplied with an array of the sensor placement binaries using an element-wise array multiplication (\circ). As a consequence, \mathbb{B}_{Γ} is a diagonal matrix with dimensions $m \times m$. Conventionally, the output vector of a state-space system is appointed as \mathbf{y} . The sparse rows of \mathbb{B}_{Γ} are removed to prevent the system from having outputs that remain zero. By doing so, \mathbb{B}'_{Γ} is obtained with dimensions $m' \times m$, where m' depends on the number of activated sensors. In this way, \mathbb{B}'_{Γ} determines which of the original state-space system output signals are available to be used as feedback signals, in correspondence with

which sensors are active or not. Consequently, (4.4) shows how binaries $b_{sen,...}$ define what signals from the output equation are available for the feedback loop, indicated as \mathbf{y}' . If system matrix $\mathbf{D}'_{pro,c}$ is not a zero matrix (which in continuous state-space systems rarely occurs), the previously mentioned actuator placement binaries $b_{act,...}$ also have an impact on \mathbf{y}' in the form of \mathbb{B}_{Θ} .

$$\begin{aligned} \mathbf{y}' &= \mathbb{B}'_{\Gamma} \cdot \mathbf{y} \\ &= \mathbb{B}'_{\Gamma} \left(\mathbf{C}'_{pro,c} \cdot \mathbf{x} + \mathbf{D}'_{pro,c} \cdot \mathbb{B}_{\Theta} \cdot \mathbf{u} \right) \\ &= \begin{bmatrix} b_{sen,1} & 0 & \cdots & 0 \\ 0 & b_{sen,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{sen,m} \end{bmatrix} \cdot \left(\begin{bmatrix} C'_{1,1} & C'_{1,2} & \cdots & C'_{1,n} \\ C'_{2,1} & C'_{2,2} & \cdots & C'_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C'_{m,1} & C'_{m,2} & \cdots & C'_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \dots \\ \begin{bmatrix} D'_{1,1} & D'_{1,2} & \cdots & D'_{1,l} \\ D'_{2,1} & D'_{2,2} & \cdots & D'_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ D'_{m,1} & D'_{m,2} & \cdots & D'_{m,l} \end{bmatrix} \begin{bmatrix} b_{act,1}u_1 \\ b_{act,2}u_2 \\ \vdots \\ b_{act,l}u_l \end{bmatrix} \end{aligned} \right) \end{aligned}$$

$$(4.4)$$

In this way, the basic state-space representation from (3.2) is extended to (4.5). Note that the **B** and **C** matrix values are not design parameters for the general optimization algorithm but are constant real values describing the initial state-space process. The hardware placement optimization is done through \mathbb{B}_{Θ} for the actuator placement binaries $b_{act,...}$ and \mathbb{B}_{Γ} for the sensor placement binaries $b_{sen,...}$. This state-space system is graphically displayed in Fig. 4.2.

$$\mathbf{SS}_{pro,c} = \begin{cases} \dot{\mathbf{x}} = \mathbf{A}'_{pro,c} \cdot \mathbf{x} + \mathbf{B}'_{pro,c} \cdot \mathbb{B}_{\Theta} \cdot \mathbf{u} \\ \mathbf{y}' = \mathbb{B}'_{\Gamma} \cdot (\mathbf{C}'_{pro,c} \cdot \mathbf{x} + \mathbf{D}'_{pro,c} \cdot \mathbb{B}_{\Theta} \cdot \mathbf{u}) \end{cases}$$
(4.5)



Figure 4.2: Graphical representation of a state-space system, augmented with matrices \mathbb{B}_{Θ} and \mathbb{B}'_{Γ} defining the actuator and sensor placement, respectively see (4.5)

4.2 Actuator & Sensor Selection

Next to the presence or absence of the actuators and sensors, the overall co-design optimizing algorithm should also be able to select between different types of actuators and sensors. Therefore, hardware selection integer values $i_{act,...}$ and $i_{sen,...}$ are introduced and grouped in vectors \mathbb{Z}_{act} and \mathbb{Z}_{sen} , respectively. They correspond with the different types of actuators and sensors at every active actuator or sensor position. This is mathematically shown in (4.6) and (4.7) for the actuator and sensor selection integers, respectively.

$$\mathbb{Z}_{act} = [i_{act,1}, i_{act,2}, \dots, i_{act,l}], \quad \forall i_{act,\dots} \in [0:p],$$

with l = number of actuator inputs, and
 p = number of possible actuator types (4.6)

$$\mathbb{Z}_{sen} = [i_{sen,1}, i_{sen,2}, \dots, i_{sen,m}], \quad \forall i_{sen,\dots} \in [0:q]$$
with $m =$ number of sensor outputs, and
$$q =$$
 number of possible sensor types
$$(4.7)$$

If multiple actuator types are possible, then the optimizing algorithm can effectively choose between each actuator and sensor type with specific properties and constraints (e.g., different cost, maximum actuator output, or resolution) by changing the corresponding actuator and sensor integer values. For example, a more expensive actuator can have a higher maximum control effort. Based on this integer value, the accompanying properties are considered as non-linear constraints during the general co-design optimization (further detailed in Chapter 6).

Notice that the actuator and sensor selection integers grouped (\mathbb{Z}_{act} and \mathbb{Z}_{sen}) are not directly part of the state-space system, but they may have an indirect influence on this state-space system because values for the actuator and sensor placement binaries (\mathbb{B}_{Θ} and \mathbb{B}_{Γ}) depend on these actuator and sensor selection integer values. Furthermore, the selection of the type of actuators can also have an influence on the adjustment of the system matrices $\mathbf{A}_{pro,c}$, $\mathbf{B}_{pro,c}$, $\mathbf{C}_{pro,c}$, and $\mathbf{D}_{pro,c}$, respectively, if different actuators have different inertia values. For example, if it is defined that a different type of actuator has a different inertia value $J = f(i_{act})$, then this inertia value will have to be adjusted in the system matrices.

The values of the actuator and sensor selection integers will directly determine the actuator and sensor placement binaries. If an actuator or sensor selection integer is equal to zero, this corresponds to the absence of this actuator or sensor, and the corresponding actuator or sensor placement binary will also be equal to zero. If an actuator or sensor selection integer is at least equal to one, this will result in the corresponding actuator or sensor presence, and the associated placement binary will therefore be equal to one. The sensor and actuator selection integers are design parameters for the overall co-design methodology. As a result, this co-design methodology becomes a mixed-integer optimization problem. Additionally, the objective function becomes highly discontinuous and non-convex by introducing these discrete design parameters [86].

4.2.1 Mechanical Synchronization Setup

For the illustrative example of a mechanical synchronization case (introduced in Section 3.6.4 on page 64), the general co-design optimizing algorithm can select the presence and type of the two actuators at the sides of the load inertia from a list of three possible motors, as listed in Table 4.1. Every motor has a specific cost and a corresponding maximum actuator torque. This changing maximum actuator torque can be included in the co-design methodology as a saturation of the input signal during the system response calculation (see Chapter 5 for more information). As mentioned earlier, these costs are entirely case-specific. Therefore, all costs are indicated with a percentage (%) instead of an actual cost, relative to the most expensive possible situation. In this mechanical synchronization case, all costs are indicated relative to a setup where the two most expensive actuators, the most expensive sensors and all possibilities of control architecture are active.

The actuator selection integers $i_{act,1}$ and $i_{act,2}$ for the left and right actuator can have values from zero to three, according to the selected actuator. If the actuator selection integer $i_{act,...}$ is equal to zero, no actuator is present on that location, and the corresponding actuator placement binary $b_{act,...}$ also equals zero. The actuator selection integers $i_{act,1}$ and $i_{act,2}$ are grouped in \mathbb{Z}_{act} .

	Actuator	Actuator		Maximum
Actuator	selection	placement	Cost	torque
selection	integer $i_{act,}$	binary $b_{act,}$	[%]	[Nm]
No actuator	0	0	0	0
Actuator type 1	1	1	5.3	1
Actuator type 2	2	1	13.3	3
Actuator type 3	3	1	26.7	5

Table 4.1: Different actuator types with their accompanying actuator selection integers, actuator placement binaries, cost, and maximum torque

As explained above, the absence of an actuator may also mean that the system matrices need to be adjusted. For example, suppose that an actuator is present at the left side of the central load inertia (meaning that $i_{act,1} \ge 1$ and thus $b_{act,1} = 1$) and no actuator is present at the right side of the central inertia (meaning that $i_{act,2} = 0$ and thus $b_{act,2} = 0$). In this case, the initial system matrices $\mathbf{A}_{pro,c}$ and $\mathbf{B}_{pro,c}$ (see (3.11)–(3.12)) are adjusted to:

$$\mathbf{A}_{pro,c}' = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-k_1}{J_1} & 0 & \frac{k_1}{J_1} & \frac{-b_1}{J_1} & 0 & \frac{b_1}{J_1} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{k_1}{J_L} & 0 & \frac{-k_1}{J_L} & \frac{b_1}{J_L} & 0 & \frac{-b_1}{J_L} \end{bmatrix}$$
(4.8)

and

$$\begin{bmatrix} \overline{J_L} & 0 & \overline{J_L} & \overline{J_L} & 0 & \overline{J_L} \end{bmatrix}$$
$$\mathbf{B}'_{pro,c} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{J_1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{J_L} \end{bmatrix}.$$
(4.9)

The initial state matrices $\mathbf{C}_{pro,c}$ and $\mathbf{D}_{pro,c}$ do not need to be adjusted because they do not contain actuator specific values.

In the mechanical synchronization setup, there is a maximum of three inertias. This means that three possible sensor locations can provide position and speed feedback from the corresponding inertia. It is more convenient to add a sensor on the actuator inertias since most actuators already provide relatively easy mounting capabilities for encoders. This is why a sensor on the load inertia is more expensive than a sensor on the outer inertias. Table 4.2 shows the possible sensor types and the accompanying cost. Each sensor is assumed to feed back angular displacement and velocity. As a result, the sensor selection integers $i_{sen,1}$ and $i_{sen,2}$ for the left and right actuator can each have integer values from zero to two. The sensor selection integer is zero if no sensor is used, one if a sensor on the corresponding actuator inertia is used, and two if a sensor on the middle inertia is used. If the sensor selection integer $i_{sen,...}$ is equal to zero, no sensor is present, and the accompanying sensor placement binary $b_{sen,...}$ also equals zero. The sensor selection integers $i_{sen,1}$ and $i_{sen,2}$ are grouped in \mathbb{Z}_{sen} .

 Table 4.2: Different sensor types with their accompanying sensor selection integers, sensor placement binaries, and cost

	Sensor	Sensor	
Sensor	selection	placement	Cost
selection	integer i_{sen}	binary b _{sen}	[%]
No sensor data used	0	0	0
Sensor on corresponding actuator	1	1	6.7
Sensor on load inertia	2	1	9.3

4.3 Open-Loop Analysis

The number of possible different open-loop hardware configurations w is determined as:

$$w = 2^{l+m},$$
 (4.10)

with *l* and *m* being the number of actuator and sensor placement binaries, respectively. With modern-day computing power, it would be possible to perform an exhaustive search (also called 'brute-force') calculation of every possible openloop hardware configuration with a fixed control loop. However, in the co-design methodology presented in this PhD the control loop is not fixed but simultaneously optimized with the hardware architecture. The real controller tuning parameters \mathbb{R}_{tuning} to be optimized result an optimization problem with an design space that takes on proportions that cannot be calculated with an exhaustive search technique. (This is even without considering the control architecture optimization, mentioned in Chapter 5.)

Obviously, certain combinations of actuators and sensors will lead to an infeasible situation, even without considering the control configuration. For example, it is easy to see that if all actuator integers are equal to zero, the system will not exhibit good performance since then the actuator placement binaries will also equal zero, and no actuators are present in the setup. In those cases, it is unnecessary to let the optimization algorithm lose time trying to calculate a closed-loop response. This can be avoided by using an open-loop analysis, as presented in this section.

Moreover, there are different degrees of interconnections in systems consisting of multiple subsystems, varying from very strong to non-existent. A subsystem without an actuator can still be sufficiently controlled by actuating another strongly coupled subsystem. However, it is less evident for large and complex systems to immediately see how strong the coupling is between individual subsystems and whether the entire system can be controlled correctly. An example is given in Fig. 4.3, in which a system is shown consisting of multiple interconnected subsystems. Some of these subsystems are controlled by an actuator, while the subsystem interconnections are displayed as arrows. Black arrows represent a strong connection, while blue dashed arrows represent weak connections. Subsystem three is not directly actuated, but due to the strong interconnections with actuated subsystems two and four, this subsystem can still be controlled to a desired state. However, there is no actuator directly connected to subsystem four, and this subsystem does not have sufficiently strong interconnections with actuated subsystems. Because of this, the state of subsystem four cannot be controlled.

The extent to which the system input u(t) can steer the given initial system states $\mathbf{x}(t_0)$ to a desired final state value in finite time is referred to as 'controllability' (also known as 'reachability'), introduced by Rudolph Kalman in the 1960s [52]. If it is indeed possible to steer all system states from any initial value to a desired final value within finite time, the system is said to be controllable. Otherwise, the system is uncontrollable [163].



Figure 4.3: Graphical overview of a system consisting of multiple interconnected subsystems. Actuators are present to control specific subsystems, while the strong and weak subsystem interconnections are shown in black and dashed blue arrows, respectively.

There are several tests to check the state controllability of a system, see [36] for details. In brief, a first possible test is done by checking the input pole vector from a specific expansion of the input-output transfer function matrix [168]. A second possible controllability check is based on the rank of the Gramian matrix. A third and most suitable possible controllability check is to define if an n-th order multi-input system with state equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4.11}$$

is completely controllable if the matrix

$$\mathbf{C}_{\mathbf{M}} = \begin{bmatrix} \mathbf{B} & \mathbf{A}\mathbf{B} & \mathbf{A}^{2}\mathbf{B} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}$$
(4.12)

is of rank n, where C_M is called the controllability matrix [163]. (The rank of a matrix is defined by the number of linearly independent matrix columns or rows [169].)

Similarly, the concept of 'observability' indicates the ability to deduce the state variables x from a knowledge of the input u(t) and the output y(t). In other words,

a system is observable if the initial state vector $\mathbf{x}(t_0)$ can be found from the input u(t) and the output y(t), measured over a finite time interval from t_0 . Otherwise, the system is said to be unobservable [163].

Again, there are several tests to check the state observability of a system [36], which are variations of the controllability checks. Once again, the most suitable observability check is to define if an n-th order multi-input system with state and output equations

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$
 (4.13)

is completely observable if the matrix

$$\mathbf{O}_{\mathbf{M}} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A} \\ \mathbf{C}\mathbf{A}^{2} \\ \vdots \\ \mathbf{C}\mathbf{A}^{n-1} \end{bmatrix}$$
(4.14)

is of rank n, where O_M is called the observability matrix [163]. When a system is observable, this means that an observer (see Section 3.6.2 on page 60 for details) can be used to correctly estimate the progress of the system states based on the measured system outputs.

In literature, definitions for other terms related to observability and controllability can be found. For example, there also exists a 'stabilizability' condition, indicating that the state variables remain bounded during the system behavior [170]. Another variation is 'detectability,' stating that all the unobservable states remain stable [171]. However, these extensions are not further addressed in this work but could potentially be implemented in future work. More information on the system controllability, observability and related variations with accompanying proofs can be found in [36, 52, 160, 172].

Testing the controllability and observability of all w different (open-loop) statespace systems can be done by comparing the rank of respectively the controllability matrix and the observability matrix with the system order. Mind that changing system matrices in function of the active actuators and sensors must be taken into account. In the absence of specific actuators, certain inertias may no longer be applicable in the system state equations and the state-space system must be adjusted accordingly. Calculating this matrix rank is not computationally intensive, so this method can be used to relatively quickly perform an open-loop analysis to determine which hardware architectures are controllable and observable, and thus feasible. The workflow of this open-loop analysis is graphically shown in Fig. 4.4.

The valuable information obtained during this open-loop analysis can be stored to be used later in the co-design methodology. More specifically, due to this openloop analysis, it can be decided during the iterative optimization algorithm not to



Figure 4.4: Graphical overview of the open-loop state-space analysis to determine infeasible open-loop systems

calculate the closed-loop system and its response if the open-loop system already appears to be infeasible.

The total time gain by applying this open-loop analysis will not be the same for every setup, but will depend on the percentage of infeasible open-loop systems of all possible w open-loop systems. The larger the percentage of infeasible open-loop systems, the more the optimization algorithm can avoid unnecessarily calculating the closed-loop systems with responses, and thus the more significant the total time gain will be.

4.3.1 Mechanical Synchronization Setup

Next, the open-loop analysis is performed on the illustrative example of a synchronization setup, introduced in Section 3.6.4 on page 64. The number of actuator placement binaries is two, being $b_{act,1}$ and $b_{act,1}$ for the left and right inertia, respectively. The number of sensor placement binaries is three, being $b_{sen,1}$, $b_{sen,2}$, and $b_{sen,L}$, corresponding with sensor measurements on the left, right and central load inertia. Therefore, w = 5 and $2^w = 2^5 = 32$ different hardware architectures are possible. The open-loop analysis determines which of the hardware architectures are controllable and observable (as discussed in the previous subsection).

When setting up the state-space system, external disturbance inputs can be defined in the system input matrix \mathbf{u}_{pro} . These inputs cannot be controlled, and should not be taken into account when determining the system controllability. Otherwise it would be possible for a system to be appointed as controllable on the basis of a non-controllable input.

Note that the system matrices for the open-loop analysis change in function of the active actuators and sensors. Since the open-loop analysis can determine the controllability and observability with respect to system matrices $\mathbf{A}_{pro,c}$, $\mathbf{B}_{pro,c}$ and $\mathbf{C}_{pro,c}$, the influence of the presence of actuators and sensors on matrix $\mathbf{D}_{pro,c}$ is not important in the open-loop analysis. If sensors are not active, the corresponding rows in the matrix $\mathbf{C}_{pro,c}$ are set to zero. In this mechanical synchronization example, the first, second and third row of matrix $\mathbf{C}_{pro,c}$ are set to zero if a sensor at, respectively, the inertia 1, inertia 2 and load inertia position is not present.

For defining the absence of actuators in the open-loop analysis, it is not sufficient to set the corresponding rows of the $\mathbf{A}_{pro,c}$ and $\mathbf{B}_{pro,c}$ matrix to zero, because then the state of the corresponding inertia will still affect the calculation of the controllability and observability (via the rank of the controllability matrix and observability matrix), while it is actually supposed to be removed from the system. Therefore, in the absence of an actuator on inertia 1 ($b_{act,1} = 0$), for example, the corresponding state equations are removed, resulting in a state-space system with the following matrices:

$$\mathbf{x}_{pro} = \begin{bmatrix} \theta_2\\ \theta_L\\ \dot{\theta}_2\\ \dot{\theta}_L \end{bmatrix}, \qquad (4.15)$$

$$\mathbf{y}_{pro} = \begin{bmatrix} \theta_2\\ \theta_L \end{bmatrix}, \tag{4.16}$$

$$\mathbf{u}_{pro} = u_2, \tag{4.17}$$

$$\mathbf{A}_{pro,c} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-k_2}{J_2} & \frac{k_2}{J_2} & \frac{-b_2}{J_2} & \frac{b_2}{J_2} \\ \frac{k_2}{J_L} & \frac{-k_2}{J_L} & \frac{b_2}{J_L} & \frac{-b_2}{J_L} \end{bmatrix},$$
(4.18)
4.3 Open-Loop Analysis

$$\mathbf{B}_{pro,c} = \begin{bmatrix} 0 & 0\\ 0 & 0\\ \frac{1}{J_2} & 0\\ 0 & \frac{-1}{J_L} \end{bmatrix},$$
(4.19)

and

$$\mathbf{C}_{pro,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$
 (4.20)

Similarly, the absence of an actuator on inertia 2 ($b_{act,2} = 0$) results in a state-space system with the following matrices:

$$\mathbf{x}_{pro} = \begin{bmatrix} \theta_1 \\ \theta_L \\ \dot{\theta}_1 \\ \dot{\theta}_L \end{bmatrix}, \qquad (4.21)$$

$$\mathbf{y}_{pro} = \begin{bmatrix} \theta_1 \\ \theta_L \end{bmatrix}, \tag{4.22}$$

$$\mathbf{u}_{pro} = u_1, \tag{4.23}$$

$$\mathbf{A}_{pro,c} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-k_1}{J_1} & \frac{k_1}{J_1} & \frac{-b_1}{J_1} & \frac{b_1}{J_1} \\ \frac{k_1}{J_L} & \frac{-k_1}{J_L} & \frac{b_1}{J_L} & \frac{-b_1}{J_L} \end{bmatrix}, \qquad (4.24)$$
$$\mathbf{B}_{pro,c} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{J_1} & 0 \\ 0 & \frac{-1}{J_L} \end{bmatrix}, \qquad (4.25)$$

and

$$\mathbf{C}_{pro,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$
 (4.26)

The results of this open-loop analysis are shown in Table 4.3. A hardware architecture is considered feasible if it is both controllable and observable. Therefore, the open-loop analysis shows that this mechanical synchronization setup is only feasible if there is at least one actuator present with a sensor on the actuated inertia or the central load inertia.

Table 4.3: Table showing all possible combinations of the hardware placement binaries with corresponding controllability, observability, and feasibility for the example of a mechanical synchronization setup. See Fig. 3.12 for a graphical overview of the setup.

							Feasible
$b_{act,1}$	$b_{act,2}$	$b_{sen,1}$	$b_{sen,2}$	$b_{sen,L}$	Controllable?	Observable?	hardware
,	,		,	ŕ			architecture?
0	0	0	0	0	0	0	0
0	0	0	0	1	0	•	0
0	0	0	1	0	0	0	0
0	0	0	1	1	0	•	0
0	0	1	0	0	0	0	0
0	0	1	0	1	0	•	0
0	0	1	1	0	0	0	0
0	0	1	1	1	0	•	0
0	1	0	0	0	•	0	0
0	1	0	0	1	•	•	•
0	1	0	1	0	•	•	•
0	1	0	1	1	•	•	•
0	1	1	0	0	•	0	0
0	1	1	0	1	•	•	•
0	1	1	1	0	•	•	•
0	1	1	1	1	•	•	•
1	0	0	0	0	•	0	0
1	0	0	0	1	•	•	•
1	0	0	1	0	•	0	0
1	0	0	1	1	•	•	•
1	0	1	0	0	•	•	•
1	0	1	0	1	•	•	•
1	0	1	1	0	•	•	•
1	0	1	1	1	•	•	•
1	1	0	0	0	•	0	0
1	1	0	0	1	•	•	•
1	1	0	1	0	•	•	•
1	1	0	1	1	•	•	•
1	1	1	0	0	•	•	•
1	1	1	0	1	•	•	•
1	1	1	1	0	•	•	•
1	1	1	1	1	•	•	•

The calculation of the open-loop analysis on this mechanical synchronization example only took 0.25s using MATLAB® scripts on an Intel® Xeon® CPU @ 3.10 GHz with 64 GB of RAM. The limited calculation time confirms the computational efficiency of the open-loop analysis. Next, the influence of this open-loop analysis on the required time for the complete co-design methodology is tested. The co-design optimization algorithm uses a non-deterministic Genetic Algoritm, which means that repeated optimization runs do not always yield precisely the same results. Therefore, the co-design optimization for this mechanical synchronization example is done several times, both with and without open-loop analysis. For these optimizations, a Genetic Algorithm is applied with 43 design parameters, containing nine binary values ($\in \mathbb{B}$), two integer values ($\in \mathbb{Z}$) and 32 real numbers ($\in \mathbb{R}$), see Chapter 6. The Genetic Algorithm uses a population size of 200 individuals, an elite count of 10 and a crossover fraction of 0.4. The stopping criteria is defined as an average cumulative change in the fitness function value over 50 generations that is less than 10^{-5} . More details on applying a Genetic Algorithm for the co-design optimization can be found in Chapter 6.

Table 4.4 shows the fitness value and the time needed to obtain a solution for multiple optimization runs, both with and without incorporating the open-loop analysis. The resulting fitness values for all 14 optimizations range from 30.655 to 30.672. The negligible differences among the results demonstrate that the proposed open-loop analysis does not affect the obtained solution. In addition, the averages of the total computation time for the different runs demonstrate that a considerable time reduction of 22% is obtained if the open-loop analysis is applied.

	Without open	-loop analysis	With open-loop analysis	
	Fitness value	Total time [s]	Fitness value	Total time $[s]$
Optimization run 1	30.672	336.67	30.656	283.91
Optimization run 2	30.657	372.99	30.661	261.40
Optimization run 3	30.655	341.24	30.656	244.24
Optimization run 4	30.658	359.42	30.662	390.62
Optimization run 5	30.668	377.01	30.666	284.48
Optimization run 6	30.661	328.97	30.656	231.61
Optimization run 7	30.657	405.97	30.659	276.00
Average values	30.661	360.32	30.659	281.75

Table 4.4: Table showing the fitness value and the total calculation time over several optimization runs, both with and without open-loop analysis. The average values show that a time gain of 22% is achieved when applying the open-loop analysis.

4.4 Chapter Conclusions

This chapter explains how the presence and absence of actuators and sensors can be defined in a state-space system using actuator and sensor placement binaries $(b_{act,...} \text{ and } b_{sen,...})$ through diagonal matrices \mathbb{B}_{Θ} and \mathbb{B}_{Γ} , respectively. It is also explained how different types of actuators and sensors can be represented using actuator and sensor selection integers $\mathbb{Z}_{act,...}$ and $\mathbb{Z}_{sen,...}$. The values of the hardware placement binaries $(b_{act,...} \text{ and } b_{sen,...})$ depend on the chosen hardware selection integers $\mathbb{Z}_{act,...}$ and $\mathbb{Z}_{sen,...}$. In turn, these hardware selection integers are variables that can be adjusted by the overall optimization algorithm. In this way, the hardware architecture, being the hardware placement and selection, can be described in a mathematical formulation that can be used by an optimization algorithm.

The various combinations of the hardware placement binaries also define the different open-loop systems. An open-loop analysis is performed on each of these open-loop systems to determine its feasibility without having to calculate a closed-loop system and response. This information can be used to run the general co-design optimization algorithm more efficiently. Incorporating this open-loop analysis in the co-design optimization results in a case-specific, yet substantial total calculation time reduction, without influencing the obtained result.

Naturally, the calculation time to perform an open-loop analysis will increase as it is applied to systems with a higher number of possible hardware combinations. However, it is clear that applying the open-loop analysis results in extensive time savings in executing the overall co-design methodology.

Chapter 5

Control Configuration Optimization

In this chapter, the control configuration is discussed in more detail as part of the hardware architecture and control configuration co-design (see Fig. 5.1). The control configuration optimization consists of the control architecture optimization on the one hand and the controller tuning optimization on the other hand. First, it is explained how the control architecture optimization can be performed with a large freedom in its design space. A number of concepts and necessary notations are provided that enable this control architecture optimization.

Furthermore, it is mentioned that next to the control architecture, also the controller tuning optimization can be performed as part of the control configuration optimization. In the scope of this control configuration optimization, a closed-loop state-space methodology is established to reformulate an open-loop state-space system with an extensive control architecture into one closed-loop state-space system. This allows the compact and consistent modeling of a system with a wide variety of control loops and enables the fast calculation of the corresponding closedloop response.

Finally, the findings of this chapter are also applied to a practical mechanical synchronization example.



Figure 5.1: Overview of the different aspects of the total system composition optimization with the focus of this chapter marked in blue

5.1 Control Architecture Optimization

5.1.1 Background on Control Architecture Topologies

There are three general topologies concerning the control architecture for systems consisting of multiple subsystems, being centralized, decentralized, and distributed control (see Figure 5.2 for a graphical overview). With centralized control, the entire system is manipulated using only one central controller. This also means that the same controller is used to control all system inputs u in a desired and logical way based on all the available system outputs y. The manual fine-tuning and precise diagnosis of a completely centralized controller quickly becomes infeasible as the size and number of subsystems increase.

Conversely, in decentralized control, a multi-variable process is decomposed into multiple individual single-input-single-output (SISO) processes, and the controllers are designed for each individual loop without taking the loop interactions into account. So far, decentralized control is still the dominant control scheme in multi-variable control because it has many advantages, such as flexibility in operation, failure tolerance, and simplified design and tuning. Decentralized control can generally work well when loop interactions are modest. However, when the processes are closely coupled, it inevitably leads to performance degradation compared to centralized control schemes, as the tuning of the controller is aimed at a compromise between improving system performance and conquering loop interactions [173].

Distributed control is placed somewhere in between entirely centralized and decentralized control. In this control topology, the control tasks are mapped on different processing units actuating on different subsystems, while control data is transferred between the controllers using a communication system [30].

In [174], the differences between centralized and decentralized control for the Airbus A320 longitudinal and lateral dynamics are investigated. When applying the centralized method, the aircraft takes a longer time to minimize the oscillation response and get the airplane into a stable position compared to using two controllers to control each aircraft motion (longitudinal and lateral) individually. In that example, decentralized control is preferred over centralized control, while distributed control is not examined. The optimal control architecture was also examined for a spatial six-degree-of-freedom electro-hydraulic parallel robot [175], as seen in Fig. 5.3. The physical dynamic coupling effect existing in such a 6-DOF parallel robot can limit or degrade the performance since the coupling may restrict the further improvement of control performances of various control schemes and the development of the potential of such robot. Eliminating or reducing the coupling of 6-DOF electro-hydraulic parallel robots can improve system control accuracy and trajectory tracking performance in all DOFs. However, the reduction of decoupling is not always straightforward and mostly application-dependent.

In [176], an overview and classification are provided of decentralized and distributed control topologies for large-scale systems. The design of distributed con-



(b)



(c)



Figure 5.2: Schematic overview of a system consisting of four interacting subsystems (a) with one centralized controller C, (b) with four decentralized controllers C_1-C_4 , one for each subsystem, (c) with distributed controllers C_1-C_4 . The communication system used for the information exchange between the distributed controllers is highlighted in blue.



Figure 5.3: Configuration of a 6-DOF parallel robot (from [175])

trollers for a class of systems consisting of several identical subsystems was detailed in [177]. The work in that paper can only be applied if the system's statespace matrices satisfy a specific structural property. Similarly, [178] considers a distributed linear-quadratic regulator (LQR) framework for a network of identical dynamically decoupled multi-agent systems. In [179], a distributed Model Predictive Control (MPC) method was presented and compared with centralized and decentralized MPC for a sextuple water tank system. Results show that the optimal control method depends on the extent to which the model is identified and whether reference tracking or disturbance rejection is preferred.

In the previous examples, different control architectures are considered and compared. The conclusions show that the optimal control architecture is application-dependent and that, therefore, not one particular control architecture can be designated as superior. It also follows that the influence of the control architecture on the final performance of a system is difficult to estimate in advance. The cited papers compare different control architectures by running them separately and then comparing the results, but so far, there is no easy way to automatically optimize a system's control architecture. As far as the co-design of hardware architecture and control configuration is concerned, in current literature, only the controller tuning is optimized while the control architecture is always There are, however, many control architecture features that assumed fixed. are very often used in industrial motion control and for which (to the author's knowledge) no other work is found that considers changing control architectures during optimization. This is a substantial disadvantage of current literature as the optimization of the control architecture has the potential to generate a considerable gain in performance. The general co-design optimization algorithm presented in this PhD dissertation allows an optimization algorithm to consider and apply

different control architectures. Thus it allows to simultaneously optimize the control architecture (together with the controller tuning) as part of the control configuration optimization. This results in a very powerful tool capable of performing a profound co-design of hardware architecture and control configuration. Hereafter follows an overview of different control architecture features that can be optimized as part of the general hardware and control co-design methodology.

5.1.2 Decentralized/Distributed Control Architecture Features

In literature, the description of control loops in the state-space domain is mostly limited to an output or state feedback controller [36, 159], with or without integral reference tracking [163], or an observer to estimate non-measurable states [52, 160]. Common examples of feedback control design methods in the statespace formulation are linear-quadratic regulator (LOR) [56, 180, 181], H₂ [36, 148] or robust H_{∞} [62, 182, 183] control. These methods have been intensively researched but are mostly limited to determining a single feedback loop only. Unfortunately, state-space feedback controllers and their individual matrix element gains are complicated to assess qualitatively, making it a challenging task to conduct user fine-tuning and diagnosis. Therefore, multiple interacting Single-Input Single-Output (SISO) control loops are still preferred by many industrial users in modern motion control applications. Examples of these commonly used control architecture features are decentralized and distributed cascaded loops, feedforward control, and synchronizing control [184-186]. Next, more background information is given of how these different control features are implemented in feedback control loops.

Decentralized controllers are the most basic feedback controllers and are part of a feedback loop acting on a system input based on the difference between the desired system value and a measured (or observed) system output. Moreover, distributed controllers can be used to allow interaction between different control loops. In the scope of this work, the subscripts of the decentralized controllers indicate which control loop they are part of, while the subscripts of the distributed controllers indicate between which control loops the interaction is active. Therefore, the subscripts of decentralized controllers will consist of one part, while the subscripts of decentralized controllers will be two-part. For example, a decentralized controller that is part of a feedback loop acting on system input 3 will be denoted as C_3 , while a distributed controller that has its controller inputs from a subsystem 9 to control a subsystem 7 control loop will be denoted as C_{9-7} . Fig. 5.4 shows a general example of a system with two inputs u_1 and u_2 , controlled by decentralized controllers C_1 and C_2 and distributed controllers C_{1-2} and C_{2-1} , with system outputs y_1 and y_2 , setpoints r_1 and r_2 , and controller inputs e_1 and e_2 .



Figure 5.4: Block diagram for a general control architecture with decentralized controllers C_1 and C_2 , distributed controllers C_{1-2} and C_{2-1} , inputs u_1 and u_2 , system outputs y_1 and y_2 , setpoints r_1 and r_2 , and controller inputs e_1 and e_2

Decentralized Control Variation: Cascaded Control

A first mentioned variation of decentralized control is cascaded control, in which multiple SISO (Single-Input Single-Output) controllers are connected one after another. In this case, the outer controller generates a control signal that serves as the setpoint for the following controller. In the scope of this work, the cascade levels are indicated with letters. The letter *a* represents the most inner loop cascade level, followed by the letters *b*, *c*, ... for the subsequent outer cascade levels. This is graphically shown in Fig. 5.5, with r_1 and r_2 being the outer and inner loop setpoints, y_1 , y_2 being the plant outputs, *u* being the plant input, and C_a and C_b being the cascade level *a* and *b* controllers.

This cascade control is commonly used in the process control where, for example, an outer loop flow controller controls an inner loop valve orifice controller [187]. It is also a common way of controlling a motion application's position, where an outer and an inner controller are connected one after another. Typically, the inner loops have faster dynamics than the outer loops. That is why in motion control applications, the inner and outer controllers are often speed and position controllers, respectively. The outer position controller generates a control signal that serves as the setpoint for the inner speed controller [36].



Figure 5.5: Block diagram for a general cascade control architecture, with r_1 and r_2 being the outer and inner loop setpoints, y_1 , y_2 being plant outputs, u being the plant input, and C_a and C_b being the cascade level a and b controllers

Decentralized Control Variation: Feedforward Control

In addition to cascaded control, feedforward control is another commonly used variation of decentralized control. This control variation is achieved by pre-commanding the system depending on a reference signal or a disturbance signal, without using any feedback from the system itself. As a result, reference feedforward or disturbance feedforward can be obtained. See Fig. 5.6 for a general example of a reference feedforward control loop, with r being the reference signal, y being the system output, C_{ff} and C_{fb} being the feedforward and feedback controllers, u_{ff} and u_{fb} being the feedforward and feedback controller outputs, and u being the system input.

In process control, for example, a feedforward control can be used to immediately adjust a valve to a changing flow demand without using a feedback loop [187]. For motion applications, a reference feedforward control can clearly reduce the tracking error and thus improve the trajectory tracking by forwarding the derivative(s) of the reference signal parallel with the controllers, multiplied with a specific feedforward gain [36]. Feedforward controllers C_{ff} for motion applications often consist of a derivative of the reference, multiplied with a single proportional gain K_{ff} .



Figure 5.6: Block diagram for a general reference feedforward control, with r being the reference signal, y being the system output, C_{ff} and C_{fb} being the feedforward and feedback controllers, u_{ff} and u_{fb} being the feedforward and feedback controller outputs and u being the system input

Distributed Control Variation: Synchronizing Control

Finally, synchronizing control can be used as a distributed control architecture extension to improve output synchronization between coupled subsystems that are both parts of a more extensive system. Thereby, the synchronous control of subsystem outputs is improved by taking into account the output deviations. This is graphically shown in Fig. 5.7, with r being the setpoint for both system outputs y_1 and y_2 , C_{sc} being the synchronizing controller, e_1 and e_2 being the inputs for the decentralized controllers C_1 and C_2 with controller outputs u_1 and u_2 , d_1 and d_2 being disturbances and u being the system input.

Synchronizing control has been widely used in, for example, robotics, electric vehicles, steel rolling, and paper-making [188]. This synchronizing control can be applied by adding separate controllers (e.g., P/PI/PID) acting on the relative fault signals between different states or outputs, reducing the error between these different signals when one or more subsystems are disturbed internally or externally [188]. An example of synchronizing speed control is where the output difference between two actuated system speed outputs is taken as a compensation signal. A controller C_{sc} acts on this compensation signal, and its output is sent to the corresponding speed reference signal. In this way, the speed error between the two speed outputs is reduced when disturbances occur.



Figure 5.7: Synchronizing control architecture to improve the synchronization between two system outputs y_1 and y_2 , with r being the setpoint for both system outputs, C_{sc} being the synchronizing controller, e_1 and e_2 being the inputs for the decentralized controllers C_1 and C_2 with controller outputs u_1 and u_2 , d_1 and d_2 being disturbances and u being the system input

5.1.3 Implementation in Co-Design Methodology

For this PhD's general co-design optimization methodology, the design space of the possible control architecture features applied to a system is decided by the end-user and can consist of combinations of one or more previously mentioned architecture features. In this control architecture design, reference signals Ref and input disturbances *Dist* can also be taken into account. In addition, each control architecture feature will have a certain cost that can be freely determined by the end-user. The cost estimation can be influenced by, for example, hardware cost, programming cost, or estimated training cost for operators or control technicians. Each separate controller C_{...} can be of various forms (e.g., P, PI, PID). For example, a control architecture with multiple cascade control levels can be applied on a motion control setup with PID decentralized controllers, PID distributed controllers, PI synchronizing controllers, and P feedforward controllers on both speed and position cascade loops. Each control architecture feature is accompanied by a binary $b_{1} \in [0,1]$ indicating its presence or absence. For clarity, the same subscripts are used for the control architecture binaries b_{μ} as for the corresponding control architecture features. Subscripts b_{dec} and b_{dis} can be used to group decentralized and distributed controllers from the same cascade levels. For example, the presence of all distributed controllers for a specific cascade level c can be determined simultaneously by one control architecture binary $b_{c.dis}$. The control architecture selection binaries are grouped in vector \mathbb{B}_{Υ} . The general co-design optimization methodology can switch between different control architectures by changing the binary values in \mathbb{B}_{Υ} . These binaries are then taken into account during the calculation of the closed-loop system response. As a result, every decentralized controller, distributed controller, feedforward controller, and synchronizing controller can be activated or deactivated by a corresponding binary in the general co-design optimization methodology. This is a unique feature of this co-design methodology that has not yet been described elsewhere.

5.1.4 Mechanical Synchronization Setup

Next, the control architecture design space is defined for the illustrative mechanical synchronization setup, introduced in Section 3.6.4 on page 64. For this motion application, the process state-space system with control architecture possibilities are depicted in Fig. 5.8. The open-loop process of this mechanical synchronization setup is seen in light blue (A_{pro} , B_{pro} , C_{pro} , and D_{pro}). As can be seen in (3.8) on page 65, the state matrix for this example contains both the position and the speed of the inertias, while only the positions are available as process outputs (see (3.9)). This means that the speed cannot be fed back directly from the process outputs. A natural way of obtaining the speed of the inertias is by deriving these from the inertia position information. However, by way of example, an observer is used to estimate the inertia speeds in this mechanical synchronization case.

The state observer is shown in dark blue (A_{obs} , B_{obs} , C_{obs} , D_{obs} , and L_{obs}).

A rule of thumb in observer design is to choose the observer poles so that their dynamics are ten times faster than the process's dynamics [163]. This is done to ensure that the estimated states will approximate the process states sufficiently quickly. More information on the observer design can be found in Subsection 3.6.2 on page 60.

For this control architecture example, there are two cascade levels denoted with subscripts a and b for the inner speed control and outer position control, respectively. The cascaded decentralized PID controllers and distributed PID controllers can be seen in red and orange, respectively. Next to cascade control, feedforward control can be applied to reduce the tracking error between the system outputs and the references (marked in light green in Fig. 5.8). In this case, this feedforward control is applied on the outer position control cascade level b by forwarding the derivative of the reference signals, multiplied with gains $K_{ff,b,...}$. The feedforward control is shown in Fig. 5.8 in purple. This motion control application is a setup where synchronization is essential. That is why synchronizing PID controllers on both speed and position cascade levels are applied to improve output synchronization between the coupled inertias.

The presence of the different control architecture features is indicated with binaries, grouped in vector \mathbb{B}_{Υ} . The general co-design optimization algorithm considers the design parameters in \mathbb{B}_{Υ} to optimize the control architecture as part of the control configuration optimization. Table 5.1 shows the possible control architecture features with their corresponding binaries and respective implementation costs. These costs are arbitrary and can be changed by the end-user according to their estimated value.

	Corresponding	Cost
Control architecture feature	binary	[%]
Cascade level a decentralized speed control	$b_{a,dec}$	2.67
Cascade level a distributed speed control	$b_{a,dis}$	3.73
Cascade level b decentralized position control	$b_{b,dec}$	1.33
Cascade level b distributed position control	$b_{b,dis}$	2.13
Cascade level b feedforward control	$b_{b,ff}$	4.27
Cascade level a synchronizing speed control	$b_{a,sc}$	8.27
Cascade level b synchronizing position control	$b_{b,sc}$	8.27

 Table 5.1: Control architecture features with their corresponding binaries and implementation cost



Figure 5.8: Overview of the control scheme of the mechanical synchronization process $(\mathbf{A}_{pro}, \mathbf{B}_{pro}, \mathbf{C}_{pro}, \text{and } \mathbf{D}_{pro})$ with a state observer $(\mathbf{A}_{obs}, \mathbf{B}_{obs}, \mathbf{C}_{obs}, \mathbf{D}_{obs}, \text{and } \mathbf{L}_{obs})$ to obtain the measured states \hat{x} , controlled by decentralized and distributed PID cascade controllers on cascade levels *a* and *b* for the speed and position control, respectively. Also with feedforward P control and synchronizing PID control on both cascade levels *a* and *b* to follow the references Ref_1 and Ref_2 , while input disturbance Dist is present. $e_{...}$ are the controller input signals.

5.2 Controller Tuning Optimization

In previous parts of this chapter, it is mentioned that the control architecture can be changed and thus optimized by the general co-design optimization methodology. It is important to note that the co-design methodology discussed in this PhD not only optimizes the control architecture but also simultaneously optimizes the control parameters (= controller tuning) as part of the control configuration optimization.

These controller tuning parameters are grouped in vector \mathbb{R}_{tuning} and are real values. The amount of controller tuning parameters in \mathbb{R}_{tuning} depends on the design space of the control architecture. For example, a simple state feedback matrix for a system with n states and l inputs will have a feedback matrix \mathbf{K} with dimensions $l \times n$. In that case, there will also be $l \cdot n$ variables that need to be optimized by the co-design methodology as part of the controller tuning optimization. In the case of proportional-integral-derivative (PID) control, each P, PI, and PID controller will have one, two, and three controller tuning variables, respectively. The P, I, and D values will be represented as proportional gain K_p , integral time T_i , and derivative time T_d respectively.

Fig. 5.9 shows a general example of a control loop for a system that is controlled by a state feedback matrix \mathbf{K}_0 . Furthermore, integral action is applied to remove steady-state errors between the output y and reference r. A feedback matrix \mathbf{K}_i determines the gain of this integral action. In the simplest case, the state-space feedback matrices \mathbf{K}_0 and \mathbf{K}_i values can be chosen directly as design parameters in the optimization algorithm. This way, the general optimization algorithm chooses the best values for the feedback matrices \mathbf{K}_0 and \mathbf{K}_i respecting the applied objectives and constraints.



Figure 5.9: A general example of a control loop for a system that is controlled by a state feedback matrix \mathbf{K}_0 with integral action matrix \mathbf{K}_i to remove steady-state errors between the system output y and reference r to be followed. e is the error signal, x_i is the integrated error signal, x represent the system states, and u is the system input.

Other ways can be used to obtain the controller feedback values. For example, a linear-quadratic regulator (LQR) method can be applied (introduced in Chapter 2). An LQR method inherently involves a form of optimization in which the enduser assigns weights to matrices \mathbf{Q} and \mathbf{R} to adjust the impact of the states and inputs, respectively. The performance of an LQR controller comes down to the determination of the diagonal values of the weighting matrices \mathbf{Q} and \mathbf{R} , and this proves to be a challenging task. One of the reasons is that these weights are completely relative to each other and therefore are not directly related to time-domain criteria such as maximum actuator effort or output range [36].

By applying the LQR method in the scope of this PhD, the diagonal values of matrices \mathbf{Q} and \mathbf{R} are chosen as design parameters and optimized as a function of the objectives and constraints. As a result, time-domain criteria such as maximum actuator effort can easily be taken into account by programming these in the objective or constraint functions. This was done in the scope of this PhD, where the hardware architecture and control parameters are simultaneously optimized for a steel ladle application using the LQR method. More details can be found in [189]. This shows that the algorithm is able to adjust the \mathbf{Q} and \mathbf{R} weights to obtain a controller that makes optimal use of the available actuator effort. In a similar way, the tuning of robust controllers (e.g., H_2 and H_{∞}) can be optimized by designating the weights in the frequency domain as design parameters in the general co-design methodology. See Section 6.4 on page 153 for an example of the presented hardware and control co-design methodology applied to an LPV composite plate model with robust H_{∞} output feedback controller design.

5.2.1 Mechanical Synchronization Setup

The complete control architecture for the mechanical synchronization case is depicted in Fig. 5.8. It shows that P controllers are used for feedforward control. The tuning of these controllers is determined by a single gain $K_{p,ff,...}$ that needs to be optimized. In addition, PID controllers are used for the decentralized and distributed cascade control and the synchronization control. Each PID controller $PID_{...}$ is of the standard form:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_d \frac{de(t)}{dt} \right),$$
(5.1)

with e(t) and u(t) being the PID controller input and output [190]. Therefore, every PID controller has three parameters to be optimized: K_p for the proportional part, T_i for the integral part, and T_d for the derivative part. Table 5.2 lists all controllers for the mechanical synchronization setup with their corresponding tuning parameters. This shows that 32 controller tuning parameters (grouped in vector \mathbb{R}_{tuning}) can be optimized as part of the control tuning optimization. The values of these controller tuning parameters can be overridden by the corresponding control architecture binaries \mathbb{B}_{Υ} during the operation of the optimization algorithm. For example, if a feedforward control binary b_{ff} equals zero, the corresponding feedforward gains K_{ff} will also be set to zero so that they do not affect the closed-loop system response. The control configuration and hardware architecture co-design is further detailed in Chapter 6.

Table 5.2: List of all applicable controllers in the mechanical synchronization example's control architecture with their corresponding tuning parameters. This shows that there are 32 controller tuning parameters, grouped in vector \mathbb{R}_{tuning} .

		Controller
Controller description	Controller	tuning
	annotation	parameters
Decentralized cascade level <i>a</i> speed		$K_{p,a,1}$
control on feedback loop 1	$PID_{a,1}$	$T_{i,a,1}$
F		$T_{d,a,1}$
Decentralized cascade level a speed	DID	$K_{p,a,2}$
control on feedback loop 2	$PID_{a,2}$	$T_{i,a,2}$
L		$T_{d,a,2}$
Distributed cascade level a speed		$K_{p,a,1-2}$
control from feedback loop $\hat{1}$ to 2	$PID_{a,1-2}$	$T_{i,a,1-2}$
		$I_{d,a,1-2}$
Distributed cascade level a speed	חות	$\Gamma_{p,a,2-1}$
control from feedback loop 2 to 1	$\Gamma I D_{a,2-1}$	$T_{i,a,2-1}$
		$I_{d,a,2-1}$
Decentralized cascade level b position		$T_{p,a,1}$
control on feedback loop 1	$1 1 D_{b,1}$	$T_{i,b,1}$
		$K_{d,b,1}$
Decentralized cascade level b position	PID	$T_{1,b,2}$
control on feedback loop 2	1126,2	$T_{i,b,2}$
		K_{nb1-2}
Distributed cascade level <i>b</i> position	PID_{h1-2}	T_{ib1-2}
control from feedback loop 1 to 2	0,1 2	$T_{d,b,1-2}$
		$K_{p,b,2-1}$
Distributed cascade level <i>b</i> position	$PID_{b,2-1}$	$T_{i,b,2-1}$
control from feedback loop 2 to 1		$T_{d,b,2-1}$
Synchronizing control on cascade		$K_{p,a,sc}$
level a speed control	$PID_{a,sc}$	$T_{i,a,sc}$
		$T_{d,a,sc}$
Synchronizing control on cascade		$K_{p,b,sc}$
level <i>b</i> position control	$PID_{b,sc}$	$T_{i,b,sc}$
		$T_{d,b,sc}$
Feedforward control on cascade	$K_{ff,b,1}$	$K_{ff,b,1}$
level <i>b</i> position control	$ K_{ff,b,2}$	$K_{ff,b,2}$
		$\ \mathbb{R}_{tuning}\ $

5.3 Closed-Loop State-Space Methodology

Chapter 3 mentioned that derivative-free optimization algorithms are used to implement the general co-design methodology in this PhD. These algorithms require a large number of closed-loop response calculations to arrive at a solution. The closed-loop system consists of the open-loop process with extended control architecture features. These possible control architecture features are mentioned in Section 5.1.2, where each controller can be a P, PI, or PID controller on its own. The general co-design methodology has the important contribution that a comprehensive control structure with a wide range of control features can be applied and efficiently optimized. This is possible by developing a closed-loop state-space methodology to create a discrete closed-loop state-space system from an open-loop state-space process with an observer and an extensive feedback control structure, which is presented in this section.

The section starts with an elaborate motivation for applying the closed-loop state-space modeling methodology, outlining its concrete benefits. After that, back-ground information is provided on how to formulate the different parts of the extensive feedback control loop in the discrete time-domain, followed by a definition of the nomenclature for the different components. Next, the different steps of the generic closed-loop state-space methodology are explained in detail, after which an application on the mechanical synchronization case is illustrated. Finally, the results are compared with existing methods to demonstrate the validness and effectiveness of the presented closed-loop state-space methodology.

5.3.1 Background and Motivation

Nowadays, in complex system design, the dynamic behavior is first modeled and defined in a mathematical representation. This mathematical representation can be accomplished using different model types, e.g., state-space models, transfer function models, zero-pole gain models, or frequency response data models. Together with the open-loop model, feedback control loops can be included to simulate the closed-loop behavior of the system before making expensive physical proto-types. Conventionally, feedback control loops with complex control architectures are modeled using graphical programming tools, such as MATLAB[®] Simulink or National Instruments[®] LabVIEW. These causal block diagrams (CBD) offer the advantages that the input and output relationships between the different inter-connected control loop blocks can be flexibly adjusted and graphically arranged. Another advantage is that libraries of pre-programmed blocks can be implemented [13, 14, 191].

However, the main disadvantage of these graphical programming tools is that they take a relatively long time to calculate a closed-loop response. This is not a big issue for a single simulation, but it does pose a problem if it is to be used in iterative optimization algorithms. Frequently used evolutionary-based optimization algorithms use iterative calculations of system responses to arrive at an optimal solution. When applying these iterative optimization algorithms, typically, a large number of simulations is required. Therefore, the previously mentioned graphical programming tools cannot be used to quickly and efficiently calculate the closed-loop responses when using iterative optimization algorithms.

On the contrary, calculating the response of discrete open-loop state-space system models can be done very quickly, thanks to fast matrix calculations. In current literature, the description of control loops in the state-space domain is limited to an output or state feedback controller [36, 159], with or without an integral reference tracking [163], or an observer to estimate non-measurable states [52, 160]. No examples were found of methods to implement a comprehensive control loop in the state-space domain such as the ones shown in Fig. 5.8 on page 96. That is why in this PhD dissertation, a generally applicable methodology is introduced to generate a closed-loop state-space formulation of a process with an observer and with a large variety of control loop possibilities. The presented methodology enables the inclusion of an observer with combinations of cascaded decentralized and distributed PID control, feedforward control and synchronizing PID control while taking into account reference tracking and input disturbances. Moreover, the influence of non-linearities, such as actuator output saturation, can be considered using this methodology. The impact of adding or omitting specific controller structures can also be investigated quickly by setting the corresponding controller gains to zero based on the active control architecture binary values. Together with the flexible adjustment of the controller architecture, process parameters can also easily be adjusted, and their influence on the closed-loop system response can efficiently be determined. Therefore, the presented closed-loop state-space methodology enables to efficiently execute the simultaneous co-design of both the hardware architecture and the control configuration, which is the primary goal of this PhD.

Another advantage of this generic methodology is that it enables to apply a wide range of different state-space analysis techniques on the closed-loop model with extended control structures [15, 163, 192, 193]. For example, the complete closed-loop system stability can be examined in function of the applied control structures with corresponding controller tuning. This makes it possible to predetermine the unstable behavior of specific closed-loop configurations without the need to fully calculate the response.

This closed-loop state-space methodology only considers linear time-invariant (LTI) systems in this PhD work, but the proposed methodology also allows for the integration of non-linear state-space models like linear parameter-varying (LPV) [194] or linear time-varying (LTV) [172] systems.

5.3.2 Discrete State-Space Process and Observer Representation

In Section 3.6 on page 59, background information is given on the continuous time-domain state-space modeling. These continuous systems cannot be directly simulated using a computer. For this purpose, discrete system descriptions are needed instead. A discrete (open-loop) state-space representation $SS_{pro.d}$ is used to represent a linear time-invariant (LTI) process in matrix form. A continuous-time LTI model can be converted to a discrete-time model using various discretization methods, for example, 'zero-order hold' [195], 'first-order hold' [196], 'impulse variant' [197], or 'Tustin' [198]. The discretized state-space structure of the continuous LTI state-space system formulation in (3.2) is shown in (5.2), with *n* being the number of states in the state vector \mathbf{x} , m being the number of outputs in the output vector \mathbf{v} and l being the maximum number of available actuator inputs from input vector u [52]. The resulting discrete state-space process matrix values depend on the desired sampling time T_s . An alternative way to achieve a discrete system representation is to obtain it through the use of 'difference equations' (in analogy with 'differential equations' for continuous representations). More details on discrete LTI systems can be found in [52, 160, 199].

$$\mathbf{SS}_{pro,d} = \begin{cases} \mathbf{x}(k+1) = \mathbf{A}_{pro,d} \cdot \mathbf{x}(k) + \mathbf{B}_{pro,d} \cdot \mathbf{u}(k) \\ \mathbf{y}(k) = \mathbf{C}_{pro,d} \cdot \mathbf{x}(k) + \mathbf{D}_{pro,d} \cdot \mathbf{u}(k) \end{cases},$$

with $\mathbf{x} \in \mathbb{R}^{n}$, $\mathbf{u} \in \mathbb{R}^{l}$, $\mathbf{y} \in \mathbb{R}^{m}$ (5.2)

Assume that the discrete LTI process matrices $\mathbf{A}_{pro,d}$, $\mathbf{B}_{pro,d}$, $\mathbf{C}_{pro,d}$, $\mathbf{D}_{pro,d}$ and input vector \mathbf{u} are known for every sample from starting time T_0 to total simulation time T_m with sampling time intervals T_s . In this case, the response of the system, being the coarse of the system state vector \mathbf{x} and the output vector \mathbf{y} based on the input vector \mathbf{u} , can be calculated iteratively according to the pseudocode in Algorithm 1.

Algorithm 1 Calculation of a discrete process state $\mathbf{x}(k)$ and output response $\mathbf{y}(k)$ with known inputs $\mathbf{u}(k)$.

1: for k = 1: T_m/T_s 2: $\mathbf{x}(k+1) = \mathbf{A}_{pro,d} * \mathbf{x}(k) + \mathbf{B}_{pro,d} * \mathbf{u}(k)$ 3: $\mathbf{y}(k) = \mathbf{C}_{pro,d} * \mathbf{x}(k) + \mathbf{D}_{pro,d} * \mathbf{u}(k)$ 4: end

Similarly, the continuous time-domain observer description from (3.3) can be discretized, and this discrete observer response can be calculated using the pseudocode shown in Algorithm 2.

Algorithm 2 Calculation of a discrete observer estimated states $\hat{\mathbf{x}}(k)$ and estimated output response $\hat{\mathbf{y}}(k)$ with known inputs $\mathbf{u}(k)$ and process output measurements $\mathbf{y}(k)$.

1: for k = 1: T_m/T_s 2: $\hat{\mathbf{x}}(k+1) = \mathbf{A}_{obs,d} * \hat{\mathbf{x}}(k) + \mathbf{B}_{obs,d} * \mathbf{u}(k) + \mathbf{L}_{obs,d} * (\mathbf{y}(k) - \hat{\mathbf{y}}(k))$ 3: $\hat{\mathbf{y}}(k) = \mathbf{C}_{obs,d} * \hat{\mathbf{x}}(k) + \mathbf{D}_{obs,d} * \mathbf{u}(k)$ 4: end

5.3.3 Discrete State-Space PID Representation

The algorithms mentioned above only consider the open-loop response of the process. The methodology described in this chapter aims to extend the algorithms mentioned above to calculate the closed-loop response of the process controlled by comprehensive loop structures. Therefore, the next step is to describe a continuous Proportional-Integral-Derivative (PID) controller in a discrete state-space representation. The relation from controller input e(t) to controller output u(t) for a standard form PID controller in the continuous time-domain can be seen in (5.3), of which Fig. 5.10 (a) gives a graphical overview [15].

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right)$$

= $K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt}$ (5.3)

The integral and derivative parts are discretized using numerical definitions to forms suitable for computer computation. This can be seen in (5.4) and (5.5) for the derivative and integral parts, respectively.

$$\frac{de(t)}{dt} \approx \frac{e(t) - e(t-1)}{T_s}$$
(5.4)

$$\int_0^t e(t)dt \approx T_s \sum_0^k e(k) \tag{5.5}$$

The discretized PID algorithm is shown below in (5.6).

$$u(k) = K_p e(k) + \frac{K_p T_s}{T_i} \sum_{0}^{k} e(k) + \frac{K_p T_d \left(e(k) - e(k-1)\right)}{T_s}$$
(5.6)

(5.7) is obtained by time-shifting (5.6) back one sample.

$$u(k-1) = K_p e(k-1) + \frac{K_p T_s}{T_i} \sum_{0}^{k-1} e(k) + \frac{K_p T_d \left(e(k-1) - e(k-2)\right)}{T_s}$$
(5.7)

By subtracting (5.7) from (5.6) and rearranging, (5.8) is obtained.

$$u(k) = u(k-1) + K_p (e(k) - e(k-1)) + \dots$$

$$\frac{K_p T_s}{T_i} e(k) + \frac{K_p T_d}{T_s} (e(k) - 2e(k-1) + e(k-2))$$

$$= u(k-1) + K_p \left(1 + \frac{T_s}{T_i} + \frac{T_d}{T_s}\right) e(k) + \dots$$

$$- K_p \left(1 + 2\frac{T_d}{T_s}\right) e(k-1) + \frac{K_p T_d}{T_s} e(k-2)$$
(5.8)

This equation is also known as the discrete PID velocity form [200]. After rearranging, (5.9) is obtained.

$$u(k) = u(k-1) + c_0 e(k) + c_1 e(k-1) + c_2 e(k-2),$$

with: $c_0 = K_p \left(1 + \frac{T_s}{T_i} + \frac{T_d}{T_s} \right),$
 $c_1 = -K_p \left(1 + 2\frac{T_d}{T_s} \right),$ and
 $c_2 = \frac{K_p T_d}{T_s}$ (5.9)

This discrete PID controller is graphically shown in Fig. 5.10 (b), from which a discrete state-space representation can be derived with states x_1 and x_2 . The pseudocode to iteratively calculate the discrete PID controller output u(k) from an input e(k) can be seen in Algorithm 3.

Algorithm 3 Calculation of a discrete PID controller output response u(k) to a known input e(k) with separate calculations for the states $x_1(k)$ and $x_2(k)$.

1:
$$c_0 = K_p(1 + T_s/T_i + T_d/T_s)$$

2: $c_1 = -K_p(1 + 2T_d/T_s)$
3: $c_2 = K_pT_d/T_s$
4: for $k = 1$: T_m/T_s
5: $u(k) = x_1(k) + c_0 * e(k)$
6: $x_1(k+1) = x_2(k) + c_1 * e(k) + u(k)$
7: $x_2(k+1) = c_2 * e(k)$
8: end

Using vertical matrix concatenation, the two equations to calculate the states (lines 6 & 7 in Algorithm 3) can be combined into one equation that simultaneously calculates both states, as shown in Algorithm 4.





(b) Schematic overview of a PID controller in discrete time domain. See (5.9) for clarification on values c_0 , c_1 and c_2 .

Algorithm 4 Calculation of a discrete PID controller output response u(k) to a known input e(k) with combined calculations for the states $\mathbf{x}(k)$.

1: $c_0 = K_p(1 + T_s/T_i + T_d/T_s)$ 2: $c_1 = -K_p(1 + 2T_d/T_s)$ 3: $c_2 = K_pT_d/T_s$ 4: for $k = 1 : T_m/T_s$ 5: $u(k) = \mathbf{x}(k, 1) + c_0 * e(k)$ 6: $\mathbf{x}(k+1, :) = \begin{bmatrix} 0 & 1; 0 & 0 \end{bmatrix} * \mathbf{x}(k, :) + \begin{bmatrix} c_1; c_2 \end{bmatrix} * e(k) + \begin{bmatrix} 1; 0 \end{bmatrix} * u(k)$ 7: end

5.3.4 Extensive Control Structure Possibilities

Modern industrial drives used for motion control have very extensive control architecture possibilities, as can be seen in Section 5.1. This closed-loop state-space methodology aims to implement these industrially relevant control loop possibilities together with the existing process model into one closed-loop state-space representation. Fig. 5.8 on page 96 depicts an example of a system with an extensive control scheme. This figure is valid for both the continuous and the discrete timedomain, which is why no distinction is made between the two time-domains for the moment. The closed-loop state-space system is referred to as SS_{CL} , which is composed of the open-loop state-space process SS_{pro} (containing matrices A_{pro} , \mathbf{B}_{pro} , \mathbf{C}_{pro} , and \mathbf{D}_{pro}) with an observer (containing matrices \mathbf{A}_{obs} , \mathbf{B}_{obs} , \mathbf{C}_{obs} , \mathbf{D}_{obs} , and \mathbf{L}_{pro}) and its control structures. The process is driven by two actuator inputs 1 and 2 with decentralized and distributed cascaded PID controllers on the inner cascade level a $(PID_{a,1}, PID_{a,1-2}, PID_{a,2-1}, \text{ and } PID_{a,2})$ and outer cascade level b ($PID_{b,1}$, $PID_{b,1-2}$, $PID_{b,2-1}$, and $PID_{b,2}$), feedforward control on the outer cascade level b ($K_{ff,b,1}$ and $K_{ff,b,2}$), synchronizing PID control on both inner cascade level a $(PID_{a,sc})$ and outer cascade level b $(PID_{b,sc})$, reference tracking $(Ref_1 \text{ and } Ref_2)$, and input disturbance (Dist). Table 5.3 shows the nomenclature for all relevant matrices and signals. Although the methodology mainly refers to the example in Fig. 5.8, it is certainly not limited to this topology and can be extended with, for example, more subsystem inputs (3, 4, etc.) or more nested cascade loop levels (c, d, etc.). Typically, the number of cascade levels (a, d, etc.) b, c, etc.) will depend on the order of the system.

Matrices			
A	State-space system matrix		
B	State-space input matrix		
C	State-space output matrix		
D	State-space feedforward matrix		
Signals			
\mathbf{x}_{pro}	LTI Process state vector		
$\hat{\mathbf{x}}_{pro}$	Observer state vector		
y	Output vector		
$Ref_{}$	Reference trajectories		
Dist	Input disturbances		
e	Controller inputs		
<i>u</i>	LTI process inputs		
Subscripts			
$\cdots pro$	Referring to the LTI Process		
··· obs	Referring to the observer		
$\ldots_a, \ldots_b,$ etc.	Referring to the cascade loop level		
₁ , ₂ , etc.	Referring to the actuator input		
$\cdots ff$	Referring to feedforward control		
sc	Referring to synchronizing control		
Controllers			
PID	PID controller		
<i>Kp</i>	PID controller proportional gain		
Ti	PID controller integration time		
<i>Td</i>	PID controller derivative time		
Nomenclature	examples		
\mathbf{B}_{pro}	LTI process state-space input matrix		
\mathbf{C}_{obs}	Observer state-space output matrix		
$PID_{a,2}$	Cascade level a decentralized PID controller on actuator input 2		
מות	Cascade level b distributed PID controller from actuator input 1 to		
$PID_{b,1-2}$	actuator input 2		
$e_{b,sc}$	Input signal for the synchronizing PID controller on cascade level b		
$K_{ff,a,1}$	Cascade level a feedforward controller gain on actuator input 1		
V	Cascade level a decentralized controller on actuator input 2		
$\kappa_{p,a,2}$	proportional gain		
T	Cascade level <i>b</i> distributed controller from actuator 2 to actuator 1		
<i>⊥i,b,2−1</i>	integration time		
	Cascade level a decentralized controller on actuator input 1		
<i>⊥ d</i> , <i>a</i> ,1	derivative time		

 Table 5.3: Nomenclature for the relevant matrices, signals and gains, referring to Fig. 5.8 on page 96

5.3.5 SS_{CL} Methodology Workflow

This section establishes a reasoning for the various steps an end-user must take to obtain the desired closed-loop state-space formulation. Figure 5.11 shows an overview of the methodology. Next, the different steps are explained in more detail.



Figure 5.11: Overview of the different steps behind the logic of the closedloop state-space methodology. The numbering of the different tasks corresponds to the numbering in the text.

1. Label each controller input $e_{...}$ with a logical name. An overview of the subscripts for the controller input names is given in Table 5.3. Next, identify the external inputs. These are the inputs from reference trajectories and disturbances (light green and dark green in Fig. 5.8, respectively).

2. Create a set of equations representing the relationships between the process, the observer, and the controllers, using the appropriate inputs and outputs from the previous step. As stated earlier, the LTI process SS_{pro} has *n* states, *m* outputs and *l* inputs and there are *o* PID controllers in the overall control structure, with each PID controller having two states. The separate discrete state-space calculations for an LTI process, an observer, and a PID controller are shown in algorithms 1, 2, and 4, respectively. All separate LTI process, observer, and controller state equations must be set up correctly using unique (and preferably) logical naming. The parameters c_0 , c_1 and c_2 for every PID controller are determined by (5.9).

The causality of the control loop determines the order of the equations and interactions. This causality is represented as a Causal Block Diagram (CBD) in Fig. 5.8 [13]. In this case, the order of the equations and interactions starts from the external reference trajectory via the cascaded loop levels to the LTI process and observer (in other words, from left to right, referring to Fig. 5.8). The resulting pseudocode can be seen in Algorithm 5. The calculation of the system response is in the discrete time-domain, but the subscripts '*d*' are not used in the pseudocodes for the sake of simplicity.

Algorithm 5 Extensive iterative calculation of a process with an observer, decentralized and distributed cascaded loop control, synchronizing control, reference trajectories and disturbance inputs. An overview of the entire system can be seen in Fig. 5.8 on page 96.

1: f	or $k = 1: T_m/T_s$
	Cascade levels a & b synchronizing controller inputs
2:	$e_{b,sc}(k) = y(1,k) - y(2,k)$
3:	$e_{a,sc}(k) = \hat{x}(1,k) - \hat{x}(2,k)$
	Cascade levels a & b synchronizing controller equations
4:	for $\mathbf{i} = [\mathbf{a}, \mathbf{b}]$
5:	$PID_{i,sc,u}(k) = PID_{i,sc,x}(1,k) + c_{i,sc,0} * e_{i,sc}(k)$
6:	$PID_{i,sc,x}(:,k+1) = [0\ 1;0\ 0] * PID_{i,sc,x}(k) + \dots$
	$[c_{i,sc,1}; c_{i,sc,2}] * e_{i,sc}(k) + [1;0] * PID_{i,sc,u}(k)$
7:	end
	Cascade level b controller inputs
8:	$e_{b,1}(k) = Ref_1(k) - y(1,k) - PID_{b,sc,u}(k)$
9:	$e_{h,2}(k) = Ref_2(k) - y(2,k) + PID_{h,sc,u}(k)$

Algorithm 6 (Continued from Algorithm 5.)

Cascade level *b* decentralized and distributed controller equations 10: for j = [1, 2, 1-2, 2-1]11: $PID_{b,i,u}(k) = PID_{b,i,x}(1,k) + c_{b,i,0} * e_{b,i}(k)$ $PID_{h,i,x}(:, k+1) = [0\ 1; 0\ 0] * PID_{h,i,x}(k) + \dots$ 12: $[c_{b,i,1}; c_{b,i,2}] * e_{b,i}(k) + [1;0] * PID_{b,i,u}(k)$ 13: end Cascade level *a* controller inputs $e_{a,1}(k) = PID_{b,1,u}(k) + PID_{b,2-1,u}(k) + \dots$ 14: $K_{ff,b,1} * (Ref_1(k) - Ref_1(k-1)) / T_s - \hat{x}(1,k) - PID_{a,sc,u}(k)$ $e_{a,2}(k) = PID_{b,2,u}(k) + PID_{b,1-2,u}(k) + \dots$ 15: $K_{ff,b,2} * (Ref_2(k) - Ref_2(k-1)) / T_s - \hat{x}(2,k) + PID_{a,sc,u}(k)$ Cascade level *a* decentralized and distributed controller equations for j = [1, 2, 1-2, 2-1]16: $PID_{a,j,u}(k) = PID_{a,j,x}(1,k) + c_{a,j,0} * e_{a,j}(k)$ 17: $PID_{a,i,x}(:, k+1) = [0\ 1; 0\ 0] * PID_{a,i,x}(k) + \dots$ 18: $[c_{a,j,1}; c_{a,j,2}] * e_{a,j}(k) + [1;0] * PID_{a,j,u}(k)$ end 19: Process inputs $u(1,k) = PID_{a,1,u}(k) + PID_{a,2-1,u}(k)$ 20: $u(2,k) = PID_{a,2,u}(k) + PID_{a,1-2,u}(k)$ 21: 22: u(3,k) = Dist(k)Process and observer equations $x(:, k+1) = A_{pro} * x(:, k) + B_{pro} * u(:, k)$ 23: $y(:, k+1) = C_{pro} * x(:, k) + D_{pro} * u(:, k)$ 24: $\hat{x}(:, k+1) = A_{obs} * \hat{x}(:, k) + B_{obs} * u(:, k) + L_{obs} * (y(:, k) - \hat{y}(:, k))$ 25: 26: $\hat{y}(:, k+1) = C_{obs} * \hat{x}(:, k) + D_{obs} * u(:, k)$ 27: end

Algorithm 5 allows to quickly and efficiently calculate the closed-loop response of the system with an extensive control structure. It is also possible to include actuator output saturation. To do so, code lines 20 and 21 of Algorithm 5 above are extended as shown in Algorithm 7, in which $u_{max,Act,...}$ is the maximum actuator output of the corresponding actuator. Depending on the chosen actuator selection integers \mathbb{Z}_{act} , other values for $u_{max,Act,...}$ may apply. In this way, the general co-design optimization algorithm can take into account changing, non-linear actuator output saturation related to the selected actuator types. **Algorithm 7** Edit of the process input calculations to implement (non-linear) actuator output saturation, with $u_{max,act,...}$ being the maximum actuator output value

	Process inputs
1:	$u(1,k) = \min(u_{max,act,1}, max(-u_{max,act,1}, \dots$
	$PID_{a,1,u}(k) + PID_{a,2-1,u}(k)))$
2:	$u(2,k) = \min(u_{max,act,2}, max(-u_{max,act,2}, \dots$
	$PID_{a,2,u}(k) + PID_{a,1-2,u}(k)))$

The closed-loop system response calculation equations in Algorithm 5 are not in the general, condensed state-space form as in (5.2). The following steps describe how to obtain this appropriate representation of SS_{CL} .

3. Successive calculations for separate PID controllers having the same inputs can be combined using matrix concatenations. An example of equations for separate controllers PID_p and PID_q sharing the same input *e* are shown in (5.10) and (5.11).

$$PID_{p,u}(:,k) = PID_{p,x}(1,k) + c_{p,0}e(:,k)$$

$$PID_{p,x}(:,k+1) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} PID_{p,x}(:,k) + \dots$$

$$\begin{bmatrix} c_{p,1} \\ c_{p,2} \end{bmatrix} e(:,k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} PID_{p,u}(:,k)$$

$$PID_{q,u}(:,k) = PID_{q,x}(1,k) + c_{p,0}e(:,k)$$

$$PID_{q,x}(:,k+1) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} PID_{q,x}(:,k) + \dots$$

$$\begin{bmatrix} c_{q,1} \\ c_{q,2} \end{bmatrix} e(:,k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} PID_{q,u}(:,k)$$
(5.11)

These equations can be combined to obtain a controller PID_r with equations:

$$PID_{r,u}(:,k) = PID_{r,x}(1,k) + PID_{r,x}(3,k) \begin{bmatrix} c_{p,0} \\ c_{q,0} \end{bmatrix} e(:,k)$$

$$PID_{r,x}(:,k+1) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} PID_{r,x}(:,k) + \dots$$

$$\begin{bmatrix} c_{p,1} \\ c_{p,2} \\ c_{q,1} \\ c_{q,2} \end{bmatrix} e(:,k) + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} PID_{r,u}(:,k)$$
(5.12)

This approach can be applied on Algorithm 5 to obtain algorithm 9 in Appendix A.

4. The equations can be further grouped by integrating intermediate solutions into the successive equations. More specifically for this methodology, the controller input equations can be entered directly into the controller state equations. An example follows of how the equations for the inputs e_1 and e_2 for a system PID_x in (5.13) can be converted to a combined equation for PID_x in (5.14).

$$e_{1}(k) = Ref_{1}(k) - y(1,k) - PID_{sc,u}(2,k)$$

$$e_{2}(k) = Ref_{2}(k) - y(2,k) - PID_{sc,u}(2,k)$$

$$PID_{x}(:,k+1) = S_{pre,2}PID_{x}(:,k) + \dots$$

$$\begin{bmatrix} c_{1,1} & 0 \\ c_{1,2} & 0 \\ 0 & c_{2,1} \\ 0 & c_{2,2} \\ c_{1-2,1} & 0 \\ c_{1-2,2} & 0 \\ 0 & c_{2-1,1} \\ 0 & c_{2-1,2} \end{bmatrix} \begin{bmatrix} e_{1}(k) \\ e_{2}(k) \end{bmatrix} + S_{pre,3}PID_{u}(:,k)$$
(5.13)

$$PID_{x}(:, k+1) = S_{pre,2}PID_{x}(k) + \begin{bmatrix} c_{1,1} & 0\\ c_{1,2} & 0\\ 0 & c_{2,1}\\ 0 & c_{2,2}\\ c_{1-2,1} & 0\\ c_{1-2,2} & 0\\ 0 & c_{2-1,1}\\ 0 & c_{2-1,2} \end{bmatrix} \dots$$

$$\begin{bmatrix} \left[Ref_{1}(k) \\ Ref_{2}(k) \right] + \left[-y(1,k) \\ -y(2,k) \right] + \dots \\ \begin{bmatrix} 0 & -1\\ 0 & -1 \end{bmatrix} PID_{sc,u}(:,k) \end{bmatrix} + S_{pre,3}PID_{u}(k)$$
(5.14)

This step will reduce the number of lines for the discrete calculation, but it will make the equations themselves more lengthy. Additionally, the process and observer equations are combined. This approach can be used on Algorithm 9 to obtain Algorithm 11, see Appendix A.

5. Similarly as in the previous step, calculating the output u of each separate part can be included in the state equations. During the iterative calculations, the outputs of the separate parts are calculated for sample k, while the states are calculated for the next sample k + 1. In other words, the calculation of the states at sample k + 1 is always based on signals at the previous time sample k.

Next, an example is given of how a system with separate equations for the controller output u and controller states x (5.15) can be converted to a system with only state equations (5.16) by integrating the output equations into the state equations using matrix algebra. Part of the matrices are denoted as [...] to save space and keep the equations organized. The values of these matrices follow from the previous steps.

$$\begin{split} PID_{sc,u}(:,k) = [...]PID_{sc,x}(:,k) + [...]x(:,k) + [...]y(:,k) \\ PID_{sc,x}(:,k+1) = [...]PID_{sc,u}(:,k) + [...]PID_{sc,x}(:,k) + ... \\ [...]x(:,k) + [...]y(:,k) \\ PID_{b,u}(k) = [...]PID_{sc,u}(:,k) + [...]PID_{b,x}(:,k) + ... \\ [...]y(:,k) + [...] \begin{bmatrix} Ref_1(k) \\ Ref_2(k) \end{bmatrix} \\ PID_{b,x}(:,k+1) = [...]PID_{sc,u}(:,k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{b,x}(k) + [...]y(:,k) + [...] \begin{bmatrix} Ref_1(k) \\ Ref_2(k) \end{bmatrix} \\ PID_{a,u}(k) = [...]PID_{sc,u}(:,k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{a,x}(:,k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{a,u}(k) + [...]PID_{a,x}(:,k) + ... \\ [...]PID_{a,u}(k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + ... \\ [...]PID_{a,u}(k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + [...]PID_{b,u}(k) + ... \\ [...]PID_{b,u}(k) + ... \\$$

$$\begin{split} PID_{sc,x}(:, k+1) &= [...]PID_{sc,x}(:, k) + [...]x(:, k) + [...]y(:, k) \\ PID_{b,x}(:, k+1) &= [...]PID_{sc,x}(:, k) + [...]PID_{b,x}(k) + ... \\ & [...]x(:, k) + [...]y(:, k) + [...] \begin{bmatrix} Ref_1(k) \\ Ref_2(k) \end{bmatrix} \\ PID_{a,x}(:, k+1) &= [...]PID_{sc,x}(:, k) + [...]PID_{b,x}(k) + ... \\ & [...]PID_{a,x}(k) + [...]x(:, k) + ... \\ & [...]y(:, k) + [...] \begin{bmatrix} Ref_1(k) \\ Ref_2(k) \end{bmatrix} \\ u(:, k) &= [...]PID_{sc,x} + [...]PID_{b,x} + [...]PID_{a,x} + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]PID_{b,x} + [...]PID_{a,x} + ... \\ & [...]x(:, k) + [...]PID_{b,x} + [...]PID_{a,x} + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]PID_{b,x} + [...]PID_{a,x} + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]y(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + [...]x(:, k) + ... \\ & [...]x(:, k) + ...$$

After this step, very extensive equations can be obtained, with large parts remaining constant throughout the iterative calculation of the closed-loop system response. As stated before, these parts are grouped in predetermined matrices and calculated in advance so that they do not need to be recalculated for each sample. These sparse matrices $S_{pre,...}$ are too large to be shown here clearly, but they follow from the previous steps in the described methodology. By doing so, the calculations of the total closed-loop system SS_{CL} are accelerated. This approach can be used on Algorithm 11 to obtain Algorithm 13, see Appendix A.

6. The last step is to combine the equations into one closed-loop state-space system. For this purpose, the individual states (from the process, observer, PID controllers, etc.) are grouped into one state matrix, and the system matrices \mathbf{A}_{CL} , \mathbf{B}_{CL} , \mathbf{C}_{CL} , and \mathbf{D}_{CL} are correspondingly grouped, ensuring that the original equations remain valid. The complete closed-loop system \mathbf{SS}_{CL} states consist of the LTI process states \mathbf{x}_{pro} , observer states $\hat{\mathbf{x}}_{pro}$, and

states of each PID controller $\mathbf{x}_{PID...}$ (with subscripts referring to the corresponding input and appropriate cascade level). The general form of the state matrix \mathbf{x}_{CL} for the closed-loop system \mathbf{SS}_{CL} is shown in (5.17).

$$\mathbf{x}_{CL} = \begin{bmatrix} \begin{bmatrix} \mathbf{x}_{OL} \\ n \times 1 \\ \begin{bmatrix} \hat{\mathbf{x}}_{OL} \\ n \times 1 \\ \\ \mathbf{x}_{PID...} \end{bmatrix} \\ 2 \times 1 \\ \begin{bmatrix} \mathbf{x}_{PID...} \\ 2 \times 1 \\ \\ 2 \times 1 \\ \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{CL} \\ (2 \cdot n + o \cdot 2) \times 1 \\ (2 \cdot n + o \cdot 2) \times 1 \end{bmatrix}$$
(5.17)

with n the number of states in the open-loop LTI process, and

o the total number of PID controllers in the control structure.

This also means that the size of matrix \mathbf{A}_{CL} will be $[(2 \cdot n + o \cdot 2) \times (2 \cdot n + o \cdot 2)]$. The closed-loop system \mathbf{SS}_{CL} input matrix \mathbf{u}_{CL} consists of the LTI process outputs \mathbf{y}_{OL} (from the previous sample), complemented by external disturbances $Dist_{...}$ and reference trajectories $Ref_{...}$.

A general representation of this input matrix \mathbf{u}_{CL} can be seen in (5.18).

$$\mathbf{u}_{CL} = \begin{bmatrix} \begin{bmatrix} \mathbf{y}_{OL} \\ n \times 1 \\ \begin{bmatrix} \mathbf{Dist}_{\dots} \\ q \times 1 \\ \begin{bmatrix} \mathbf{Ref}_{\dots} \end{bmatrix} \\ r \times 1 \end{bmatrix} = \begin{bmatrix} [\mathbf{u}_{CL}] \\ [(n+q+r) \times 1] \end{bmatrix}$$
(5.18)

with n the number of states in the LTI process,

q the number of external disturbance signals,

r the number of reference trajectories.

The outputs for this closed-loop system SS_{CL} are the same *m* outputs as for the LTI process. The closed-loop matrices C_{CL} and D_{CL} will have dimensions $m \times (2 \cdot n + o \cdot 2)$ and $m \times (n + q + r)$, respectively.

Below in Algorithm 8, the pseudocode is shown to iteratively calculate the closed-loop system SS_{CL} response based on external reference signals $Ref_{...}$ and disturbance signals $Dist_{...}$ after the closed-loop matrices A_{CL} , B_{CL} , C_{CL} , and D_{CL} have been determined using the previous steps.
Algorithm 8 General example of how the response of the resulting closed-loop state-space system SS_{CL} is iteratively calculated.

Non-iterative determination of the invariable closed-loop matrices (following from previous methodology steps)

1: $A_{CL} = \dots$

- 2: $B_{CL} = ...$
- 3: $C_{CL} = ...$
- 4: $D_{CL} = ...$

Iterative response calculation

- 5: for k = 1 : T_m/T_s
 Determine the inputs on sample k

 6: u(:,k) = [y(:,k); Dist_{...}(:,k); Ref_{...}(:,k)]
- Determine the states for the next sample k + 1
- 7: $x(:, k+1) = A_{CL} * x(:, k) + B_{CL} * u(:, k) + L * (y(:, k) \hat{y}(:, k))$ Determine the outputs for the next sample k + 1
- 8: $y(:, k+1) = C_{CL} * x(:, k) + D_{CL} * u(:, k)$

```
9: end
```

5.3.6 Mechanical Synchronization Setup

Applying the Methodology

Starting from the open-loop LTI process, the methodology described in this chapter can be applied to the mechanical synchronization case introduced in Section 3.6.4. This methodology is applied to obtain a closed-loop state-space system SS_{CL} containing the LTI process with state observer and extensive control structures, as can be seen in Fig. 5.8 on page 96. The states, inputs, and outputs for SS_{CL} of this mechanical synchronization case are shown in (5.19), (5.20), and (5.21), respectively.

$$\begin{bmatrix} \mathbf{x}_{CL} \\ 32 \times 1 \end{bmatrix} = \begin{bmatrix} [\mathbf{x}_{pro}] \\ 6 \times 1 \end{bmatrix}; \begin{bmatrix} \hat{\mathbf{x}}_{pro} \\ 6 \times 1 \end{bmatrix}; \begin{bmatrix} \mathbf{x}_{PID,a} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load} \\ \hat{\theta}_{1} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load} \end{bmatrix}; \begin{bmatrix} \hat{\theta}_{1} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load} \\ \hat{\theta}_{1} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load} \end{bmatrix}; \begin{bmatrix} \hat{\theta}_{1} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load} \\ \hat{\theta}_{1} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load} \end{bmatrix}; \begin{bmatrix} \mathbf{x}_{PID,a,1,1} \\ \mathbf{x}_{PID,a,2,2} \\ \mathbf{x}_{PID,a,2,2} \\ \mathbf{x}_{PID,a,1-2,1} \\ \mathbf{x}_{PID,a,2-1,1} \\ \mathbf{x}_{PID,b,2,2} \\ \mathbf{x}_{PID,b,2,2} \\ \mathbf{x}_{PID,b,1-2,1} \\ \mathbf{x}_{PID,b,2,2} \\ \mathbf{x}_{PID,b,1-2,1} \\ \mathbf{x}_{PID,b,2-1,2} \end{bmatrix}; \begin{bmatrix} \mathbf{x}_{PID,sc,a,1} \\ \mathbf{x}_{PID,sc,b,2} \\ \mathbf{x}_{PID,b,2-1,1} \\ \mathbf{x}_{PID,b,2-1,1} \\ \mathbf{x}_{PID,b,2-1,2} \end{bmatrix}; \begin{bmatrix} \mathbf{x}_{PID,sc,b,1} \\ \mathbf{x}_{PID,sc,b,2} \\ \mathbf{x}_{PID,b,2-1,2} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{u}_{CL} \\ \mathbf{0} \times \mathbf{1} \end{bmatrix} = \begin{bmatrix} [\mathbf{y}_{pro}] \\ 6 \times 1 \end{bmatrix}; \begin{bmatrix} Dist \\ 1 \times 1 \end{bmatrix}; \begin{bmatrix} Ref_{1} \\ \theta_{2} \\ \theta_{load} \\ \hat{\theta}_{1} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load}} \end{bmatrix}$$
(5.20)
$$\begin{bmatrix} \mathbf{y}_{CL} \\ \theta_{2} \\ \theta_{load} \\ \hat{\theta}_{1} \\ \hat{\theta}_{2} \\ \hat{\theta}_{load}} \end{bmatrix}$$
(5.21)

Results and Comparison with Existing Methods

As a result, the closed-loop state-space system matrices \mathbf{A}_{CL} , \mathbf{B}_{CL} , \mathbf{C}_{CL} , and \mathbf{D}_{CL} are obtained, containing the open-loop LTI process \mathbf{SS}_{pro} with an observer and extensive control structure in a closed-loop state-space formulation. The symbolic representations of these matrices are so extensive that they cannot be displayed here in an orderly manner. Note that the presented closed-loop state-space methodology is generally applicable, but other matrices will be obtained for other applications with different feedback structures. Algorithm 8 on page 117 shows the pseudocode on how to calculate the closed-loop system response for the resulting closed-loop matrices \mathbf{A}_{CL} , \mathbf{B}_{CL} , \mathbf{C}_{CL} , and \mathbf{D}_{CL} .

Next, the time savings of the proposed methodology are demonstrated by comparing the simulation of the model with the proposed methodology using standard MATLAB[®] workspace scripts versus the simulation of the same model with MATLAB[®] Simulink using the built-in 'ode3' (Bogacki-Shampine) solver [201]. Both simulation runs are performed on the same Intel[®] Xeon[®] CPU @ 3.10 GHz with 64 GB of RAM. In total, 1000 closed-loop response simulations are performed with a total simulation time $T_m = 30s$ and a sample time $T_s = 0.01s$ using the proposed SS_{CL} methodology, with different controller values for the closed-loop controllers for every simulation. The total calculation time is determined, being 36.6 s. Next, the Simulink model is simulated for the same 1000 different combinations of controller values as in the previous case and with the same simulation time T_m and sample time T_s . The time needed to perform these MATLAB[®] Simulink simulations is 3477.4 s. This shows that the calculation of 1000 simulations is approximately 95 times faster with the proposed methodology, as depicted in Fig. 5.12. This also indicates that this closed-loop state-space methodology will offer a significant advantage in calculating the general co-design methodology.



Figure 5.12: Calculation time comparison between using MATLAB[®] Simulink versus using the proposed SS_{CL} methodology with standard MATLAB[®] workspace scripts to perform the same 1000 model simulations

For one of the 1000 simulations in the previous comparison, the outputs of both the MATLAB[®] Simulink simulation and the proposed SS_{CL} methodology are compared. As shown in Fig. 5.13 and 5.14, practically the same results are obtained from both approaches, validating that the proposed SS_{CL} methodology provides reliable simulation results.

In addition to the time advantage, the proposed methodology also allows the activation or inactivation of specific control structures by adjusting the corresponding controller values. For example, the distributed cascaded PID controllers, synchronizing PID controllers, and feedforward controllers can be deactivated by setting the controller values to zero. The decentralized PID controllers can be deactivated by setting $1/Ti_{\dots}$ and Td_{\dots} to zero and setting Kp_{\dots} equal to 1.

Existing state-space analysis techniques can be applied on the closed-loop matrices \mathbf{A}_{CL} , \mathbf{B}_{CL} , \mathbf{C}_{CL} , and \mathbf{D}_{CL} (e.g., stability or sensitivity analysis [15, 52, 163]), which is not directly possible in existing graphical programming environments with techniques to simulate systems with extensive control structures.



Figure 5.13: The course of the reference trajectory and the displacement of the three inertias θ_1 , θ_2 and θ_{load} when simulating the MATLAB[®] Simulink model



Figure 5.14: The course of the reference trajectory and the displacement of the three inertias θ_1 , θ_2 and θ_{load} when simulating the model using the proposed SS_{CL} methodology

5.4 Chapter Conclusions

In this chapter, the control configuration optimization is detailed as part of the general hardware architecture and control configuration co-design. This control configuration optimization consists of a control architecture optimization and a controller tuning optimization.

First, the control architecture optimization is discussed, and an overview is given of three different control architecture topologies: centralized, decentralized, and distributed. A literature review on these different control architectures is carried out, showing that the optimal control architecture method is applicationdependent. With this information in mind, a control architecture optimization method is developed as part of the general hardware and control co-design methodology to determine the optimal control architecture automatically. This control architecture design space can be defined freely by the end-user and can consist of combinations of controller architecture features common in industrial applications. Examples of these architecture features are decentralized and distributed control, cascaded control, synchronizing control, and feedforward control. The presence or absence of the control architecture features is indicated by binaries, grouped in vector \mathbb{B}_{Υ} . The literature review also shows that simultaneous optimization of the control architecture as part of a hardware and control co-design is unprecedented. Not only the controller architecture but also the controller tuning parameters are optimized as part of the control configuration optimization. These controller tuning parameters are grouped in vector \mathbb{R}_{tuning} .

Subsequently, a general methodology is proposed to establish a closed-loop state-space system SS_{CL} from an open-loop LTI process with an extensive control structure. At the same time, the necessary reference trajectories and input disturbances can be taken into account. In this chapter, a generic methodology is described for a two-level cascaded control structure. However, the described generic methodology can be applied on more extensive (more cascaded levels or more inputs/outputs) or smaller systems. In that case, the methodology stays the same, but different resulting closed-loop matrices A_{CL} , B_{CL} , C_{CL} , and D_{CL} will be obtained. Extensions to non-linear systems (e.g., linear parameter-varying (LPV) or linear time-varying (LTV)) instead of linear time-invariant (LTI) systems can also be handled.

There are several advantages of using the proposed closed-loop state-space methodology compared to conventional methods that allow the implementation of comprehensive control structures. First and foremost, calculating the closed-loop system SS_{CL} response to external inputs and disturbances is much faster than conventional methods because the closed-loop response calculation is reduced to rapidly performed matrix calculations. For example, it is shown that the proposed methodology is up to 95 times faster than conventional methods for calculating different model simulations for a mechanical synchronizing setup with different controller settings. This advantage will have a significant impact when using iterative

algorithms (e.g., evolutionary algorithms), which require numerous simulations.

Moreover, the proposed SS_{CL} methodology enables to easily adjust the activation or deactivation of specific control structures in the same model framework by correspondingly adjusting specific controller parameters. In this way, the impact of the activation or elimination of specific control structures on the closed-loop functioning of the system can be calculated quickly and efficiently. That is why this closed-loop state-space methodology enables the efficient application of the general co-design methodology presented in this PhD.

Furthermore, existing state-space analysis techniques can be applied after implementing the proposed methodology, which is not directly possible with conventional simulation tools. For example, stability or sensitivity analyses can be carried out on the closed-loop matrices of the entire system.

Finally, the application of both the control architecture optimization and controller tuning optimization is shown on the practical example of a mechanical synchronization case that is used as an example throughout this dissertation.

Chapter 6 Hardware and Control Co-Design

The previous chapters specify how the hardware architecture and control configuration co-design problem can be formulated in a mathematically optimizable form. In this chapter, the application of an optimization algorithm to perform the hardware and control co-design is discussed.

First, a general description of the objective function, design parameters, and constraints concerning the proposed co-design optimization problem is given. An explanation is provided on the Genetic Algorithm (GA) implementation and the typical GA settings to perform the optimization. Next, the hardware architecture and control configuration co-design optimization is performed on a linear time-invariant (LTI) mechanical synchronization model, of which some parts have already been covered in the previous chapters. The co-design results of this case are compared with existing controller tuning methods, and the smoothness properties of the objective function are discussed.

Additionally, the flexibility of the presented co-design methodology is demonstrated by applying it to a linear parameter-varying (LPV) composite plate model with the integration of external toolboxes to incorporate a H_{∞} robust control design.

6.1 Co-Design Optimization Properties

The general mathematical formulation of the resulting hardware and control codesign optimization problem can be stated as:

$$\min_{\mathbb{Z}_{act},\mathbb{Z}_{sen},\mathbb{B}_{\Upsilon},\mathbb{R}_{tuning}} FV(\mathbb{Z}_{act},\mathbb{Z}_{sen},\mathbb{B}_{\Upsilon},\mathbb{R}_{tuning})$$
subject to
$$[\mathbb{Z}_{act},\mathbb{Z}_{sen},\mathbb{B}_{\Upsilon}] \in \mathbb{Z}$$

$$0 \leq \mathbb{Z}_{act} \leq p$$

$$0 \leq \mathbb{Z}_{sen} \leq q$$

$$0 \leq \mathbb{B}_{\Upsilon} \leq 1$$

$$LB_{tuning} \leq \mathbb{R}_{tuning} \leq UB_{tuning}$$

$$NonlinConstrF(\mathbb{Z}_{act},\mathbb{Z}_{sen},\mathbb{B}_{\Upsilon},\mathbb{R}_{tuning}) = \mathbf{c} \leq 0,$$
(6.1)

with FV the fitness value (calculated using the non-linear objective function), p the number of possible actuator types, q the number of possible sensor types, LB_{tuning} and UB_{tuning} the lower and upper bounds on the controller tuning parameters, and NonlinConstrF the non-linear constraint function.

6.1.1 Objective Function

An optimization problem always comes with an 'objective function' to specify the optimization goal (as introduced in Section 2.2 on page 21). Using this objective function, a 'fitness value' (FV) is calculated that quantifies the performance of the system for a specific set of design parameters. The proposed optimization methodology presented in this PhD dissertation allows a great freedom in determining the fitness value. For example, the methodology can handle discontinuous objective functions in which multiple objectives can be taken into account simultaneously, such as reference tracking, vibration levels, settling times, energy consumption, etc. This is done by calculating a weighted average according to their importance. Additionally, external toolboxes can be used in the objective function to determine the fitness value, such as MATLAB[®] Simulink for the graphical programming of control loops or external toolboxes like LCToolbox [202] to apply a H_{∞} framework and perform robust controller design with frequency-domain criteria.

6.1.2 Design Parameters

As previously mentioned, the design parameters for the co-design optimization methodology presented in this PhD dissertation consist of the actuator selection integers \mathbb{Z}_{act} , sensor selection integers \mathbb{Z}_{sen} , control architecture binaries \mathbb{B}_{Υ} , and the controller tuning parameters \mathbb{R}_{tuning} . The actuator placement binaries \mathbb{B}_{act} and sensor placement binaries \mathbb{B}_{sen} change according to the actuator selection integers

 \mathbb{Z}_{act} and sensor selection integers \mathbb{Z}_{sen} , respectively, and therefore do not directly belong to the design parameters.

The co-design optimization algorithm must deal with binary numbers, integer values, and real numbers, resulting in a mixed-integer optimization problem. This also implies that the objective function is discontinuous and, therefore, non-smooth and non-convex. One reason for this behavior is because the actuator selection integers determine the presence or absence of actuators, which has a non-linear impact on the calculated fitness value. A schematic overview of the design parameters is depicted in Table 6.1.

Table 6.1: Overview of the design parameters, with p and q being the number of possible actuator and sensor types, respectively

Design Parameters	Туре		Grouped in
Actuator selection	integer	$\in \mathbb{Z}^+ \le p$	\mathbb{Z}_{act}
Sensor selection	integer	$\in \mathbb{Z}^+ \le q$	\mathbb{Z}_{sen}
Control architecture binaries	binary	$\in \mathbb{B}$	\mathbb{B}_{Υ}
Controller tuning parameters	real	$\in \mathbb{R}$	\mathbb{R}_{tuning}

6.1.3 Constraints

Various constraints apply in the co-design optimization problem. Integer constraints are implemented on the actuator selection integers \mathbb{Z}_{act} , sensor selection integers \mathbb{Z}_{sen} , and control architecture binaries \mathbb{B}_{Υ} , forcing these design parameters to attain integer values. In addition, the hardware selection integer values \mathbb{Z}_{act} and \mathbb{Z}_{sen} are also limited according to the number of possible actuator and sensor types, denoted as p and q, respectively. The upper bound on the control architecture binaries \mathbb{B}_{Υ} is equal to one, and the controller tuning parameters are real value types. Upper bounds are also defined for these controller tuning parameters to limit the design space. If an optimized controller tuning parameter is very close to its upper bound, this may indicate that the result could be improved if that parameter reaches a larger value than the corresponding upper bound. If this is the case and a larger value is feasible, it is recommended to perform the optimization again with an increased upper bound on that design parameter.

An example of a non-linear constraint is that the optimization algorithm should ensure that the individual's implementation cost 'CostTotalActual' does not exceed a predefined constraint on the maximum total implementation cost 'CostTotalMaximum.' This is done in a so-called 'non-linear constraint function' by calculating the current individual's implementation cost 'CostTotalActual' based in the current design parameters and assigning the value in a vector c as:

$$\mathbf{c} = CostTotalActual - CostTotalMaximum, \tag{6.2}$$

while the Genetic Algorithm ensures that $\mathbf{c} \leq 0$. Additionally, other generally applicable non-linear constraints on, for example, the maximum actuator effort or

tolerances on mechanical motion aspects can simultaneously be taken into account by defining additional values in the c vector based on the active design parameters and the corresponding system response. This allows the efficient programming of one or more non-linear constraints. As with the objective function, there is a great freedom in applying different software packages and toolboxes to determine the values in c.

6.2 Genetic Algorithm Implementation

As discussed in Section 3.4 on page 44, a Genetic Algorithm (GA) is the preferred optimization algorithm to perform the hardware and control co-design methodology presented in this PhD. A Genetic Algorithm allows the implementation of a non-linear objective function with (mixed-integer) constraints provided in a nonlinear constraint function. In this way, the optimal hardware and control parameters can be determined for a maximum total implementation cost. By executing the Genetic Algorithm for several different maximum implementation costs, a Pareto front can be established that graphically shows the maximum achievable performance of the system as a function of the total implementation cost (see Section 3.5 on page 58).

Fig. 6.1 shows a detailed overview of a Genetic Algorithm workflow utilized in the hardware and control co-design optimization problem. The algorithm starts with the specification of an initial population consisting of a number of individuals equal to the population size (iPop). Each individual consists of a specific combination of design parameters. Increasing the population size enables the Genetic Algorithm to search more points in one generation. However, the larger the population size, the longer the Genetic Algorithm takes to compute each generation. Based on the many optimization runs for the cases studied, it appears that a population size of 300-500 individuals is appropriate for the type of co-design optimization problems described in this PhD. The design parameters are denoted as \mathbb{B} , \mathbb{Z} , and \mathbb{R} for binary, integer and real numbers, respectively. The initial population is generated using a creation function that defines random variables for every individual (within the predefined bounds for every design parameter).

Additionally, an 'initial population matrix' (IPM) can be defined to include individuals with specific design parameters directly in the initial population. Each IPM row corresponds to an individual combination of design parameters. If the number of rows in the IPM is smaller than the population size, the remainder of the initial population is completed with randomly generated individuals. Using an IPM has the advantage that a large and varied initial search space can be applied. However, it is important that the IPM contains feasible individuals. This way, the Genetic Algorithm will not 'lose' time in calculating the fitness values of infeasible situations and will immediately proceed to apply the objective function to feasible initial individuals. After the initial population is established, it becomes the active population for the first generation of the Genetic Algorithm. Next, the fitness value for each individual is determined based on the non-linear objective function. At the same time, a non-linear constraint function is used to check that no constraints are violated. If this is the case, the fitness value is increased so that individuals with a constraint violation have little to no chance to proceed to the next generation.

After determining the fitness value of each individual, the stopping criteria are checked. These stopping criteria are listed hereafter. The Genetic Algorithm is terminated if a maximum total calculation time 'Maximum time' is exceeded, a maximum number of generations is reached, or the average change in the best fitness value over 'Maximum stall generations' is less than or equal to the 'Function tolerance' value. The value for 'Maximum stall generations' is typically 50, while an appropriate 'Function tolerance' value depends on the fitness values of the optimized solutions. If one of the stopping criteria is met, the optimal solution is defined as the individual in the current generation with the lowest fitness value.

If no stopping criteria are met, a selection of 'parents' is made from the active population. This is done by making a stochastic choice of individuals from the active population in which the fitness value determines the probability of the choice. Individuals with lower fitness values have more chance to be chosen as parents. From these parents, the children for the next generation are created using three 'reproduction functions': 'elite selection,' 'crossover,' and 'mutation.' 'Elite count' indicates how many individuals go directly and unchanged to the next generation through elite selection. This value should always be lower than the population size and is usually around 10-20 individuals for this type of optimization problems [203]. After elite selection, the rest of the children are determined via crossover and mutation. A crossover child is formed by randomly combining variables from two parent individuals. In this regard, the child's *i*-th design parameter is always determined from the parents' *i*-th design parameters. Thus, no crossover can occur between design parameters with different data types (binary, integer, or real) [204]. A mutation child is created by randomly making changes in the variables of a parent individual based on a uniformly distributed random number and with respect to the corresponding lower and upper bounds on the design parameters [205]. The 'crossover fraction' value specifies the fraction of the remaining children determined via crossover (and not via mutation). For optimization problems with a discontinuous objective function, it is recommended to set the crossover fraction relatively low [206]. In this way, a relatively large amount of mutation occurs when determining the children, resulting in a broad search space for subsequent generations, preventing the optimization algorithm from prematurely ending up in a local minimum. An appropriate 'crossover fraction' value is 40%for co-design optimization problems described in this PhD. Refer to [151] for more in-depth information on the technical operation of a Genetic Algorithm.



Figure 6.1: Detailed overview of the workflow for a Genetic Algorithm. Binary, integer and real numbers are presented using the symbols \mathbb{B} , \mathbb{Z} , and \mathbb{R} , respectively. Operations in which a form of randomness applies are indicated with a dice symbol.

6.3 LTI Case: Mechanical Synchronization Setup

This section explains how to apply the hardware and control co-design optimization algorithm to a mechanical synchronization model, introduced in 3.6.4 on page 64 and used as an example throughout the previous chapters.

6.3.1 Objective Function

Before calculating the closed-loop system response in the non-linear objective function NonlinObjF, the actuator and sensor selection integers are compared with the open-loop analysis results (see Section 4.3.1). If the open-loop analysis results show that the related hardware architecture leads to an infeasible situation, the closed-loop response is not calculated, and the individual gets a very high fitness value of 10^{12} . In this way, no calculation time is wasted on calculating predetermined infeasible situations.

Conversely, if it turns out from the open-loop analysis that the current individual does have a feasible hardware architecture, the closed-loop state-space methodology (see Chapter 5) is applied in the objective function to efficiently determine the closed-loop response of the system based on the related design parameters. As mentioned earlier, many fitness function criteria can be applied simultaneously to perform a multi-objective optimization. For this mechanical synchronization case, the simulations are run with a total simulation time $T_m = 30s$, sample time $T_s = 0.01s$, and a corresponding total number of samples $K = \frac{T_m}{T_s} = 3000$. The fitness value to be minimized is a weighted sum consisting of three parts, calculated using (6.3) below.

$$FV = w_1 \left(\sum_{k=1}^{K} (r(k) - \theta_{load}(k))^2 \right) + \dots$$

$$w_2 \left(\sqrt{\frac{1}{K} \sum_{k=1}^{K} \left| \ddot{\theta}_1 \right|^2} + \sqrt{\frac{1}{K} \sum_{k=1}^{K} \left| \ddot{\theta}_2 \right|^2} + \sqrt{\frac{1}{K} \sum_{k=1}^{K} \left| \ddot{\theta}_{load} \right|^2} \right) + \dots$$

$$w_3 \left(\sum_{k=1}^{K} (\theta_{load}(k) - \theta_1(k))^2 + \sum_{k=1}^{K} (\theta_{load}(k) - \theta_2(k))^2 \right),$$
(6.3)
with $w_1 = 30$ = weight on load inertia trajectory tracking

with $w_1 = 30$ = weight on load inertia trajectory tracking $w_2 = 20$ = weight on acceleration reduction $w_3 = 2$ = weight on inertia synchronization

The first and most important objective is to have an optimal reference trajectory r(k) position tracking θ_{load} of the load inertia (see Fig. 3.13(a) on page 67). For this, the ISE (Integral of Square Error) performance index evaluates the central

load reference tracking [207], as can be seen in the first part of (6.3). The second objective is to reduce undesired variations in inertia accelerations, as this unwanted behavior reduces the life span of the system. Therefore, the accelerations $\ddot{\theta}_1$, $\ddot{\theta}_2$, and $\ddot{\theta}_{load}$ on all inertias are penalized by adding the RMS (Root Mean Square) value of these accelerations to the second part of the fitness value calculation. Furthermore, the synchronization between the inertias themselves is also considered by applying the ISE error performance index.

6.3.2 Design Parameters

The design parameters to be optimized by the Genetic Algorithm for this application are two actuator selection integers $(i_{act,1} \text{ and } i_{act,2})$, two sensor selection integers $(i_{sen,1} \text{ and } i_{sen,2})$, seven control architecture binaries $(b_{a,dec}, b_{b,dis}, b_{b,dec}, b_{b,dis}, b_{a,sc}, b_{b,sc}, \text{ and } b_{b,ff})$, and 32 real numbers for the controller tuning parameters (grouped in \mathbf{R}_{tuning}). These add up to 43 design parameters in total.

For this mechanical synchronization case, the actuator selection integers, sensor selection integers, control architecture binaries, and controller tuning parameters were detailed earlier in this work in Sections 4.2.1, 4.2.1, 5.1.4, and 5.2.1, respectively.

6.3.3 Constraints

Even if the controller parameter optimization is not considered, there are already 18432 different possible combinations for the hardware selection integers and control architecture binaries, each with a corresponding cost. It would certainly not be efficient to determine the optimal control parameters for each possible system composition. Therefore, the optimal system composition is only determined for a limited number of maximum total system costs CostTotalMaximum, ranging from 6,67% to 100% with intervals of 6,67%. A cost of 100% corresponds to the most expensive solution. In this way, 15 Pareto points are determined. The Genetic Algorithm is applied for every Pareto point, with a maximum total system cost CostTotalMaximum defined in the non-linear constraint function NonlinConstrF. In this function, the total cost of the current individual CostTotalActual is calculated and compared to the maximum implementation cost CostTotalMaximum (see (6.2)).

When calculating the response of the closed-loop state-space system, the actuator output saturation is taken into account by limiting the associated actuator input signals (see Chapter 5). The maximum actuator output $u_{max,act...}$ depends on the corresponding active actuator type. Next to a constraint on the maximum implementation cost, a constraint is imposed in the non-linear constraint function NonlinConstrF to prevent the actuator from showing unstable behavior by switching too fast between its output saturation. Fig. 6.2 shows the displacement and actuator effort response for a situation in which the actuator output shows this



Figure 6.2: Time-domain displacement response (a) and actuator effort response (b) for an example in which the actuators show unstable behavior of rapidly switching between their maximum actuator output limits (from approximately 5 to 18 seconds)

unwanted unstable switching between its maximum output limits from approximately 5 to 18 seconds. The actuators switch so quickly between their saturation limits that the responses are not plotted as lines but instead appear as areas. Based on prior knowledge of the reference trajectory, a stable controller should not direct the actuator according to its minimum saturation within 10 seconds of simulation time. Therefore, a constraint is imposed that prevents the actuator effort response from reaching its negative saturation value before 10 seconds have passed in the response simulation. The effect of the constraint is that none of the obtained optimization results exhibit this unstable behavior (see Appendix B).

As explained in Subsection 6.1.3, the constraints are applied in the non-linear constraint function NonlinConstrF by ensuring that the values of a vector **c** are less than or equal to zero. Equation (6.4) shows how the constraints mentioned above can be programmed in the non-linear constraint function NonlinConstrF.

$$\mathbf{c} \leq 0, \text{ with}$$

$$\mathbf{c} = \begin{bmatrix} CostTotalActual - CostTotalMaximum\\ -min\left(u_{act1}(1:1000)\right) - 0.95 \cdot u_{max,act1}\\ -min\left(u_{act2}(1:1000)\right) - 0.95 \cdot u_{max,act2} \end{bmatrix}$$
(6.4)

Next to these constraints, there are also upper and lower bound constraints on all design parameters. Table 6.2 shows an overview of these design parameters with the corresponding lower bound and upper bound constraints.

Design			Design			
parameter	LB	UB	parameter	LB	UB	
i _{act,1}	0	3	$i_{act,2}$	0	3	
$i_{sen,1}$	0	2	$i_{sen,2}$	0	2	
$b_{a,dec}$	0	1	$b_{b,dec}$	0	1	
$b_{a,dis}$	0	1	$b_{b,dis}$	0	1	
$b_{a,sc}$	0	1	$b_{b,sc}$	0	1	
			$b_{b,ff}$	0	1	
$K_{p,a,1}$	0	20	$K_{p,b,1}$	0	20	
$T_{i,a,1}$	0	1	$T_{i,b,1}$	0	5	
$T_{d,a,1}$	0	1	$T_{d,b,1}$	0	1	
$K_{p,a,2}$	0	20	$K_{p,b,2}$	0	20	
$T_{i,a,2}$	0	1	$T_{i,b,2}$	0	5	
$T_{d,a,2}$	0	1	$T_{d,b,2}$	0	1	
$K_{p,a,1-2}$	0	20	$K_{p,b,1-2}$	0	20	
$T_{i,a,1-2}$	0	1	$T_{i,b,1-2}$	0	1	
$T_{d,a,1-2}$	0	1	$T_{d,b,1-2}$	0	1	
$K_{p,a,2-1}$	0	20	$K_{p,b,2-1}$	0	20	
$T_{i,a,2-1}$	0	1	$T_{i,b,2-1}$	0	1	
$T_{d,a,2-1}$	0	1	$T_{d,b,2-1}$	0	1	
$K_{p,a,sc}$	0	20	$K_{p,b,sc}$	0	20	
$T_{i,a,sc}$	0	1	$T_{i,b,sc}$	0	1	
$T_{d,a,sc}$	0	1	$T_{d,b,sc}$	0	1	
$K_{ff,b,1}$	0	20	$K_{ff,b,2}$	0	20	1

Table 6.2: Enumeration of the design parameters for the mechanical synchronization case with corresponding upper bound (UB) and lower bound (LB) constraints

6.3.4 Genetic Algorithm Implementation

A Genetic Algorithm is applied to define the optimal values for the design parameters according to the non-linear fitness function and non-linear constraints. A mathematical formulation of this optimization problem is specified in (6.1). An 'initial population matrix' (IPM) can be provided to incorporate a large initial search space in the Pareto point optimizations. This initial population matrix indicates the starting values of the initial population and should contain individuals with feasible design parameters. To define the IPM for this mechanical synchronization case, a Genetic Algorithm is applied on the controller tuning parameters for 225 different, but fixed combinations of hardware selection integers (\mathbb{Z}_{act} and \mathbb{Z}_{sen}) and control architecture binaries (\mathbb{B}_{Υ}) . For every combination, a Genetic Algorithm is applied to only optimize the controller tuning parameters with 250 seconds of calculation time. This calculation time is too short to find an optimal solution, but it is sufficient to provide a feasible combination of design parameters. Subsequently, these feasible individuals are used in the initial population matrix of the Genetic Algorithm for the hardware and control co-design. Fig. 6.3 shows an overview of the cost and fitness values for each individual in the initial population matrix (IPM), while a vertical zoom can be seen in Fig. 6.4. Note that there are no individuals present in the initial population matrix (IPM) with a cost below 20% because no feasible systems can be established for these costs.

The key settings for the Genetic Algorithm applied to the Pareto point optimizations are shown in Table 6.3. The maximum number of iterations is set to 'infinite.' In this way, no maximum number of iterations is given to the Genetic Algorithm as stopping criteria. The function tolerance is chosen relatively small with the purpose that the Genetic Algorithm would not conclude too quickly that a solution is found. The crossover fraction is set to have a high mutation level when determining a new generation of possible solutions. This is desirable to keep the search field large and not to end up at a local minimum. The total Pareto optimization took 7.3 hours of calculation and was performed on an Intel[®] Xeon[®] CPU @ 3.10 GHz with 64 GB of RAM.

Description	Value	Unit
Population size	500	individuals
Maximum calculation time	3600	seconds
Maximum number of generations	infinite	generations
Function tolerance	10^{-10}	1
Maximum stall generations	100	generations
Crossover fraction	40	%
Elite count	20	individuals

Table 6.3: Genetic Algorithm settings for one single Pareto point optimization for the mechanical synchronization case



Figure 6.3: Overview of the cost and fitness values for the individuals in the initial population matrix (IPM) for the mechanical synchronization case



Figure 6.4: Zoom of the cost and fitness values for the individuals in the initial population matrix (IPM) for the mechanical synchronization case

6.3.5 Optimization Results

Tables 6.4 and 6.5 depict the optimization results for every point in the Pareto front. Figures 6.5 and 6.6 show an overview and a vertical zoom of the resulting Pareto front together with the individuals in the initial population matrix. The time-domain displacement and control effort output responses for every Pareto point can be found in Appendix B.

The results show that very high fitness values are obtained for the first four Pareto points. This indicates that there is insufficient reference tracking and that these Pareto points do not yield a viable solution. Therefore, it can be concluded that a minimum cost of 26.6% should be implemented in order to obtain a minimum working condition. The fitness values and the time-domain responses of the various Pareto points (see Appendix B) show that the performance systematically improves as more cost is allowed. Each of these Pareto points represents an optimized situation depending on the implementation cost, for which the Pareto front provides a valuable understanding of the trade-off between the maximum achievable performance and the implementation cost. It will be up to the design engineer to make a final decision on which Pareto point is most appropriate for the specific application. However, without further information, Pareto point 8 with an associated cost of 43.2% and a fitness value of 29.05 seems the most interesting because the fitness value improves relatively little when more cost is allowed while reducing the implementation cost indicates a significant reduction in performance.

					Ha	rdwa	re sel	ectio	n inte	egers	and			Controller tuning										
					c	ontro	l arcl	nitect	ure b	inari	es			parameters										
Pareto point	Cost [%]	Fitness value []	Actuator 1 selection integer $(i_{act,1})$	Actuator 2 selection integer $(i_{act,2})$	Sensor 1 selection integer $(i_{sen,1})$	Sensor 2 selection integer $(i_{sen,2})$	Cascade level a decentralized control $(b_{a,dec})$	Cascade level a distributed control $(b_{a,dis})$	Cascade level b decentralized control $(b_{b,dec})$	Cascade level b distributed control $(b_{b,dis})$	Cascade level a synchronizing control $(b_{a,sc})$	Cascade level b synchronizing control $(b_{b,sc})$	Cascade level b feedforward control $(b_{b,ff})$	$[K_{p,a,1} T_{i,a,1} T_{d,a,1}]$	$[K_{p,a,2} T_{i,a,2} T_{d,a,2}]$	$[K_{p,a,1-2} \ T_{i,a,1-2} \ T_{d,a,1-2}]$	$[K_{p,a,2-1} T_{i,a,2-1} T_{d,a,2-1}]$	$[K_{p,b,1} \ T_{i,b,1} \ T_{d,b,1}]$	$[K_{p,b,2} T_{i,b,2} T_{d,b,2}]$	$[K_{p,b,1-2}T_{i,b,1-2}T_{d,b,1-2}]$	$[K_{p,b,2-1} T_{i,b,2-1} T_{d,b,2-1}]$	$[K_{p,a,sc} T_{i,a,sc} T_{d,a,sc}]$	$[K_{p,b,sc} T_{i,b,sc} T_{d,b,sc}]$	$[K_{ff,b,1} \; K_{ff,b,2}]$
1	0	$9e^{8}$	0	0	0	0	0	0	0	0	0	0	0											
2	8.8	$3e^8$	1	0	1	0	0	0	•	0	0	0	0											
3	20	2e'	2	0	1	0	0	0	0	0	0	0	0											
4	25.3	$5e^{5}$	1	2	0	1	0	0	0	0	0	0	0											
5	26.6	76.40	1	2	0	1	0	0	•	0	0	0	0											
6	28.8	65.41	1	2	0		0	0	•	•	0	0	0							<u> </u>				
/	39.7	45.37	2	2	0		•	•	0	0	0	0	0	-				see	Table	6.5				
8	43.2	29.05	2	2	0	1	•	•	•	•	0	0	0	_										
9	39.7	28.42	2	2	0	1	•	•	•	•	•	•	0	-										
10	49.9	20.75	3	3	0	1	•	•	•	•				-										
12	00.4	20.30	3	3	0	1	•	-	•	•	-	-		-										
12	90.7	20.33	3	3	1	1	-		•	•	•	-	•	-										
14	100	20.30	3	3	1	2	•	•	•	•	-	•	•	-										
	100	27.72	5	5	1																			

					Controller	• tuning para	ameters				
	$[K_{p,a,1} T_{i,a,1} T_{d,a,1}]$	$[K_{p,a,2} T_{i,a,2} T_{d,a,2}]$	$[K_{p,a,1-2} T_{i,a,1-2} T_{d,a,1-2}]$	$[K_{p,a,2-1} T_{i,a,2-1} T_{d,a,2-1}]$	$[K_{p,b,1} \; T_{i,b,1} \; T_{d,b,1}]$	$[K_{p,b,2} T_{i,b,2} T_{d,b,2}]$	$\llbracket K_{p,b,1-2} T_{i,b,1-2} T_{d,b,1-2} \rrbracket$	$\llbracket K_{p,b,2-1} \ T_{i,b,2-1} \ T_{d,b,2-1} \rrbracket$	$\left[K_{p,a,sc} T_{i,a,sc} T_{d,a,sc}\right]$	$[K_{p,b,sc} T_{i,b,sc} T_{d,b,sc}]$	$[K_{ff,b,1} K_{ff,b,2}]$
4	0	0	0	0	0	0	0	0	0	0	0
.6.	0	0	0	0	[20 1 1]	[0 5 0]	0	0	0	0	0
ble	0	0	0	0	0	0	0	0	0	0	0
Та	0	0	0	0	0	0	0	0	0	0	0
ee	0	0	0	0	[1 0 0.3]	[20 1 0.9]	0	0	0	0	0
01	0	0	0	0	[0.1 0 0]	[20 1 0.4]	[0.4 0.1 0]	[20 1 0.3]	0	0	0
	[1.3 0.1 0.6]	[19 1 0]	[19 0 1]	[0.1 1 0.5]	0	0	0	0	0	0	0
	[0.2 0.06 0]	[11 0.3 0]	[0.1 1 0.1]	[16 0.1 0]	[0.2 0.4 0]	[6 2 0.1]	[0 0.8 0.1]	[1 0.4 0.1]	0	0	0
	[0.04 0 0.3]	[11 0 0]	[0.1 1 0.1]	[15 0 0]	[3.6 2 0]	[7 2 0.1]	[0 1 0]	[0 0 0.9]	[0.1 0.1 0.1]	[0 0 0.9]	0
	[0.01 0.9 1]	[11 0 0]	[0.08 0 0]	[20 0.2 0]	[15 0 0]	[7 2 0.1]	[0 0.3 0]	[0 0.3 1]	0	0	0
	[0.1 0.3 1]	[11 0.1 0]	[0 0.5 0.8]	[20 0 0]	[0.1 0.4 0]	[8 2 0.1]	[0 1 0.9]	[0.5 0.1 0]	[0.2 0.1 0.1]	[0 0 1]	0
	[4.7 0.1 0.1]	[11 0 0]	[2.8 1 0]	[20 0.4 0]	[0 0 0.1]	[7 2 0]	[0.1 1 0.7]	[0 0.2 0.1]	[0.3 0.01 0]	[0 0 1]	[0 0.1]
	[11 0.1 0]	[1 0 0]	[11.600]	[0 0.3 0.4]	[4.1 1 0]	[11 5 0.1]	[8.8 4 0]	[0.2 0 0.1]	[0.5 0 0]	[10 0 0]	[0.1 0.1]
	[12.1 0 0]	[1 1 0.3]	[19.9 0 0]	[0.1 0.3 0]	[5 1 0]	[0 2 0]	[0.1 4.5 1]	[4.7 0.9 0]	[0 0.8 0.1]	[5 1 0.1]	[0.1 0.1]

Table 6.5: Pareto front results (rounded) for the mechanical synchronization application. Inactive controllers are indicated with 'o'.



Figure 6.5: Overview of the resulting Pareto front (blue) for the mechanical synchronization case optimization with initial population matrix (IPM) (red)



Figure 6.6: Vertical zoom of the resulting Pareto front (blue) for the mechanical synchronization case optimization with initial population matrix (IPM) (red)

6.3.6 Comparison to Existing Controller Tuning Methods

The optimized controller values are compared to existing controller tuning methods to validate the effectiveness of the controller tuning optimization as part of the presented co-design methodology. Keep in mind that these existing controller tuning methods do not include the determination of the hardware components nor the controller architecture aspects.

Linear-Quadratic Regulator

First, an existing linear-quadratic regulator (LQR) method is applied to obtain the controller tuning parameters. In an LQR control design, relative weights are given to the states and input variables by adjusting the diagonal matrix Q and R values, respectively. In turn, these values define the relative weights on the reference tracking and the actuator output. Based on these values, a cost function minimization routine determines the values of the feedback matrices [56]. Thus, the LQR controller design comes down to adjusting the weights of the Q and R matrices according to the desired system behavior. Still, finding suitable weights for this LQR method turns out to be another difficulty. The most straightforward choice is to take identity matrices I for the Q and R matrices. 'Bryson's rule' describes a different approach to determine these weights as the inverse of the square of the maximum permissible error value for the corresponding state or input [181]:

$$Q_{i,i} = \frac{1}{\text{maximum acceptable value of } (error_{states})^2}, i \in 1, 2, ..., n$$

$$Q = \begin{bmatrix} Q_{1,1} & 0 & \cdots & 0 \\ 0 & Q_{2,2} & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & Q_{n,n} \end{bmatrix},$$

$$R_{j,j} = \frac{1}{\text{maximum acceptable value of } (error_{inputs})^2}, j \in 1, 2, ..., l$$

$$R = \begin{bmatrix} R_{1,1} & 0 & \cdots & 0 \\ 0 & R_{2,2} & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & R_{l,l} \end{bmatrix},$$
(6.6)

with n being the number of states and l being the number of inputs.

Also, trial and error can be applied to improve the output response. For this mechanical synchronization case, the LQR control uses an integrator to eliminate the steady-state error. For this purpose, an extra state is added to the system to compute the integral of the error signal. For details on LQR control design with integral action, see [52, 159, 163, 208].

The control loop design for applying this LQR control method to the mechanical synchronization model is shown in Figure 6.8. For the weights of the Q and R matrices, identity matrices I, Bryson's rule, and trial and error were used and tested on the model with different types of actuators while taking into account their limitations. The results of this LQR control design are shown in Table 6.6. A comparison of these LQR results with the optimized Pareto front is depicted in Figure 6.7. The results show that none of the applied LQR methods yield a performance that is as good as the optimized Pareto points using the presented co-design methodology.



Figure 6.7: Comparison of the optimized Pareto front using the presented co-design methodology and the results of applying an LQR method to obtain the controller tuning parameters



Figure 6.8: Control loop design for the application of an LQR control method with the mechanical synchronization process matrices in light blue, state observer matrices in dark blue, disturbance in dark green, reference trajectories in light green, actuator output saturation in yellow, state feedback matrix K in red, and integral feedback matrix K_i in red

$[i_{act,1}$	$[i_{sen,1}$	Weights on Q and	Fitness	Cost
$i_{act,2}]$	$i_{sen,2}]$	R matrices	value []	[%]
[1 1]	[1 1]	$Q = I_7$ $R = I_2$	10^{8}	32.3
[1 1]	[1 1]	$Q = \left[\frac{1}{0.1^2} \frac{1}{0.1^2} \frac{1}{0.1^2} \frac{1}{0.05^2} \frac{1}{0.05^2} \frac{1}{0.05^2} \frac{1}{0.05^2} \frac{1}{0.01^2}\right]$ $R = \left[\frac{1}{r^2} \frac{1}{r^2}\right]$	$9\cdot 10^7$	32.3
[1 1]	[1 1]	$\begin{array}{c} [5^{-}5^{-}] \\ Q = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 10^{7} \end{bmatrix} \\ R = \begin{bmatrix} 0.01 & 0.01 \end{bmatrix}$	$8.5 \cdot 10^{7}$	32.3
[2 2]	[1 1]	$Q = I_7$ $R = I_2$	47077	48.3
[2 2]	[1 1]	$Q = \left[\frac{1}{0.1^2} \frac{1}{0.1^2} \frac{1}{0.1^2} \frac{1}{0.05^2} \frac{1}{0.05^2} \frac{1}{0.05^2} \frac{1}{0.01^2} \frac{1}{0.01^2}\right]$ $R = \left[\frac{1}{12} \frac{1}{12} $	2606	48.3
[2 2]	[1 1]	$Q = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 10^{0} \end{bmatrix}$ $R = \begin{bmatrix} 0.001 & 0.001 \end{bmatrix}$	93.35	48.3
[3 3]	[1 1]	$Q = I_7$ $R = I_2$	47077	74.9
[3 3]	[1 1]	$Q = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 10^7 \end{bmatrix}$ $R = \begin{bmatrix} \frac{1}{5^2} & \frac{1}{5^2} \end{bmatrix}$	2752	74.9
[3 3]	[1 1]	$Q = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 10^7 \end{bmatrix}$ $R = \begin{bmatrix} 0.001 & 0.001 \end{bmatrix}$	84.44	74.9

Table 6.6: Results for the LQR control design applied to the mechanical synchronization model

PID Autotuning

The obtained results of the co-design methodology are also compared with an internal MATLAB[®] Simulink autotuning procedure. For this purpose, the control scheme was established within MATLAB[®] Simulink, in which an autotuning can be performed on the PID blocks. However, this turned out to be impossible when the complete control architecture (see Fig. 5.8 on page 96) was active due to internal linearization errors within MATLAB®. Therefore, only the cascade level a and b decentralized PID controllers were kept (see Fig. 6.9) on which an autotuning procedure was performed, suggesting PID controller values with a perfect balance between aggressiveness and robustness. The autotuning results were tested on hardware configurations with actuators of type one, two, and three. The results are shown in Table 6.7, and the resulting time-domain responses for these configurations can be seen in Fig. 6.10, 6.11, and 6.12. In Fig. 6.10, it can be observed that the three inertias' rotational displacements strongly drift away from the desired reference trajectory, and thus no viable situation is obtained. Fig. 6.11 and 6.12 show a much better reference tracking by the inertias, but by comparing the resulting fitness values, it is shown that none of the 'conventional' controller tuning methods achieve the performance of the optimized controller tuning parameters from the co-design methodology.

 Table 6.7: Results after applying the MATLAB[®] Simulink PID autotuning procedure to the mechanical synchronization model

$[i_{act,1},i_{act,2}]$	$[i_{sen,1} \ i_{sen,2}]$	$[K_{p,a,1} T_{i,a,1} T_{d,a,1}]$	$[K_{p,a,2} T_{i,a,2} T_{d,a,2}]$	$[K_{p,b,1}T_{i,b,1}T_{d,b,1}]$	$[K_{p,b,2}T_{i,b,2}T_{d,b,2}]$	Fitness value []	Cost [%]
[1 1]	[1 1]					10^{8}	32.3
[2 2]	[1 1]	[5.6 0.3 0.1]	[19.1 0.3 -139]	[1.7 0.3 0.6]	[0 0.4 0]	4912	48.3
[3 3]	[1 1]					1986	74.9



Figure 6.9: Control scheme used for the MATLAB[®] Simulink autotuning procedure with the mechanical synchronization process matrices in light blue, state observer matrices in dark blue, disturbance in dark green, reference trajectories in light green, actuator output saturation in yellow, and PID controllers in red



Figure 6.10: Time-domain response of the inertia displacement (top) and actuator effort (bottom) after applying the MATLAB[®] Simulink PID autotuning procedure with actuators of type 1. The rotational displacements strongly drift away from the desired reference trajectory.



Figure 6.11: Time-domain response of the inertia displacement (top) and actuator effort (bottom) after applying the MATLAB[®] Simulink PID autotuning procedure with actuators of type 2. The rotational displacement of the inertias show poor reference tracking.



Figure 6.12: Time-domain response of the inertia displacement (top) and actuator effort (bottom) after applying the MATLAB[®] Simulink PID autotuning procedure with actuators of type 3. The rotational displacement of the inertias show mediocre reference tracking.

6.3.7 Objective Function Surface Plots

A surface plot can be established to validate the obtained optimization results. For this purpose, a 'mesh' is created around an optimization result where two design parameters (called 'mesh variables') change in a specific area around the obtained optimization point with predefined intervals. The optimization routine has correctly determined the optimal solution if none of these surrounding points show a better fitness value. Additionally, the smoothness of the objective function can also be checked in this way (see 2.2 on page 21). Table 6.8 shows the various settings for the surface plots that are mentioned hereafter.

Pareto point	Mesh variables	Mesh variable limits	Z-axis limits	Figure reference
10	$\begin{array}{c c} K_{p,a,2} \\ K_{p,a,2-1} \end{array}$	$\begin{bmatrix} 0 & 25 \\ 0 & 40 \end{bmatrix}$	[26 50]	Fig. 6.13
10	$\begin{array}{c} K_{p,b,2} \\ K_{p,b,1} \end{array}$	$\begin{bmatrix} 0 & 15 \\ 0 & 30 \end{bmatrix}$	[26 100]	Fig. 6.14
14	$\begin{array}{c c} K_{b,ff,2} \\ K_{b,ff,1} \end{array}$	$\begin{bmatrix} 0 & 20 \\ 0 & 20 \end{bmatrix}$	$[0 \ 10^7]$	Fig. 6.15

 Table 6.8: Overview of the settings for the different surface plots for the mechanical synchronization case optimization results

Fig. 6.13 shows the surface plot for the optimized Pareto point 10 with mesh variables $K_{p,a,2}$ and $K_{p,a,2-1}$. From the surface plot results, it can be seen that the optimization algorithm successfully determined the optimal point for these two design parameters within the intended range, since no point in the mesh has a better fitness value than the optimization result from the presented co-design methodology. Additionally, it can be seen in Fig. 6.14 that the objective function exhibits non-convex behavior (even for 'continuous' design parameters $K_{p,b,2}$ and $K_{p,b,1}$ for which there is no integer constraint). The objective function is not only non-convex, but it also exhibits non-smooth behavior, as can be seen on the surface plot of design parameters $K_{b,ff,2}$ and $K_{b,ff,1}$ in Fig. 6.15. In this way, it is validated that the presented co-design methodology can find the optimal solution for non-smooth objective functions. This justifies the choice of a Genetic Algorithm as the optimization algorithm used for this type of optimization problem.



Figure 6.13: Angle view (a) and top view (b) on the objective function surface plot with mesh variables $K_{p,a,2}$ and $K_{p,a,2-1}$ for z-axis limits [26 50]. The dashed line in (a) and the white point in (b) represent the optimization result for Pareto point 10.



Figure 6.14: Angle view (a) and top view (b) on the objective function surface plot with mesh variables $K_{p,b,2}$ and $K_{p,b,1}$ for z-axis limits [26 100]. The dashed line in (a) and the white point in (b) represent the optimization result for Pareto point 10.





Figure 6.15: Angle view (a) and top view (b) on the objective function surface plot with mesh variables $K_{b,ff,2}$ and $K_{b,ff,1}$ for z-axis limits $[0 \ 13 \cdot 10^7]$.
6.4 LPV Case: Composite Plate

The proposed co-design methodology is applied to the active vibration control of a composite plate for which the controller design is performed using the Linear Control Toolbox (LCToolbox) [202] to obtain a robust H_{∞} controller based on frequency domain criteria. This application demonstrates the flexibility of the proposed co-design methodology in this PhD dissertation because this application incorporates a linear parameter-varying (LPV) state-space model. This application also demonstrates that external toolboxes can be used within the non-linear objective and constraint functions of the proposed co-design methodology. The main objective is to obtain a Pareto front that shows the optimal selection of collocated actuator/sensor pairs and their corresponding fitness values for the active vibration control of a composite plate with an H_{∞} LPV control design. For this purpose, the general co-design workflow (see Fig. 3.1 on page 40) is used in which some steps are not relevant for this case. Fig. 6.16 shows the workflow for the co-design methodology applied to this linear parameter-varying (LPV) composite plate model.



Figure 6.16: Workflow for the co-design methodology applied to an LPV composite plate model



Figure 6.17: Overview of the simplified composite plate model with force disturbances d_1 , d_2 , d_3 , and d_4 and possible locations for the collocated actuator/sensor pairs (from [210])

6.4.1 Model Properties

The setup consists of a composite plate of dimensions [503mm x 400mm x 2.5mm] composed of unidirectional carbon fiber laminates with a symmetric lay-up of $[-45^{\circ}45^{\circ}0^{\circ}90^{\circ}]_s$ [209]. The plate is equipped with nine M2814-P1 macro-fiber composite (MFC) actuators of dimensions [28mm x 14mm x 0.3mm] and nine accelerometers. Force disturbances d_1 , d_2 , d_3 , and d_4 act on the plate at four locations, as shown in Fig. 6.17. Since the weights of the accelerometers (approximately 5 grams per unit) are negligible compared to the actuator weights, the mechanical influence of the sensors on the model dynamics is neglected.

The state-space model is depicted as:

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ -M^{-1}\Theta \end{bmatrix} u + \begin{bmatrix} 0 \\ -M^{-1}L \end{bmatrix} w$$

$$= A \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + B_u u + B_w w,$$

$$(6.7)$$

with displacement q, mass matrix M, damping matrix C, spring matrix K, piezoelectric coupling matrix Θ , disturbance localization matrix L, force disturbances w, and input voltages u. The acceleration output y is defined as:

$$y = \begin{bmatrix} 0 & C \end{bmatrix} A \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 & C \end{bmatrix} \begin{bmatrix} B_u & B_w \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix}$$

= $C \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + D \begin{bmatrix} u \\ w \end{bmatrix}.$ (6.8)

This state-space model is a linear time-invariant (LTI) system. An artificial linear parameter-varying (LPV) system is constructed by parameterizing the system matrix A with α as the artificially generated scheduling parameter with values in the interval [0.05, 0.1] resulting in $A(\alpha(t)) = A + \alpha A$. More details on this model can be found in earlier work [210, 211].



Figure 6.18: Control configuration for the disturbance rejection control design problem with *l* actuators and sensors

6.4.2 Objective Function

The control objective is to accomplish the force disturbance rejection for the composite plate active vibration control. The corresponding control configuration is shown in Fig. 6.18, with $w = [d_1, d_2, d_3, d_4]^T$ representing the force disturbances and W_D and W_U being the static loop-shaping weights for the disturbance channel D and the input sensitivity channel U, respectively. The H_{∞} control design framework is used to obtain a full-order dynamic output feedback LPV controller K to minimize the closed-loop specification's cost function, as defined in (6.9).

minimize:
$$\left\| \begin{bmatrix} W_D D \\ W_U U \end{bmatrix} \right\|_{\infty}$$
 (6.9)

The weights are selected as static gains, being $W_D = 0.32I$ and $W_U = 10^{-2.25}I$ to reduce the vibrations at each performance channel and limit the actuator effort [183]. The LCToolbox is used to solve the control design problem and obtain a controller for a specific selection of active collocated actuator/sensor pairs. A ' γ ' value quantifies the closed-loop performance [183], with lower γ values representing better closed-loop performances. This γ value is used as the fitness value throughout this application.

6.4.3 Design Parameters

As mentioned earlier, the actuators and sensors are applied in collocated pairs. There are nine possible locations for these actuator/sensor pairs, resulting in nine hardware placement binaries $b_{act/sen}$ defining the presence or absence of the corresponding actuator/sensor pair. No different types of sensors or actuators are possible. The controller design follows from the LCToolbox calculations based on the active actuator/sensor pairs, so no controller tuning parameters need to be explicitly defined in the co-design optimization methodology.

6.4.4 Constraints

Integer constraints with boundaries [0 1] apply to each hardware placement binary. The Pareto front is established by determining the optimal selection of the active actuator/sensor pairs for each possible number of hardware placement binaries (= [1-9]). Therefore, for each Pareto point optimization, an additional constraint applies that ensures that the current number of actuator/sensor pairs is equal to the number of actuator/sensor pairs of the corresponding Pareto point.

6.4.5 Exhaustive Search

The nine binary design parameters result in 512 possible combinations, and for each possible combination, the control design is calculated during an exhaustive search approach. These exhaustive search results are depicted in Fig. 6.19 and took 107 minutes of total calculation time. All optimization calculations on the composite plate case were performed on the same Intel[®] Xeon[®] CPU @ 3.10 GHz with 64 GB of RAM.



Figure 6.19: Exhaustive search results for the composite plate optimization

6.4.6 Genetic Algorithm Implementation and Results

Optimal Selection of Five Actuator/Sensor Pairs

First, a Genetic Algorithm (GA) is used to determine the optimal selection of five collocated actuator/sensor pairs. From the exhaustive search, it can be concluded that the optimal γ value for five actuator/sensor pairs is equal to 145.02. Since the GA optimization is non-deterministic by nature, the optimization procedure is repeated five times with the same settings and starting conditions to check the result distribution. Table 6.9 depicts the GA settings for these composite plate GA optimizations, while the optimization results are shown in Table 6.10. The results show the optimized location selection of five actuator/sensor pairs for the consecutive GA optimizations, together with the corresponding fitness values and total calculation times. The results show that the optimal solution was found in three out of five optimizations. For the two non-optimal solutions, the GA apparently got stuck in a local minimum close to the global minimum.

Description	Value	Unit
Population size	30	individuals
Maximum calculation time	600	seconds
Maximum number of generations	infinite	generations
Function tolerance	10^{-5}	/
Maximum stall generations	10	generations
Crossover fraction	90	%
Elite count	4	individuals

Table 6.9: Genetic Algorithm settings for the composite plate application

Table 6.10: Composite plate GA optimization results for the optimal selection of five collocated actuator/sensor pairs. The optimizations for which the optimal solution is found are indicated in green, optimizations resulting in sub-optimal solutions are indicated in red.

Optimization	Optimized actuator/sensor	Fitness	Total calculation
algorithm	pair selection	value [γ]	time [min]
Exhaustive search	1,3,4,6,9	145.02	107
GA 1	1,3,4,6,9	145.02	3.54
GA 2	1,3,4,6,9	145.02	6.60
GA 3	1,4,5,6,9	146.47	5.05
GA 4	1,4,6,8,9	147.28	7.02
GA 5	1,3,4,6,9	145.02	7.01

Pareto Front Optimization

Next, a GA optimization is performed that defines the optimal selection for one to nine possible actuator/sensor pairs to establish a Pareto front. This optimization process is repeated three times to check the result distribution. Table 6.11 depicts the optimization results. The rows represent the various individual Pareto point optimizations involving the selection of the locations for a fixed number of actuator/sensor pairs. The columns represent the results of the different optimization algorithms, being the exhaustive search and the three consecutive Genetic Algorithms (GA). The resulting selection of locations for the actuator/sensor pairs is indicated, together with the corresponding fitness value. GA optimizations for which the optimal location of the actuator/sensor pairs was found are indicated in green. A graphical representation of these results is given in Fig. 6.20.

Table 6.11: Composite plate optimization results for the Pareto front optimizations. The optimizations for which the optimal solution is found are indicated in green, optimizations resulting in sub-optimal solutions are indicated in red.

Optimization	Exhaustive				
algorithm	search	GA 1	GA 2	GA 3	
1 act/sensor pair	4	4	4	4	
(Fitness value $[\gamma]$)	(234.70)	(234.70)	(234.70)	(234.70)	
2 act/sensor pairs	1,4	1,4	1,2	1,4	
(Fitness value $[\gamma]$)	(168.40)	(168.40)	(177,40)	(168.40)	
3 act/sensor pairs	1,4,6	1,4,6	1,4,6	1,4,6	
(Fitness value $[\gamma]$)	(155.88)	(155.88)	(155.88)	(155.88)	
4 act/sensor pairs	1,4,6,9	1,4,6,9	1,3,4,9	1,2,6,9	
(Fitness value $[\gamma]$)	(147.74)	(147.74)	(148,74)	(153.20)	
5 act/sensor pairs	1,3,4,6,9	1,3,4,5,9	1,3,4,6,9	1,3,6,8,9	
(Fitness value $[\gamma]$)	(145.02)	(147,43)	(145.02)	(150.15)	
6 act/sensor pairs	1,3,4,5,6,9	1,3,4,5,6,9	1,3,4,5,6,9	1,3,4,5,6,9	
(Fitness value $[\gamma]$)	(143.81)	(143.81)	(143.81)	(143.81)	
7 act/sensor pairs	1,3,4,5,6,7,9	1,2,3,4,6,7,9	1,3,4,5,6,7,9	1,2,3,4,5,6,9	
(Fitness value $[\gamma]$)	(143.22)	(144,02)	(143.22)	(143.43)	
8 act/sensor pairs	1,3,4,5,6,7,8,9	1,3,4,5,6,7,8,9	1,3,4,5,6,7,8,9	1,3,4,5,6,7,8,9	
(Fitness value $[\gamma]$)	(142.80)	(142.80)	(142.80)	(142.80)	
9 act/sensor pairs	1,2,3,4,5,6,7,8,9	1,2,3,4,5,6,7,8,9	1,2,3,4,5,6,7,8,9	1,2,3,4,5,6,7,8,9	
(Fitness value $[\gamma]$)	(142.43)	(142.43)	(142.43)	(142.43)	
Total calculation	107	25.23	30.79	24.42	
time [min]	107	25.25	50.79	27.72	



Figure 6.20: Pareto front results for the exhaustive search and the Genetic Algorithm (GA) optimizations on the composite plate case

6.4.7 Case Conclusions

The exhaustive search results show that the γ values for all possible combinations of five collocated actuator/sensor pairs range from 145.02 to 234.8. The GA optimization results for selecting five actuator/sensor pairs show that the optimal solution was found in three out of five optimizations, with a considerably shorter calculation time compared to the exhaustive search approach. It can also be concluded that the γ values for the sub-optimal solutions from the GA optimizations are very close to the optimal solution. Considering the full range of possible fitness values, the worst obtained result from the GA optimizations is only 2.5% worse than the optimal result.

The Genetic Algorithm Pareto front optimizations show that the optimal selection of actuator/sensor pairs was successfully performed for most Pareto points. By combining multiple optimization results, the optimal Pareto front can be determined. Again, the non-optimal Pareto points are close to the optimal solutions.

This composite plate application shows that the proposed hardware architecture and control configuration co-design methodology can cope with linear parametervarying (LPV) systems and H_{∞} control design using the LCToolbox. Further extensions on the co-design methodology are possible, but are not worked out in this dissertation. For example, the previously static weights $W_D = 0.32I$ and $W_U = 10^{-2.25}I$ can also be defined as variable design parameters and can be optimized based on time-domain system response criteria. In this way, criteria in both the time domain and the frequency domain can be considered in the co-design optimization methodology. Additionally, an extension can be introduced in which a consideration of different types of actuators and sensors could be made. For this purpose, actuator and sensor selection integers could be used, as described in Section 4.2. An additional extension is to remove the restriction that the actuators and sensors must be implemented in pairs.

6.5 Chapter Conclusions

This chapter explains how a Genetic Algorithm (GA) can be applied to execute the presented hardware architecture and control configuration co-design methodology. A description of the objective function, design parameters, and constraints concerning this optimization problem is given. The total calculation time can be reduced by integrating specific prior knowledge from an open-loop analysis. It can be concluded from the optimization results that a GA is able to determine the optimal system composition with the maximum achievable performance depending on the intended implementation cost and other (non-linear) constraints. The results are graphically represented in a Pareto front and can significantly help the design engineer to obtain a better understanding of the trade-off between the optimal achievable performance and the total implementation cost.

The hardware architecture and control configuration co-design optimization is performed on a mechanical synchronization case of which some properties have already been covered in the previous chapters. The optimization results are compared to existing linear-quadratic regulator (LQR) and PID autotuning methods, showing that none of the existing controller tuning methods achieve the performance of the presented optimization results. These optimization results are further validated by checking that none of the configurations with surrounding design parameter values has better performance. For this purpose, the objective function surface plots are established. These surface plots are also used to identify the smoothness of the objective function, emphasizing that the objective function exhibits strong nonsmooth behavior for some design parameters (even without considering the design parameters with integer constraints).

Finally, the flexibility of the presented co-design methodology is demonstrated by applying it to a linear parameter-varying (LPV) composite plate model with the integration of an external LCToolbox to incorporate a H_{∞} robust control design. For this application, the optimal selection of collocated actuator/sensor pairs is performed for various numbers of possible hardware components. The optimization results are compared to an exhaustive search approach, showing that the optimal solution is found in the majority of the optimization results with much shorter calculation times. Moreover, sub-optimal optimization results are found to be close to the optimal solutions.

Chapter 7

Case: Active Car Suspension Setup

In this chapter, the presented hardware architecture and control configuration codesign methodology is applied to the state-space model of a physical downscaled active car suspension lab setup. This setup is designed based on the widely available theoretical full-car active suspension model and mimics a car driving over a road surface while active components in the suspension are used to increase the driver's comfort by counteracting unwanted vibrations. Kinematic and dynamic analyses are performed to ensure that the system behavior matches a typical passenger car.

The result of the co-design optimization methodology is a Pareto front that graphically represents the trade-off between the maximum performance and the total implementation cost. Additionally, the co-design results are validated with measurements on the physical active car suspension setup, and the obtained controller tuning parameters are compared with existing controller tuning methods.

7.1 Introduction on Active Suspension Systems

An automotive suspension system exists in many forms and designs but usually consists of a mechanical linkage system with springs and dampers (often called 'shock absorbers') that connect the wheels to the car and allow relative movement between the two [212]. This suspension is designed to provide a desirable trade-off between road handling and ride quality. A stiff suspension maintains good road handling capabilities but causes the driver to perceive much of the unevenness in the road surface. On the other hand, a soft suspension provides greater comfort for the driver on uneven road surfaces, but good contact between the wheels and the road surface cannot be guaranteed. This is crucial for road handling because all the forces between the car and the road surface are transmitted through the wheels. The vast majority of ordinary passenger cars use a passive suspension with a spring



Figure 7.1: Overview of a typical passenger car passive suspension system with a spring (yellow) and a damper or shock absorber (blue) (edited from [214])

that absorbs impacts and a damper (or shock absorber) that limits the spring movements. Fig. 7.1 shows a spring-damper system of a typical passive car suspension. For everyday use passenger cars, optimal ride comfort is the suspension system's principal goal. To further improve this ride comfort, active suspension systems can be used in which the suspension is externally controlled to attenuate the vibration of the vehicle body. Over the past decades, active suspension control technologies have become an extensive research topic, hence these systems have a significant influence on the subjective driver comfort impression [213]. A distinction is made between semi-active (or adaptive) and active suspension systems.

7.1.1 Semi-Active and Adaptive Suspension Systems

Semi-active and adaptive systems can only change the damping coefficient of the suspension and thus cannot inject energy directly into the system. Typically, adaptive systems allow for several predetermined damping coefficients for which the damping values are adjusted according to the desired driving mode (e.g., comfort, normal, sport). On the other hand, semi-active systems are equipped with components having a much faster response time to respond in real-time to changing

road conditions and car dynamics. The most basic and economical form of a semiactive suspension is using a solenoid valve that can adjust the flow of the hydraulic medium in the shock absorber, resulting in a change of the damping coefficient. The first production car to be equipped with this type of suspension was the Toyota Soarer in 1983 [215, 216].

A semi-active suspension can also be obtained by using magnetorheological dampers. These dampers contain a fluid with metal particles that align according to the magnetic field. The viscosity of the liquid depends on the alignment of the particles, allowing the damping coefficient to be adjusted according to an electric current through an electromagnet. This system can present very fast reaction times of a few milliseconds, allowing, for instance, to make a softer passing by a single wheel over a bump in the road [217–219]. Magnetorheological dampers were developed for production cars by 'General Motors' and first applied under the proprietary name 'MagneRide' on a Cadillac car in 2002 [220]. To this day, MagneRide technology continues to be applied in cars from manufacturers such as Chevrolet, Ferrari, Lamborghini, Range Rover, Audi, and Ford. In addition to automotive suspensions, magnetorheological fluids are also used in, for instance, prosthetic limbs [221] and stabilization systems for helicopter rotor blades [222].

7.1.2 Active Suspension Systems

With active suspensions, actuators are added that can effectively apply a force to certain suspension elements so that the displacement of each wheel can be controlled separately. These actuators can be fully hydraulic [223], whereby the vehicle body can be raised or lowered in a matter of seconds, improving the aerodynamic performance at high speeds. Probably the most well-known example of this hydraulic suspension is the Citroën DS, developed by Paul Magès in 1954 [224]. A more modern variant is an electronically controlled hydraulic system. In that case, sensors monitor the body movement while hydraulic servos provide counter forces during driving maneuvers. The Williams Grand Prix Engineering team equipped their F1 cars with such an active car suspension system in 1992. This turned out to be so profitable that the team won 10 races in that season's Grand Prix, after which the 'Fédération Internationale de l'Automobile' (FIA) banned the technology [225]. A variation of this system was introduced by Mercedes-Benz in 1999 in high-segment production cars as an 'Active Body Control.' This feature uses high-pressure hydraulic servos controlled to actively lean into curves and is still available today [226, 227].

Active car suspensions can also be implemented electromechanically, in which each wheel is equipped with electric motors. This form of active suspension provides an extremely fast response and also allows energy to be recovered by using the motors as generators. These electric systems exhibit better performance than the slower and more energy-consuming hydraulic systems. The 'University of Texas Center for Electromechanics' patented an electronically controlled active suspension system in 1999 that was later used in military vehicles [228]. This system exceeded all performance specifications in terms of absorbed power to the vehicle operator, stability, and handling [229]. The 'Bose Corporation' (most known for their audio products) made a proof of concept in 2004 of a passenger car equipped with an impressive electromechanical car suspension [230]. Due to its high cost and weight, the system was not implemented in production cars. The technology was sold to the company 'ClearMotion' that further developed this technology into a finished and implementable product [214]. The company even claims to greatly reduce motion sickness in cars [231]. Since 2017, Audi has supplied their A8 models with an advanced active suspension system, see Fig. 7.2. This active suspension is controlled via inputs from the forward-facing camera that 'reads' the road surface, transmitting signals to the control to raise or lower the suspension accordingly to cope with the road surface. The system has the capability of generating 16 kN per second and alters the suspension every 15 milliseconds [232, 233].



Figure 7.2: Overview of an Audi A8 active suspension system with the active spring-damper systems highlighted in yellow (edited from [233])

7.2 Active Car Suspension Lab Setup

7.2.1 Setup Properties

In the scope of this PhD, a setup representing a downscaled active car suspension is built, as shown in Fig. 7.3. This lab setup mimics the behavior of a car driving on a road surface in which active suspension components can counteract on vibrations of a central platform representing a car body. The actuators consist of servomotors from which a rotational torque is converted to a linear force through an electromechanical cylinder. The test setup allows the examination of a wide variety of hardware architectures and control configurations. It is important to emphasize that the primary goal of this setup is not to improve the current state-of-the-art in active car suspension systems but rather to validate the presented model-based co-design methodology with measurements on a physical setup.



Figure 7.3: Picture of the physical full-car active suspension lab setup. The locations of the sliders and the spring-damper-actuator systems are indicated in accordance with the diagram in Fig. 7.4. The linear slider systems for vertical guidance also have an actuator, but these are not connected to use the lab setup as an active car suspension. The springs are not present in this picture, but were mounted to perform the mentioned tests.

A theoretical full-car suspension model is widely available [234–237]. However, building a full-car setup based solely on this full-car suspension model is very hard as the system has a lot of degrees of freedom. Therefore, a novel active car suspension lab setup was built in order to successfully emulate a full-car active suspension. A diagram of this lab setup is shown in Fig. 7.4 and consists of a central, hexagonal platform with mass m_s supported in four places by a spring K, a damper B, and an actuator system with applied forces f in parallel. The spring-damper systems correspond to a classic car suspension, while the actuators represent an active component to counteract the unwanted vibrations of the central platform. The subscripts fr, fl, rr, and rl represent the wheel location, being front-right, front-left, rear-right, and rear-left, respectively. As a result, the lab setup can be labeled as a parallel robot, extensively studied in [238–240].



Figure 7.4: Full-car active suspension lab setup diagram

A kinematic study was performed to guarantee that the lab setup will have enough degrees of freedom to accurately emulate an active car suspension. The lower side of the spring, damper, and actuator rods cannot translate. This has as an effect that when the central mass is pitching along the Y-axis, the spring, damper, and actuator rods should be able to tilt from their initial vertical position. That is why these rods are attached to the base plate and the central mass m_s with ball couplings, providing rotational ability and fixed relative translation. Linear sliders are provided on the two opposite sides of the central platform, preventing the central mass from tipping over undesirably. As a result, the central platform can translate along the Z-axis with a heave height z and rotate around the Y-axis with an angular pitch rotation θ . This makes it possible to emulate a car driving straight forward on a road profile.

The parameters to represent a passenger car suspension model are obviously not the same for every car, but typical values can be found in previous work [65, 213, 234, 236, 241]. Additionally, the lab setup will not have the same dimensions of a regular passenger car. A geometrical downscaling factor of two is used for the lab setup compared to a regular passenger car. For the lab setup to have the same dynamics, its weight will not be half, but $2^3 = 8$ times smaller than a regular passenger car. This implies that the appropriate scaling laws must be applied for some fundamental parameters [236, 242]. Table 7.1 depicts the downsizing of typical passenger car parameters to the lab setup parameters. Although the lab setup parameters do not perfectly match the downsized values of a typical passenger car, they are close enough so that the lab setup will exhibit similar dynamic characteristics to a typical mid-size passenger car.

 Table 7.1: Downscaling typical passenger car parameters to lab setup parameters with a geometrical scaling factor 1:2

	Downsized value from a typical	Lab setup
	passenger car	parameter
Mass inertia (I_{yy}) $[kgm^2]$	$4000/2^5 = 125$	104.45
Suspension stiffness $(K_s) [N/m]$	23000/2 = 11500	13000
Damping coefficient (B_s) $[Ns/m]$	$6000/2^2 = 1500$	1800
Dimension front-rear $(a + b) [m]$	2.5/2 = 1.25	1
Body mass $(m_s) [kg]$	$1400/2^3 = 175$	58.26

7.2.2 State-Space Model Identification

The system has four inputs, namely the four actuator forces u_{fr} , u_{fl} , u_{rr} , and u_{rl} . These system inputs are used to apply both the road profile disturbances d_{fr} , d_{fl} , d_{rr} , and d_{rl} (see Subsection 7.2.3) and the control effort signals f_{fr} , f_{fl} , f_{rr} , and f_{rl} eliminating unwanted central platform vibrations. The applied feedback control has no information on the road disturbance signals. The model inputs are as follows:

$$\mathbf{u} = \begin{bmatrix} u_{fr} \\ u_{fl} \\ u_{rr} \\ u_{rl} \end{bmatrix} = \begin{bmatrix} d_{fr} + f_{fr} \\ d_{fl} + f_{fl} \\ d_{rr} + f_{rr} \\ d_{rl} + f_{rl} \end{bmatrix}.$$
 (7.1)

The model states x are the horizontal position z, speed \dot{z} , acceleration \ddot{z} , and pitch angle θ of the central platform:

$$\mathbf{x} = \begin{bmatrix} z \\ \dot{z} \\ \ddot{z} \\ \theta \end{bmatrix}, \tag{7.2}$$

while the same system properties are used for the model output \mathbf{y} for the identification procedure:

$$\mathbf{y} = \begin{bmatrix} z \\ \dot{z} \\ \ddot{z} \\ \theta \end{bmatrix}.$$
 (7.3)

As a result, the dimensions of the state-space system matrices are defined as A $[4 \times 4]$, B $[4 \times 4]$, C $[4 \times 4]$, and D $[4 \times 4]$. For the model identification, a multisine excitation signal with measurement time $T_m = 80$ s and sample frequency $F_s = 1000$ Hz containing sine frequencies from 0.01 Hz to 490 Hz [202] is imposed on the system inputs, while the system response is measured. A time delay of 0.1 s is imposed between the front and rear actuators to obtain an angular pitch rotation θ excitation. From the measured input and output signals, the model matrices values are identification procedure is graphically displayed in Fig. 7.5. The resulting (rounded) matrix values are shown in Equations 7.4-7.7.



Figure 7.5: Graphical representation of the model identification procedure using a Prediction Error Minimization (PEM) method to identify the system's state-space matrix values based on the imposed multisine excitation signals and the measured response signals

Fig. 7.6 depicts the Bode plot of the identification measurements using the multisine signal and the Bode plot of the identified PEM state-space model. It is important to emphasize that the model will not perfectly match the actual setup. For example, it is known in advance that the dampers have a non-linear character. This non-linear behavior (along with possibly other phenomena such as static friction) is not explicitly modeled in the identified linear time-invariant (LTI) system.



Figure 7.6: Comparison of the bode plots of the identification measurements (blue) and the identified PEM state-space system (red) from system inputs u to platform velocity \dot{z} (left) and system inputs u to platform acceleration \ddot{z} (right)

7.2.3 Road Profile

Road profile disturbance signals are applied to the actuators with the intention that the central platform experiences the same forces as if it was driving over a real road surface. According to the ISO 8608 norm [244], a road profile can be mathematically composed based on the assumption that a given road has equal statistical properties everywhere along a section to be classified. That is: the road surface is a combination of a large number of longer and shorter periodic bumps with different amplitudes. Another input parameter for the road profile formulation is the road roughness factor. This factor varies from 1 to 8, with 1 being a high-quality (smooth) road surface like an asphalt layer. Conversely, a road roughness factor of 8 represents a very poor road quality, as in roadway layers consisting of cobblestones [245, 246]. The road profile is defined as a displacement disturbance, while the lab setup only allows force disturbance inputs. That is why the road displacement z_r is converted to a corresponding force disturbance d for the active car suspension lab setup. This conversion is graphically represented in Fig. 7.7.



Figure 7.7: Graphical representation of the conversion from a road displacement z_r from a traditional quarter car model to a force disturbance d for the quarter-car suspension lab setup

Fig. 7.8(a) shows an ISO 8606 road profile for a car traveling at 72 km/h for 20 seconds on a road profile of 400 meters with a roughness factor 5. Fig. 7.8(b) shows this displacement profile converted to an equivalent force profile, applicable to the lab setup. These actuator force disturbances are appointed as d_{fr} , d_{fl} , d_{rr} , and d_{rl} for the front-right, front-left, rear-right, and rear-left wheel, respectively. Based on a speed of 72 km/h and a distance between the front and rear wheel axles of 1 meter, there will be a time delay of 0.05 s between the applied front and rear wheel force disturbances.



Figure 7.8: ISO 8608 road displacement profile with roughness factor 5 (a) with corresponding actuator force disturbance (b)

7.3 Hardware and Control Co-Design

7.3.1 Objective Function

The objective of the active car suspension setup is to achieve the best driver's comfort. According to the ISO 2631 norm, the driver's comfort is quantified as the perceived acceleration levels in the three principal axes of translation (vertical, longitudinal, and lateral) [247, 248]. The lab setup only allows translation along the vertical Z-axis. That is why in this case, the control objective is to minimize the Z-axis accelerations, maximizing the driver's comfort. A 'fitness value' (FV) quantifies the performance according to the ISO 2631 norm by taking the rms of the acceleration along the Z-axis:

$$FV = \sqrt{\frac{1}{k} \sum_{K=1}^{k} \ddot{z}^2},$$
(7.8)

with k the total number of samples, being $T_m/T_s = 30s/0.001s = 30000$.

7.3.2 Design Parameters

The design parameters for this co-design optimization problem consist of two actuator selection integers, two sensor selection binaries, two control architecture binaries, and 12 controller tuning parameters.

The actuator selection integers $i_{act,f}$ and $i_{act,r}$ define which type of actuators are applied to the front wheels f and the rear wheels r, respectively. Table 7.2 provides an overview of the different actuator types, with each actuator type corresponding to an associated cost and maximum actuator output. The component cost is chosen arbitrarily and will be different for each application. Therefore, this cost is not expressed in a currency but in percentage terms relative to the most expensive setup possible.

Actuator selection integer <i>i</i> _{act,}	Cost [%]	Maximum actuator output [Nm]
0	0	0
1	5.7	1
2	14.3	2
3	42.9	5

Table 7.2: Possible actuator selection integer $i_{act,...}$ values with corresponding cost and maximum actuator output

The design-space for the feedback control architecture is shown in Fig. 7.9. This design-space shows the attainable form of the feedback control when all capabilities regarding the control configuration optimization are active. The second and fourth row of the C and D matrices of the identified model are omitted for the co-design optimization procedure so that the model has the platform position z and acceleration \ddot{z} as outputs. PID controllers can be applied separately to the front and rear wheels for both the displacement and acceleration feedback control. The sensor selection binaries $b_{sen,z}$ and $b_{sen,z}$ define the presence or absence of a sensor on the central platform heave position z and acceleration \ddot{z} , respectively. The cost for a displacement feedback sensor and an acceleration feedback sensor is 1.4% and 4.3%, respectively. The control architecture binaries $b_{CL,z}$ and $b_{CL,\ddot{z}}$ define the presence or absence of the PID controllers on the displacement and acceleration feedback control, respectively. Both control loops have a cost of 2.1%. The PID controller values $K_{p,...}$, $T_{i,...}$, and $T_{d,...}$ are also design parameters, with the subscript referring to the corresponding controller. For example, $T_{d,z,r}$ is the T_d value of the acceleration feedback PID controller connected to the rear actuators.



Figure 7.9: Design-space for the feedback control architecture with the active car suspension model in blue, road profile force disturbance signals in green, actuator output saturation in yellow, PID controllers in red, and sensor feedback in orange

7.3.3 Constraints

The number of possible actuator types determines the upper and lower bounds for the actuator selection integers, see Table 7.3. For the binary design parameters, the bounds are equal to zero and one. For the PID values, the most relaxed boundaries are determined based on system stability margins. For example, for determining the constraint on the upper bound of the acceleration feedback values $K_{p,\vec{z},f}$ and $K_{p,\vec{z},r}$, a phase margin ϕ_{PM} of 50° is suggested to obtain a sufficiently robust control [36]. The Bode plot of the system input u to the platform acceleration \vec{z} identification measurements is shown in Fig. 7.10. To obtain this phase margin ϕ_{PM} of 50°, a zero crossing of the magnitude plot must occur at a phase angle shift of $-180^{\circ} + \phi_{PM} = -180^{\circ} + 50^{\circ} = -130^{\circ}$. The phase plot shows that a phase angle shift of -130° occurs at $\omega_{PM50} = 19.5Hz$ (illustrated with a black vertical line in Fig. 7.10). The gain at phase ω_{PM50} is 25.96 dB. Thus, the corresponding gain factor is $10^{-25.96/20} = 0.05$.



Figure 7.10: Bode plot of the identification measurements (blue) from system inputs u to platform acceleration \ddot{z} . A vertical black line indicates the frequency ω_{PM50} at which the phase plot attains -130° .

Table 7.3 provides an overview of the design parameters with corresponding lower bound (LB) and upper bound (UB) constraints. Next to the design parameter bounds, a non-linear maximum implementation cost constraint is implemented for which the optimization algorithm ensures that the sum of the various components does not exceed this maximum implementation cost. By subsequently performing this optimization for several different maximum implementation costs, the progression of the maximum achievable performance relative to the total implementation cost can be determined.

Design parameter	LB	UB	Design parameter	LB	UB
$i_{act,f}$	0	3	$i_{act,r}$	0	3
$b_{sen,z}$	0	1	$b_{sen,\ddot{z}}$	0	1
$b_{CL,z}$	0	1	$b_{CL,\ddot{z}}$	0	1
$K_{p,z,f}$	0	2	$K_{p,z,r}$	0	2
$T_{i,z,f}$	0	0.01	$T_{i,z,r}$	0	0.01
$T_{d,z,f}$	0	0.01	$T_{d,z,r}$	0	0.01
$K_{p,\ddot{z},f}$	0	0.05	$K_{p,\ddot{z},r}$	0	0.05
$T_{i,\ddot{z},f}$	0	0.01	$T_{i,\ddot{z},r}$	0	0.01
$T_{d,\ddot{z},f}$	0	0.01	$T_{d,\ddot{z},r}$	0	0.01

 Table 7.3: Design parameters with their corresponding lower bound (LB) and upper bound (UB) constraints

7.3.4 Genetic Algorithm Implementation

A Genetic Algorithm (GA) is deployed to optimize the 18 design parameters based on the non-linear objective function and according to the mixed-integer and nonlinear constraints mentioned above. For this active car suspension application, the optimization is performed repeatedly with a changing constraint on the maximum cost varying from 0% to 100% in 14 equal steps. By allowing a higher maximum implementation cost in successive optimizations, more expensive and better performing parts can be used. In this way, the different points of the Pareto front are determined. Table 7.4 depicts some of the key settings for the Genetic Algorithm. The optimization took approximately 8 hours and 18 minutes on an Intel[®] Xeon[®] CPU @ 3.10 GHz with 64 GB of RAM.

 Table 7.4: Settings for the Genetic Algorithm applied to the active car suspension model

Description	Value	Unit
Population size	200	individuals
Maximum calculation time	7200	seconds
Maximum number of generations	infinite	generations
Function tolerance	10^{-10}	/
Maximum stall generations	10	generations
Crossover fraction	40	%
Elite count	20	individuals

7.4 Results and Measurements

7.4.1 Model-Based Pareto Optimizations

Table 7.5 shows the results for every Pareto point optimization, while the resulting Pareto front is graphically depicted in Fig. 7.11. These results provide a valuable understanding of the trade-off between the achievable performance and the implementation cost. As more implementation costs are allowed, actuators with higher torque outputs can be applied, resulting in better fitness values and, thus, better system performances. Each point contains an optimized situation depending on the implementation cost. It will be up to the design engineer to decide which Pareto point is most appropriate for the specific application. However, Pareto point 4 seems to be the most interesting because the results show that only a relatively small improvement in performance is obtained for Pareto points with a higher cost. None of the optimized Pareto points contain a position feedback control. Thus, these results show that applying a position feedback control gives no added value to the intended objective. This is also why the last Pareto point does not have a cost of 100%.



Figure 7.11: Pareto front representing the active car suspension model optimization results, showing the maximum achievable performance related to the implementation cost

Pareto	Cost	Fitness	$[i_{act,f},$	$[b_{sen,\ddot{z}},$	$[b_{CL,\ddot{z}},$	$[K_{p,\ddot{z},f} \dots$	$[K_{p,\ddot{z},r} \dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
point	[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}$]	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$]	$T_{i,z,f} T_{d,z,f}$]	$T_{i,z,r} T_{d,z,r}$]
0	0	85526	[0,0]	[0,0]	[0,0]	0	0	0	0
1	12.7	58318	[0,1]	[1,0]	[1,0]	[0.037 0.005 0.006]	[0.040 0.006 0.009]	0	0
2	18.7	36633	[1,1]	[1,0]	[1,0]	[0.042 0.003 0.008]	[0.048 0.001 0.009]	0	0
3	27.6	26795	[1,2]	[1,0]	[1,0]	[0.043 0.005 0.004]	[0.036 0.007 0.007]	0	0
4	36.6	17800	[2,2]	[1,0]	[1,0]	[0.049 0.009 0.003]	[0.049 0.007 0.008]	0	0
5	66.4	16332	[3,2]	[1,0]	[1,0]	[0.048 0.009 0.009]	[0.043 0.004 0.007]	0	0
6	96.3	14404	[3,3]	[1,0]	[1,0]	[0.05 0 0.01]	[0.05 0.01 0.01]	0	0

 Table 7.5: Model-based Pareto front optimization results (rounded). Inactive controllers are indicated with 'o'.

7.4.2 Validation Measurements

The results of the model-based Pareto optimizations are validated with measurements on the physical lab setup. For each hardware and control combination of Table 7.5, the system response to the same road disturbance signals is measured.

Appendix C details the differences between the model-based calculated responses and the actual measured responses for each Pareto point. Fig. 7.12 depicts the platform acceleration \ddot{z} measurements for every optimized Pareto point configuration. The progression of the graphs clearly shows that the measured accelerations decrease as a higher maximum implementation cost is allowed. Fig. 7.13 depicts the difference in platform accelerations \ddot{z} measurements between no active actuation and the optimized active actuation from Pareto point 6. The figure shows that the largest acceleration spike decreases from $-566mm/s^2$ to $-63mm/s^2$, which is a reduction of 89%. This means that a significant increase in driver's comfort can be achieved using the optimized active car suspension, which is also represented in the corresponding fitness values.



Figure 7.12: Measured platform acceleration \ddot{z} for every optimized Pareto point configuration



Figure 7.13: Comparison of the measured platform acceleration \ddot{z} for no active actuation (blue) and with optimized active actuation (red) from Pareto point 6 for the same road disturbance signals

Fig. 7.14 shows a comparison of the calculated and measured front and rear wheel actuator outputs u_f and u_r and platform acceleration signals \ddot{z} for a part of the time-domain response for Pareto point 6, demonstrating that the calculated and measured responses match well. Table 7.6 shows the fitness values for the calculated model-based optimization results and the corresponding validation measurements. These validation measurement results are depicted together with the model-based Pareto optimization results in Fig. 7.15. The calculated and measured fitness values show better agreement as the implementation cost rises and more features are allowed. Although the individual measured fitness values for configurations with smaller implementation costs do not match the model-based calculated fitness values, the overall trend is similar to that of the calculated fitness values. Therefore, conclusions on the hardware and control co-design based on the model-based optimization values will still be valid on the physical setup. The differences mentioned above can be attributed to modeling errors (e.g., friction or non-linear system behavior) and inherent measurement noise levels in the actual setup rather than limitations of the optimization algorithm.



Figure 7.14: Comparison of the calculated (blue) and measured (red) timedomain responses for the front wheel actuator output u_f (a), rear wheel actuator output u_r (b), and platform acceleration \ddot{z} (c) for the validation measurement of Pareto point 6. The actuator outputs $u_{...}$ consist of the road disturbance signals $d_{...}$ and the control effort signals $f_{...}$.



Figure 7.15: Comparison of the Pareto front results (blue) from the modelbased co-design optimization methodology and the Pareto point validation measurements on the physical lab setup (red)

Pareto	Cost	Model-based optimization	Validation measurement
point	[%]	Fitness value []	Fitness value []
0	0	85526	156349
1	12.7	58318	100677
2	18.7	36633	46322
3	27.6	26795	33926
4	36.6	17800	20448
5	66.4	16332	17489
6	96.3	14404	15106

Table 7.6: Fitness values for the model-based optimization results and the corresponding validation measurements

7.4.3 Comparison to Existing Controller Tuning Methods

The optimized controller values are compared to existing controller tuning methods to validate the optimized acceleration controller tuning parameters as part of the control configuration optimization. First, the obtained controller tuning is compared to an LQR feedback design. In an LQR control design, relative weights are given to the state and input variables by adjusting the diagonal matrix Q and R values, respectively. Based on these values, a cost function minimization routine determines the values of the feedback matrices [56]. Thus, the LQR controller design comes down to adjusting the weights of the Q and R matrices according to the desired system behavior. However, for this application, typical starting values of 1, applying the Bryson method [60], or manual adjustments of the Q and R diagonal matrices did not lead to a stable, feasible solution.

Furthermore, an existing controller tuning method is applied in which the P and PI controller values are determined based on the intended phase margins of the measured Bode plots [249]. This measured model is further denoted as G. The controller tuning method was applied to obtain P acceleration controllers with intended phase margins ϕ_{PM} equal to 30°, 60°, and 90°. Fig. 7.16 shows the Bode plot of the identification measurements for G in which the frequencies $\omega_{P,PM30}$, $\omega_{P,PM60}$, and $\omega_{P,PM90}$ for phase margins of 30°, 60°, and 90° are shown with black vertical lines. The corresponding $K_{p,P,PM30}$, $K_{p,P,PM60}$, and $K_{p,P,PM90}$ values are obtained by compensating for the difference in the magnitude ratio at $\omega_{P,PM30}$, $\omega_{P,PM60}$, and $\omega_{P,PM90}$, respectively. After that, each P controller was tested on the physical lab setup with different actuator types for a comparison with the obtained Pareto points from the co-design optimization methodology. Table 7.7 shows the results for the existing P controller tuning method and the corresponding validation measurements.



Figure 7.16: Bode plot of the identification measurements from system inputs u to platform acceleration \ddot{z} with frequencies ω_{PM30} , ω_{PM60} , and ω_{PM90} shown in vertical black lines to define the corresponding P controller values

The same approach is used to obtain PI acceleration controllers with phase margins ϕ_{PM} of 30°, 60°, and 90° at corresponding frequencies $\omega_{PI,PM30}$, $\omega_{PI,PM60}$, and $\omega_{PI,PM90}$. A PI controller consists of a gain, a pole at the origin, and a zero [52]. The pole will cause a phase delay of -90° at the frequency $\omega_{PI,PM}$ at which the magnitude plot $|G(j\omega_{PI,PM})|$ crosses 0 dB. A zero causes a phase angle shift of +90° for infinitely high frequencies, but it is more realistic to design the controller requiring a phase shift of +60° at frequency $\omega_{PI,PM}$. With this knowledge, the corresponding frequency $\omega_{PI,PM}$ can be determined as:

$$\angle G(j\omega_{PI,PM}) - 90^{\circ} + 60^{\circ} = -180^{\circ} + \phi_{PM}.$$
(7.9)

7.4 Results and Measurements

Fig. 7.17 shows the Bode plot of the identification measurements in which the frequencies $\omega_{PI,PM30}$, $\omega_{PI,PM60}$, and $\omega_{PI,PM90}$ for phase margins ϕ_{PM} of 30°, 60°, and 90° are shown with black vertical lines.



Figure 7.17: Bode plot of the identification measurements from system inputs u to platform acceleration \ddot{z} with frequencies ω_{PM30} , ω_{PM60} , and ω_{PM90} shown in vertical black lines to define the corresponding PI controller values

Once frequency $\omega_{PI,PM}$ is defined, $T_{i,PI,PM}$ can be specified as:

$$\angle (1+T_{i,PI,PM}s)|_{\omega=\omega_{PI,PM}} = 60^{\circ}, \qquad (7.10)$$

from which the value of $T_{i,PI,PM}$ is found.

Finally, the value $K_{p,PI,PM}$ is determined so that the combined magnitude plot of the system and the PI controller intersects 0 dB at frequency $\omega_{PI,PM}$. The obtained PI acceleration controllers were also tested on the physical lab setup with different actuator types. Table 7.8 shows the results for the existing PI controller tuning method and the corresponding validation measurements.

Phase Margin (ϕ_{PM})	30°			60°			90°		
$\angle G(j\omega_{P,PM})$		-150°		-120°			-90°		
$\omega_{P,PM} [Hz]$	38.28			15.8			8.76		
$ G(j\omega_{P,PM}) [dB]$	15.2			28.38			35.16		
$K_{p,P,PM}$	$10^{-15.2/20} = 0.17$		$10^{-28.38/20} = 0.04$			$10^{-35.16/20} = 0.02$			
$[i_{act,f}, i_{act,r}]$	[1,1]	[2, 2]	[3,3]	[1, 1]	[2, 2]	[3, 3]	[1, 1]	[2,2]	[3, 3]
Cost [%]	18.7 36.6 96.3		18.7	36.6	96.3	18.7	36.6	96.3	
Fitness value (FV)	1	/	1	56956	32431	27017	60260	48807	47749

Table 7.7: Existing P controller tuning and validation measurement results for different phase margins. The Fitness values for unstable and unfeasible system configurations are denoted by '/'.

Table 7.8: Existing PI controller tuning and validation measurement results for different phase margins

Phase Margin (ϕ_{PM})	30°			60°			90°			
$\angle G(j\omega_{PI,PM})$		-120)°		-90°			-60°		
$\omega_{PI,PM} \left[Hz \right]$		15.8	3	8.76			4.91			
$ G(j\omega_{PI,PM}) [dB]$	28.38			35.16			40.61			
$T_{i,PI,PM}$	$tan(60^{\circ})/(15.8 \cdot 2\pi) = 0.017$			$tan(60^{\circ})/(8.76 \cdot 2\pi) = 0.032$			$tan(60^{\circ})/(4.91 \cdot 2\pi) = 0.056$			
$K_{p,PI,PM}$	$10^{-30.6/20} = 0.030$		$10^{-39.17/20} = 0.011$			$10^{-45.35/20} = 0.054$				
$[i_{act,f}, i_{act,r}]$	[1,1] $[2,2]$ $[3,3]$		[1, 1]	[2, 2]	[3,3]	[1, 1]	[2, 2]	[3, 3]		
Cost [%]	18.7 36.6 96.3		18.7	36.6	96.3	18.7	36.6	96.3		
Fitness value (FV)	57396	35146	30643	61144	51611	51049	64982	59839	59761	
7.4 Results and Measurements

The measured performances of these P and PI controller tuning methods are depicted as black circles in Fig. 7.18. As shown in this figure, no classical controller tuning method achieves the performance obtained with the optimized values determined by the co-design methodology proposed in this PhD dissertation.



Figure 7.18: Comparison of the Pareto front results (blue) from the modelbased co-design optimization methodology, Pareto point validation measurements on the physical lab setup (red), and measurements of existing controller tuning methods (black)

7.5 Chapter Conclusions

In this chapter, the presented hardware architecture and control configuration codesign optimization methodology is applied to a downscaled active car suspension lab setup. First, an introduction to passive and (semi) active car suspensions is given, and an explanation is provided of a lab setup that can simulate the behavior of an active car suspension. Next, the active suspension setup dynamics were identified using a Prediction Error Minimization method to obtain a state-space system representation. Actuator force disturbance signals were determined to mimic a situation in which a car is driving over an actual road profile. The active components in the car suspension can be used to reduce the central platform vibrations so that the driver's comfort is increased.

With the proposed co-design methodology, the optimal type and location of actuators and sensors are determined simultaneously with the optimal control architecture and controller tuning parameters, exhibiting a profound level of co-design that has not been addressed in the current literature. This novel co-design methodology uses a Genetic Algorithm implementation, of which the results are presented in a Pareto front that graphically shows the interplay between the maximum achievable performance and the implementation cost. The results also show that using a position feedback loop has no positive effect on improving the driver's comfort. The optimization results can be of great value to a design engineer in understanding how the maximum achievable performance varies as a function of the implementation cost.

The system configurations of the resulting Pareto points are validated on the physical lab setup. These validation measurements show that the obtained modelbased trend in the Pareto front can also be observed in the corresponding measurements. The differences in the obtained performances between the calculated and the measured situations can be attributed to the inherent measurement noise levels and modeling inaccuracies (e.g., friction or non-linear system behavior) rather than to errors in the co-design optimization methodology. Additionally, existing controller tuning methods were applied to the physical setup, with no existing method achieving the performance obtained with the proposed co-design methodology. This demonstrates that the presented co-design methodology is capable of determining the optimal controller tuning parameters.

Chapter 8

General Conclusions and Future Work

8.1 General Conclusions

For complex systems, it is often difficult to estimate in advance what the exact influence is on the system performance of, for example, different types of actuators and sensors, their placement within the system, which control structure should be applied, or how the controller tuning parameters should be configured. In this work, a novel optimization workflow is presented to perform the optimal co-design of the hardware architecture and control configuration for subsystems consisting of multiple interacting subsystems. With the proposed co-design methodology, an optimization workflow is established to determine the optimal type and location of actuators and sensors simultaneously with the optimal control architecture and controller tuning parameters.

An extensive literature review shows that the most comprehensive form of codesign in current literature only considers the optimization of the hardware architecture and controller tuning parameters. The ability to consider the control architecture optimization and different hardware types is a substantial addition to the current state-of-the-art in the hardware and control co-design. The proposed methodology is applicable to a very broad range of systems. These include, for example, mechatronic, electrical, thermal, or chemical system classes.

The co-design problem is converted to a mathematically optimizable formulation, with the design parameters being the actuator selection integers \mathbb{Z}_{act} , sensor selection integers \mathbb{Z}_{sen} , control architecture binaries \mathbb{B}_{Υ} , and the controller tuning parameters \mathbb{R}_{tuning} . This co-design problem results in a discontinuous, mixedinteger optimization problem with non-linear objective functions and constraints. Multiple objectives can be taken into account simultaneously, such as reference tracking, vibration control, settling times, energy consumption, etc. The applicable constraints may include but are not limited to the total implementation cost, maxi-

mum actuator effort, or mechanical motion tolerances. Due to the properties of this optimization problem, only a few optimization algorithms are suitable, of which a Genetic Algorithm (GA) is selected as the most appropriate one. A Genetic Algorithm is a derivative-free evolutionary optimization algorithm that requires a relatively large number of iterations, making it a computationally expensive process. An additional disadvantage of using a Genetic Algorithm is its non-deterministic character, meaning that achieving a global minimum cannot be fully guaranteed. Although not guaranteed, an analysis of the objective function surface plots shows that for the applications addressed in this work, a global minimum can be found using a Genetic Algorithm with appropriate settings, as discussed in this work. The Genetic Algorithm also proves to be very flexible in applying non-linear objective and constraint function evaluations. It is also important to emphasize that the proposed co-design methodology is not solely dependent on using a Genetic Algorithm. This means that in the future, the co-design workflow can also be applied with other novel optimization algorithms that can handle the same type of problems.

The efficiency of the proposed co-design methodology is improved by applying two developed enhancements. First, an open-loop analysis is applied to determine infeasible hardware combinations in advance, and this information can be taken into account while performing the actual optimization. It is shown that a considerable time gain can be obtained when the open-loop analysis is applied. Second, this work also discusses a method for establishing a state-space system of the closed-loop system including the entire feedback control loop. This closed-loop state-space methodology allows for a much faster calculation of the closed-loop system response to external reference and disturbance signals. These two extensions provide a significant reduction in the total computation time of the overall codesign methodology. The outcome of this co-design methodology is represented in a Pareto front, graphically depicting the trade-off between the conflicting objectives, being the maximum achievable performance and the total implementation cost. The Pareto front results provide valuable insights for the design engineer into the system design.

The cases that have been elaborated in this work illustrate the flexibility and effectiveness of the proposed co-design methodology. The hardware and control co-design is performed on an illustrative example of a mechanical synchronization model, a composite plate application, and an active car suspension setup. The composite plate application demonstrates that in addition to linear time-invariant (LTI) state-space systems, also linear parameter-varying (LPV) and linear time-varying (LTV) systems can be optimized. In addition, this case highlights that external toolboxes can be used in the non-linear objective and constraint functions. The co-design optimization results on the active car suspension model are validated on a physical lab setup, showing that the results are also observed in measurements. The results of the presented co-design methodology are also compared with existing controller tuning methods. This shows that none of these existing methods lead to

a better performance than the presented co-design methodology for the cases used in this PhD. Additionally, the optimization results are validated by constructing a surface plot of the objective function and checking that none of the surrounding fitness values show a better performance than the optimization result. In this way, it is validated that with proper settings, the Genetic algorithm is capable of attaining the global minimum.

From these results, it can be concluded that using the presented methodology, a design engineer is able to achieve the optimal system performance for a specific cost through the ideal selection and design of the actuators, sensors, control architectures, and controller tuning parameters. It is clear that this can significantly help the design engineer to get a better understanding of the trade-off between optimal achievable performance and implementation cost. For larger applications with more design possibilities, it is impossible to determine the optimal combination of design parameters without using optimization algorithms. For such applications, the proposed methodology provides a solution to obtain the optimal hardware and control co-design. Additionally, for systems that are already operational, this tool can be a great added value to evaluate the potential for performance improvements related to the optimization of the controller architecture and controller tuning parameters.

8.2 **Recommendations for Future Work**

It is possible to develop further enhancements to the current methodology:

- For instance, it should be possible to show that the proposed co-design methodology is also applicable for other control strategies. In this work, Proportional-Integral-Derivative (PID), linear-quadratic regulator (LQR), and robust H_{∞} control are discussed, but other strategies, such as model predictive control (MPC), fuzzy control, fractional order control, etc. could be used. Additionally, the proposed methodology should also be able to simultaneously consider different control strategies, it would be possible to make the right trade-off based on the cost and the achievable performance of these different control strategies.
- In this work, the dynamic behavior of the actuators and sensors is neglected. Although the influence on the system dynamics of the presence or absence of actuators and sensors and non-linear actuator saturation is taken into account, it is assumed that an actuator can instantaneously achieve the desired output and that a sensor instantaneously measures the corresponding properties. Extensions could be provided so that actuators and sensors exhibit a particular dynamic input-output behavior. In turn, this behavior can be described as a state-space system to be included in the closed-loop control

system. As a result, the number of states in the closed-loop state-space system will increase, similar to the integrated PID controllers that also describe a dynamic input-output relationship. By doing so, the optimization methodology could take into account different dynamic behavior for different types of actuators and sensors. An extension could also be provided to incorporate continuous, hardware-specific design parameters. An extension could also be provided to incorporate continuous, hardware-specific design parameters as opposed to only discrete hardware types.

- The open-loop analysis only exploits the observability and controllability properties of the system to exclude infeasible hardware combinations. This open-loop analysis can potentially be extended to include related conditions such as stabilizability and detectability.
- The cost of different hardware types can be determined reasonably straightforward, but determining the cost of specific control architecture features is less obvious. Therefore, tools could be developed to support the design engineer in defining the cost of the individual control loops and strategies, possibly incorporating the implementation effort or computational burden.
- Further research can then be done to achieve a more deterministic optimization. However, this does not seem evident since the stochastic nature is inherent in using a Genetic Algorithm as an optimization algorithm.
- A final suggestion for future work is to model uncertainty. By incorporating an uncertainty analysis, it might be possible to obtain a resulting Pareto front displaying specific limits of uncertainty based on variables that are undecided within a specific range. The impact of integrating these model uncertainties on the required computation time is yet unknown.

Appendix A Resulting SS_{CL} Algorithms

This appendix shows the algorithms that depict the intermediate solutions of applying the closed-loop state-space SS_{CL} methodology, seen in Section 5.3.

Algorithm 9 Iterative calculation as in algorithm 5, but with combined PID controllers. The sparse matrices $S_{pre,...}$ remain constant throughout the iterative response calculations and can be predetermined to simplify and accelerate the calculations.

Algorithm 10 (Continued from algorithm 9)

	Combined equations on synchronizing controllers
7:	$PID_{sc,u}(:,k) = [PID_{sc,x}(1,k); PID_{sc,x}(3,k)] + \dots$
	$[c_{b,sc,0}, 0; 0, c_{a,sc,0}] * [e_{b,sc}(k); e_{a,sc}(k)]$
8:	$PID_{sc,x}(:, k+1) = S_{pre,1} * PID_{sc,x}(:, k) + \dots$
	$[c_{b,sc,1}, 0; c_{b,sc,2}, 0; c_{a,sc,1}, 0; c_{a,sc,2}, 0] * [e_{b,sc}(k); e_{a,sc}(k)] + \dots$
	$[1,0;0,0;0,1;0,0] * PID_{sc,u}(:,k)$
	Cascade level <i>b</i> controller inputs
9:	$e_{b,1}(k) = Ref_1(k) - y(1,k) - PID_{sc,u}(2,k)$
10:	$e_{b,2}(k) = Ref_2(k) - y(2,k) + PID_{sc,u}(2,k)$
	Cascade level b decentralized and distributed combined controller equations
11:	$PID_{b,u}(:,k) = [PID_{b,x}(1,k); PID_{b,x}(3,k); PID_{b,x}(5,k); \dots$
	$PID_{b,x}(7,k)] + [c_{b,1,0}, 0; 0, c_{b,2,0}; c_{b,1-2,0}, 0; 0, c_{b,2-1,0}] * [e_{b,1}(k); *e_{b,2}(k)]$
12:	$PID_{b,x}(:, k+1) = S_{pre,2} * PID_{b,x}(:, k) + \dots$
	$[c_{b,1,1}, 0; c_{b,1,2}, 0; 0, c_{b,2,1}; 0, c_{b,2,2}; c_{b,1-2,1}, 0; c_{b,1-2,2}, 0; 0, c_{b,2-1,1}; \dots$
	$0, c_{b,2-1,2}] * [e_{b,1}(k); *e_{b,2}(k)] + S_{pre,3} * PID_{b,u}(:,k)$
	Cascade level a controller inputs
13:	$e_{a,1}(k) = PID_{b,u}(1,k) + PID_{b,u}(3;k) + \dots$
	$K_{ff,b,1} * (Ref_1(k) - Ref_1(k-1)) / T_s - \hat{x}(1,k) - PID_{sc,u}(1,k)$
14:	$e_{a,2}(k) = PID_{b,u}(2,k) + PID_{b,u}(4;k) + \dots$
	$K_{ff,b,2} * \left(Ref_2(k) - Ref_2(k-1) \right) / T_s - \hat{x}(2,k) + PID_{sc,u}(1,k)$
	Cascade level <i>a</i> decentralized and distributed combined controller equations
15:	$PID_{a,u}(:,k) = [PID_{a,x}(1,k); PID_{a,x}(3,k); PID_{a,x}(5,k); \dots$
	$PID_{a,x}(7,k)] + [c_{a,1,0},0;0,c_{a,2,0};c_{a,1-2,0},0;0,c_{a,2-1,0}] * [e_{a,1}(k);e_{a,2}(k)]$
16:	$PID_{a,x}(:, k+1) = S_{pre,2} * PID_{a,x}(:, k) + \dots$
	$[c_{a,1,1}, 0; c_{a,1,2}, 0; 0, c_{a,2,1}; 0, c_{a,2,2}; c_{a,1-2,1}, 0; c_{a,1-2,2}, 0; 0, c_{a,22,1}; \dots$
	$0, c_{a,2-1,2}] * [e_{a,1}(k); e_{a,2}(k)] + S_{pre,3} * PID_{a,u}(:,k)$
	Process inputs
17:	$u(:,k) = [PID_{a,u}(1,k) + PID_{a,u}(4,k); \dots$
	$PID_{a,u}(2,k) + PID_{a,u}(3,k); Dist(k)]$
	Process and observer equations
18:	$x(:, k+1) = A_{pro} * x(:, k) + B_{pro} * u(:, k)$
19:	$y(:, k+1) = C_{pro} * x(:, k) + D_{pro} * u(:, k)$
20:	$\hat{x}(:,k+1) = A_{obs} * \hat{x}(:,k) + B_{obs} * u(:,k) + L_{obs} * (y(:,k) - \hat{y}(:,k))$
21:	$\hat{y}(:, k+1) = C_{obs} * \hat{x}(:, k) + D_{obs} * u(:, k)$
22:	end

Algorithm 11 Integrating intermediate solutions into the successive equations. More specifically for this step in the methodology, integrating the controller input equations directly into the controller equations.

Algorithm 12 (Continued from Algorithm 11)

	Process inputs
11:	$u(:,k) = [PID_{a,u}(1,k) + PID_{a,u}(4,k); \dots$
	$PID_{a,u}(2,k) + PID_{a,u}(3,k); Dist(k)]$
	Combined process and observer equations
12:	$x(:, k+1) = [A_{pro}, zeros(6, 6); zeros(6, 6), A_{obs}] * x(:, k) + \dots$
	$[B_{pro}; B_{obs}] * u(:, k) + [zeros(6, 3); L_{obs}] * (y(1:3, k) - y(4:6, k))$
13:	$y(:, k+1) = [C_{pro}, zeros(3, 6); zeros(3, 6), C_{obs}] * x(:, k) + \dots$
	$[D_{pro}; D_{obs}] * u(:, k)$
14:	end

Algorithm 13 Integrating intermediate output calculations into the successive equations. More specifically, integrating the controller output equations directly into the controller state equations. The sparse matrices S_{\dots} are too extensive to be shown clearly, but they follow from the previous steps in the described methodology. Instead, their dimensions are given Table A.1.

Iterative response calculation 1: for $k = 1 : T_m/T_s$ Controller equations $PID_{sc,x}(:, k+1) = S_{pre,1} * x(:, k) + S_{pre,2} * y(:, k) + S_{pre,3} * PID_{sc,x}(k)$ 2: $PID_{b,x}(:, k+1) = S_{pre,4} * x(:, k) + S_{pre,5} * y(:, k) + \dots$ 3: $S_{pre,6} * PID_{b,x}(k) + S_{pre,7} * PID_{sc,x}(k) + \dots$ $S_{pre,8} * Ref_1(k) + S_{pre,9} * Ref_2(k)$ $PID_{a,x}(:, k+1) = S_{pre,10} * x(:, k) + S_{pre,11} * y(:, k) + \dots$ 4: $S_{pre,12} * PID_{a,x}(k) + S_{pre,13} * PID_{b,x}(k) + \dots$ $S_{pre,14} * PID_{sc,x}(k) + S_{pre,15} * (Ref_1(k) - Ref_1(k-1)) / T_s + \dots$ $\hat{S}_{pre,16} * (Ref_2(k) - Ref_2(k-1)) / T_s + \dots$ $\hat{S}_{pre,17} * Ref_1(k) + S_{pre,18} * Ref_2(k)$ Process and observer equations $x(:, k+1) = S_{pre,19} * x(:, k) + S_{pre,20} * y(:, k) + \dots$ 5: $S_{pre.21} * PID_{a.x}(k) + S_{pre.22} * PID_{b.x}(k) + S_{pre.23} * PID_{sc,x}(k) + \dots$ $S_{pre.24} * Dist(k) + S_{pre.26} * (Ref_1(k) - Ref_1(k-1)) / T_s + \dots$ $S_{pre,27} * (Ref_2(k) - Ref_2(k-1)) / T_s + S_{pre,28} * Ref_1(k) + ...$ $S_{pre,29} * Ref_2(k) + [zeros(6,3); L_{obs}] * (y(1:3,k) - y(4:6,k))$ $y(:, k+1) = S_{pre,30} * x(:, k) + S_{pre,31} * y(:, k) + \dots$ 6: $S_{pre,32} * PID_{a,x}(k) + S_{pre,33} * PID_{b,x}(k) + S_{pre,34} * PID_{sc,x}(k) + \dots$ $S_{pre,35} * Dist(k) + S_{pre,36} * (Ref_1(k) - Ref_1(k-1)) / T_s + \dots$ $S_{pre,37} * (Ref_2(k) - Ref_2(k-1)) / T_s + \dots$ $S_{pre,38} * Ref_1(k) + S_{pre,39} * Ref_2(k)$ 7: end

Matrix	Dimensions	Matrix	Dimensions	Matrix	Dimensions
$S_{pre,1}$	$[4 \times 12]$	$S_{pre,2}$	$[4 \times 6]$	$S_{pre,3}$	$[4 \times 4]$
$S_{pre,4}$	$[8 \times 12]$	$S_{pre,5}$	$[8 \times 6]$	$S_{pre,6}$	$[8 \times 8]$
$S_{pre,7}$	$[8 \times 4]$	$S_{pre,8}$	$[8 \times 1]$	$S_{pre,9}$	$[8 \times 1]$
$S_{pre,10}$	$[8 \times 12]$	$S_{pre,11}$	$[8 \times 6]$	$S_{pre,12}$	$[8 \times 8]$
$S_{pre,13}$	$[8 \times 8]$	$S_{pre,14}$	$[8 \times 4]$	$S_{pre,15}$	$[8 \times 1]$
$S_{pre,16}$	$[8 \times 1]$	$S_{pre,17}$	$[8 \times 1]$	$S_{pre,18}$	$[8 \times 1]$
$S_{pre,19}$	$[12 \times 12]$	$S_{pre,20}$	$[12 \times 6]$	$S_{pre,21}$	$[12 \times 8]$
$S_{pre,22}$	$[12 \times 8]$	$S_{pre,23}$	$[12 \times 4]$	$S_{pre,24}$	$[12 \times 1]$
$S_{pre,25}$	$[12 \times 1]$	$S_{pre,26}$	$[12 \times 1]$	$S_{pre,27}$	$[12 \times 1]$
$S_{pre,28}$	$[12 \times 1]$	$S_{pre,29}$	$[6 \times 12]$	$S_{pre,30}$	$[6 \times 6]$
$S_{pre,31}$	$[6 \times 8]$	$S_{pre,32}$	$[6 \times 8]$	$S_{pre,33}$	$[6 \times 4]$
$S_{pre,34}$	$[6 \times 1]$	$S_{pre,35}$	$[6 \times 1]$	$S_{pre,36}$	$[6 \times 1]$
$S_{pre,37}$	$[6 \times 1]$	$S_{pre,38}$	$[6 \times 1]$		

Table A.1: Dimensions for the predetermined sparse matrices $S_{pre,...}$ used
in Algorithm 13

Appendix B

Mechanical Synchronization Case Optimization Responses

This appendix shows the time-domain responses for all Pareto point optimization results of the mechanical synchronization case (see 6.3.5 on page 137). The responses consist of the inertia displacement responses and the corresponding actuator output effort.

Pareto point 1 Results

 Table B.1: Results for Pareto point 1 (rounded) for the mechanical synchronization case

			Har	dware sele	ction integers and	Controller tuning
			col	ntrol archi	tecture binaries	parameters
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}]$	$i_{sen,2}$]	$b_{b,dis} b_{a,sc} b_{b,sc} b_{b,ff}$	\mathbf{R}_{tuning}
1	0	$9e^{8}$	[0 0]	[0 0]	[0 0 0 0 0 0 0]	see Table 6.5



Figure B.1: Time-domain displacement response for Pareto point 1



Figure B.2: Time-domain actuator effort response for Pareto point 1

Pareto point 2 Results

 Table B.2: Results for Pareto point 2 (rounded) for the mechanical synchronization case

			Har	dware sele	Controller tuning	
			co	ntrol archi	parameters	
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$		
point	[%]	value []	$i_{act,2}$]	$i_{sen,2}]$	\mathbf{R}_{tuning}	
2	8.8	$3e^8$	[1 0]	[1 0]	[0 0 1 0 0 0 0]	see Table 6.5



Figure B.3: Time-domain displacement response for Pareto point 2



Figure B.4: Time-domain actuator effort response for Pareto point 2

Pareto point 3 Results

 Table B.3: Results for Pareto point 3 (rounded) for the mechanical synchronization case

			Har	dware sele	ction integers and	Controller tuning
			col	ntrol archi	tecture binaries	parameters
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}]$	$i_{sen,2}$]	$b_{b,dis} b_{a,sc} b_{b,sc} b_{b,ff}$	\mathbf{R}_{tuning}
3	20	$2e^7$	[2 0]	[1 0]	[0 0 0 0 0 0 0]	see Table 6.5



Figure B.5: Time-domain displacement response for Pareto point 3



Figure B.6: Time-domain actuator effort response for Pareto point 3

Pareto point 4 Results

 Table B.4: Results for Pareto point 4 (rounded) for the mechanical synchronization case

			Haro	dware sele ntrol archi	ction integers and tecture binaries	Controller tuning parameters
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}$]	$i_{sen,2}]$	$b_{b,dis} \ b_{a,sc} \ b_{b,sc} \ b_{b,ff}]$	\mathbf{R}_{tuning}
4	25.3	$5e^5$	[1 2]	[0 1]	[0 0 0 0 0 0 0]	see Table 6.5



Figure B.7: Time-domain displacement response for Pareto point 4



Figure B.8: Time-domain actuator effort response for Pareto point 4

Pareto point 5 Results

 Table B.5: Results for Pareto point 5 (rounded) for the mechanical synchronization case

			Har	dware sele	Controller tuning	
			col	ntrol archi	tecture binaries	parameters
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}]$	$i_{sen,2}]$	$b_{b,dis} \ b_{a,sc} \ b_{b,sc} \ b_{b,ff}$]	\mathbf{R}_{tuning}
5	26.6	76.40	[1 2]	[0 1]	[0 0 1 0 0 0 0]	see Table 6.5



Figure B.9: Time-domain displacement response for Pareto point 5



Figure B.10: Time-domain actuator effort response for Pareto point 5

Pareto point 6 Results

 Table B.6: Results for Pareto point 6 (rounded) for the mechanical synchronization case

			Haro	dware sele ntrol archi	Controller tuning parameters	
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$		
point	[%]	value []	$i_{act,2}$]	$i_{sen,2}]$	$b_{b,dis} \ b_{a,sc} \ b_{b,sc} \ b_{b,ff}]$	\mathbf{R}_{tuning}
6	28.8	65.41	[1 2]	[0 1]	[0 0 1 1 0 0 0]	see Table 6.5



Figure B.11: Time-domain displacement response for Pareto point 6



Figure B.12: Time-domain actuator effort response for Pareto point 6

Pareto point 7 Results

 Table B.7: Results for Pareto point 7 (rounded) for the mechanical synchronization case

			Har	dware sele	ction integers and	Controller tuning
			col	ntrol archi	tecture binaries	parameters
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}]$	$i_{sen,2}$]	$b_{b,dis} b_{a,sc} b_{b,sc} b_{b,ff}$	\mathbf{R}_{tuning}
7	39.7	45.37	[2 2]	[0 1]	[1 1 0 0 0 0 0]	see Table 6.5



Figure B.13: Time-domain displacement response for Pareto point 7



Figure B.14: Time-domain actuator effort response for Pareto point 7

Pareto point 8 Results

 Table B.8: Results for Pareto point 8 (rounded) for the mechanical synchronization case

			Haro	dware sele ntrol archi	Controller tuning parameters	
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}$]	$i_{sen,2}]$	$b_{b,dis} \ b_{a,sc} \ b_{b,sc} \ b_{b,ff}]$	\mathbf{R}_{tuning}
8	43.2	29.05	[2 2]	[0 1]	[1 1 1 1 0 0 0]	see Table 6.5



Figure B.15: Time-domain displacement response for Pareto point 8



Figure B.16: Time-domain actuator effort response for Pareto point 8

Pareto point 9 Results

 Table B.9: Results for Pareto point 9 (rounded) for the mechanical synchronization case

			Har	dware sele	Controller tuning	
			col	ntrol archi	tecture binaries	parameters
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}]$	$i_{sen,2}]$	$b_{b,dis} b_{a,sc} b_{b,sc} b_{b,ff}$	\mathbf{R}_{tuning}
9	59.7	28.42	[2 2]	[0 1]	[1 1 1 1 1 1 0]	see Table 6.5



Figure B.17: Time-domain displacement response for Pareto point 9



Figure B.18: Time-domain actuator effort response for Pareto point 9

Pareto point 10 Results

 Table B.10: Results for Pareto point 10 (rounded) for the mechanical synchronization case

			Har	dware seleo ntrol archi	Controller tuning parameters	
Pareto point	Cost [%]	Fitness value []	$\begin{bmatrix} i_{act,1}\\ i_{act,2} \end{bmatrix}$	$\begin{bmatrix} i_{sen,1}\\ i_{sen,2} \end{bmatrix}$	R _{tuning}	
10	49.9	26.73	[3 3]	[0 1]	[1 1 1 1 0 0 0]	see Table 6.5



Figure B.19: Time-domain displacement response for Pareto point 10



Figure B.20: Time-domain actuator effort response for Pareto point 10

Pareto point 11 Results

 Table B.11: Results for Pareto point 11 (rounded) for the mechanical synchronization case

			Hare	dware sele	ction integers and	Controller tuning		
			col	ntrol archi	parameters			
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$			
point	[%]	value []	$i_{act,2}]$	$i_{sen,2}]$	$b_{b,dis} b_{a,sc} b_{b,sc} b_{b,ff}$]	\mathbf{R}_{tuning}		
11	86.4	26.56	[3 3]	[0 1]	[1 1 1 1 1 1 0]	see Table 6.5		



Figure B.21: Time-domain displacement response for Pareto point 11



Figure B.22: Time-domain actuator effort response for Pareto point 11

Pareto point 12 Results

 Table B.12: Results for Pareto point 12 (rounded) for the mechanical synchronization case

			Haro	dware sele ntrol archi	Controller tuning parameters	
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}$]	$i_{sen,2}]$	$b_{b,dis} \ b_{a,sc} \ b_{b,sc} \ b_{b,ff}]$	\mathbf{R}_{tuning}
12	90.7	26.33	[3 3]	[0 1]	[111111]	see Table 6.5



Figure B.23: Time-domain displacement response for Pareto point 12



Figure B.24: Time-domain actuator effort response for Pareto point 12

Pareto point 13 Results

 Table B.13: Results for Pareto point 13 (rounded) for the mechanical synchronization case

			Har	dware sele	ction integers and	Controller tuning		
			col	ntrol archi	parameters			
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$			
point	[%]	value []	$i_{act,2}]$	$i_{sen,2}]$	$b_{b,dis} \ b_{a,sc} \ b_{b,sc} \ b_{b,ff}$]	\mathbf{R}_{tuning}		
13	97.3	26.30	[3 3]	[1 1]	[111111]	see Table 6.5		



Figure B.25: Time-domain displacement response for Pareto point 13



Figure B.26: Time-domain actuator effort response for Pareto point 13

Pareto point 14 Results

 Table B.14: Results for Pareto point 14 (rounded) for the mechanical synchronization case

			Haro	dware sele ntrol archi	Controller tuning parameters	
Pareto	Cost	Fitness	$[i_{act,1}$	$[i_{sen,1}$	$[b_{a,dec} \ b_{a,dis} \ b_{b,dec}$	
point	[%]	value []	$i_{act,2}$]	$i_{sen,2}]$	$b_{b,dis} \ b_{a,sc} \ b_{b,sc} \ b_{b,ff}]$	\mathbf{R}_{tuning}
14	100	24.72	[3 3]	[1 2]	[111111]	see Table 6.5



Figure B.27: Time-domain displacement response for Pareto point 14



Figure B.28: Time-domain actuator effort response for Pareto point 14

Appendix C

Active Car Suspension Setup Validation Measurements

Chapter 7 details how the hardware architecture and control configuration optimization is performed on the model of an active car suspension setup. This appendix shows the validation measurements of each model-based result of the Pareto optimizations applied to the physical setup.

Pareto Point 0 = no active car suspension

Pareto	Cost	Fitness	$[i_{act,f} \dots$	$[b_{sen,\ddot{z}} \dots$	$[b_{CL,\ddot{z}} \dots$	$[K_{p,\ddot{z},f} \dots$	$[K_{p,\ddot{z},r} \dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
point	[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}$	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$	$T_{i,z,f} T_{d,z,f}$	$T_{i,z,r} T_{d,z,r}$
0	0	85526	[0 0]	[0 0]	[0 0]	0	0	0	0

Table C.1: Pareto front optimization results for Pareto point 0 (rounded). Inactive controllers are indicated with 'o'.



Figure C.1: Overview (left) and zoom (right) of the front and rear wheel actuator outputs u_f and u_r , and platform acceleration \ddot{z} for the calculated model-based response (red) and the response measured on the setup (blue) for Pareto point 0

Table C.2: Pareto front optimization results for Pareto point 1 (rounded). Inactive controllers are indicated with 'o'.

Pareto	Cost	Fitness	$[i_{act,f} \dots$	$[b_{sen,\ddot{z}} \dots$	$[b_{CL,\ddot{z}} \dots$	$[K_{p,\ddot{z},f}\dots$	$[K_{p,\ddot{z},r} \dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
point	[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}$]	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$]	$T_{i,z,f} T_{d,z,f}$	$T_{i,z,r} T_{d,z,r}$]
1	12.7	58318	[0 1]	[1 0]	[1 0]	[0.037 0.005 0.006]	[0.040 0.006 0.009]	0	0



Figure C.2: Overview (left) and zoom (right) of the front and rear wheel actuator outputs u_f and u_r , and platform acceleration \ddot{z} for the calculated model-based response (red) and the response measured on the setup (blue) for Pareto point 1

Table C.3: Pareto front optimization results for Pareto point 2 (rounded). Inactive controllers are indicated with 'o'.

Pareto	Cost	Fitness	$[i_{act,f} \dots$	$[b_{sen,\ddot{z}} \dots$	$[b_{CL,\ddot{z}} \dots$	$[K_{p,\ddot{z},f}\dots$	$[K_{p,\ddot{z},r} \dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
point	[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}$]	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$]	$T_{i,z,f} T_{d,z,f}$]	$T_{i,z,r} T_{d,z,r}$
2	18.7	36633	[11]	[1 0]	[1 0]	[0.042 0.003 0.008]	[0.048 0.001 0.009]	0	0



Figure C.3: Overview (left) and zoom (right) of the front and rear wheel actuator outputs u_f and u_r , and platform acceleration \ddot{z} for the calculated model-based response (red) and the response measured on the setup (blue) for Pareto point 2

Table C.4: Pareto front optimization results for Pareto point 3 (rounded). Inactive controllers are indicated with 'o'.

Pareto	Cost	Fitness	$[i_{act,f} \dots$	$[b_{sen,\ddot{z}} \dots$	$[b_{CL,\ddot{z}} \dots$	$[K_{p,\ddot{z},f}\dots$	$[K_{p,\ddot{z},r} \dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
point	[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}$]	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$]	$T_{i,z,f} T_{d,z,f}$	$T_{i,z,r} T_{d,z,r}$]
3	27.6	26795	[1 2]	[1 0]	[1 0]	[0.043 0.005 0.004]	[0.036 0.007 0.007]	0	0



Figure C.4: Overview (left) and zoom (right) of the front and rear wheel actuator outputs u_f and u_r , and platform acceleration \ddot{z} for the calculated model-based response (red) and the response measured on the setup (blue) for Pareto point 3

Table C.5: Pareto front optimization results for Pareto point 4 (rounded). Inactive controllers are indicated with 'o'.

Pare	eto C	Cost	Fitness	$[i_{act,f} \dots$	$[b_{sen,\ddot{z}} \dots$	$[b_{CL,\ddot{z}} \dots$	$[K_{p,\ddot{z},f} \dots$	$[K_{p,\ddot{z},r} \dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
poi	nt [[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}]$	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$]	$T_{i,z,f} T_{d,z,f}$	$T_{i,z,r} T_{d,z,r}$
4	3	36.6	17800	[2 2]	[1 0]	[1 0]	[0.049 0.09 0.003]	[0.049 0.007 0.008]	0	0



Figure C.5: Overview (left) and zoom (right) of the front and rear wheel actuator outputs u_f and u_r , and platform acceleration \ddot{z} for the calculated model-based response (red) and the response measured on the setup (blue) for Pareto point 4

Table C.6: Pareto front optimization results for Pareto point 5 (rounded). Inactive controllers are indicated with 'o'.

Pareto	Cost	Fitness	$[i_{act,f} \dots$	$[b_{sen,\ddot{z}} \dots$	$[b_{CL,\ddot{z}} \dots$	$[K_{p,\ddot{z},f} \dots$	$[K_{p,\ddot{z},r} \dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
point	[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}$]	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$]	$T_{i,z,f} T_{d,z,f}$	$T_{i,z,r} T_{d,z,r}$]
5	66.4	16332	[3 2]	[1 0]	[1 0]	[0.048 0.009 0.009]	[0.043 0.004 0.007]	0	0



Figure C.6: Overview (left) and zoom (right) of the front and rear wheel actuator outputs u_f and u_r , and platform acceleration \ddot{z} for the calculated model-based response (red) and the response measured on the setup (blue) for Pareto point 5

Table C.7: Pareto front optimization results for Pareto point 6 (rounded). Inactive controllers are indicated with 'o'.

Pareto	Cost	Fitness	$[i_{act,f} \dots$	$[b_{sen,\ddot{z}} \dots$	$[b_{CL,\ddot{z}}$	$[K_{p,\ddot{z},f} \dots$	$[K_{p,\ddot{z},r}\dots$	$[K_{p,z,f} \dots$	$[K_{p,z,r} \dots$
point	[%]	value []	$i_{act,r}]$	$b_{sen,z}$]	$b_{CL,z}$]	$T_{i,\ddot{z},f} T_{d,\ddot{z},f}$]	$T_{i,\ddot{z},r} T_{d,\ddot{z},r}$]	$T_{i,z,f} T_{d,z,f}$]	$T_{i,z,r} T_{d,z,r}$
6	96.3	14404	[3 3]	[1 0]	[1 0]	[0.05 0 0.01]	[0.05 0.01 0.01]	0	0



Figure C.7: Overview (left) and zoom (right) of the front and rear wheel actuator outputs u_f and u_r , and platform acceleration \ddot{z} for the calculated model-based response (red) and the response measured on the setup (blue) for Pareto point 6
Bibliography

- [1] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. London, UK: Princeton University Press, 2015.
- [2] R. Routledge, *Discoveries and inventions of the nineteenth century*. London, UK: George Routlegde and Sons, 2018.
- [3] Greelane, "James Hargreaves de uitvinden ing van de Spinning Jenny." 2019. [Onhttps://www.greelane.com/nl/geesteswetenschappen/ line]. Available: geschiedenis--cultuur/who-invented-the-spinning-jenny-4057900/
- [4] E. A. Posselt, "The Jacquard machine analyzed and explained: with an appendix on the preparation of Jacquard cards," Pennsylvania museum and school of industrial art, Philadelphia, Tech. Rep., 1887.
- [5] S. Beckert, *Katoen. De opkomst van de moderne wereldeconomie.* Amsterdam, The Netherlands: Hollands Diep, Overamstel Uitgevers, 2016.
- [6] J. Osterhammel, *Die Verwandlung der Welt. Eine Geschichte des 19. Jahrhunderts.* Munchen, Germany: Kosel, Krugzell, 2009.
- [7] S. Meyer and D. A. Hounshell, From the American System to Mass Production 1800-1932: The Development of Manufacturing Technology in the United States. Baltimore, Maryland: Johns Hopkins University Press, 1986, vol. 27, no. 1.
- [8] S. I. Museums, "Kelham Island Museum, Sheffield." [Online]. Available: http://www.simt.co.uk/kelham-island-museum
- [9] H. Shipler Commercial Photographers; Shipler, "Ford Auto," Utah; Salt Lake County; Salt Lake City, 1910. [Online]. Available: https: //collections.lib.utah.edu/ark:/87278/s66401f0
- [10] W. Bolton, *Programmable Logic Controllers*, 6th ed. Oxford, UK: Newnes, 2015.

- [11] K. Schwab, *The Fourth Industrial Revolution*. London, UK: Crown, dec 2017.
- [12] M. Moore, "What is Industry 4.0? Everything you need to know," *TechRadar*, pp. 1–8, nov 2019.
- [13] K. J. Åström, H. Elmqvist, and S. E. Mattsson, "Evolution of Continuous-Time Modeling and Simulation," *12th European Simulation Multiconference, ESM 98*, pp. 1–10, 1998.
- [14] P. Carreira, V. Amaral, and H. Vangheluwe, Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems. Cham, Switzerland: Springer Nature Switzerland AG, 2020.
- [15] K. Johan Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, 2nd ed. Oxfordshire, UK: Princeton University Press, 2019.
- [16] M. Dikmen and C. M. Burns, "Autonomous Driving in the Real World: Experiences with Tesla Autopilot and Summon," *International Conference* on Automotive User Interfaces and Interactive Vehicular Applications, pp. 225–228, 2016.
- [17] B. Brown, "The Social Life of Autonomous Cars," *Computer*, vol. 50, no. 2, pp. 92–96, 2017.
- [18] N. L. Tenhundfeld, E. J. de Visser, A. J. Ries, V. S. Finomore, and C. C. Tossell, "Trust and Distrust of Automated Parking in a Tesla Model X," *Human Factors*, vol. 62, no. 2, pp. 194–210, 2020.
- [19] P. J. Mosterman and J. Zander, "Industry 4.0 as a Cyber-Physical System study," *Software and Systems Modeling*, vol. 15, no. 1, pp. 17–29, 2016.
- [20] D. Lipika and C. Mashrur, *Transportation cyber-physical systems*, 1st ed. Amsterdam, The Netherlands: Elsevier, 2018.
- [21] K. Vanherpen, "A contract-based approach for multi-viewpoint consistency in the concurrent design of cyber-physical systems," Ph.D. dissertation, University of Antwerp, 2018.
- [22] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Tech. Rep., 2008.
- [23] W. W. Royce, "Managing the Development of large Software Systems," in ICSE: Proceedings of the 9th international conference on Software Engineering, Washington, DC, USA, 1970, pp. 1–9.

- [24] R. S. Pressman, Software Engineering: A Practitioner's Approach, 8th ed. New York, NY, USA: McGraw-Hill Education, 2015.
- [25] R. Aarenstrup, *Managing Model-Based Design*. Natick, Massachusetts, United States: The MathWorks, Inc., 2015.
- [26] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.
- [27] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.
- [28] T. Dyba and T. Dingsoyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 833–859, 2008.
- [29] C.-Y. Chen and C.-C. Cheng, "Integrated design for a mechatronic feed drive system of machine tools," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 1, pp. 588–593, 2005.
- [30] D. Roy, L. Zhang, W. Chang, and S. Chakraborty, "Automated synthesis of cyber-physical systems from joint controller/architecture specifications," in *Forum on Specification and Design Languages (FDL)*. Bremen, Germany: IEEE, 2016, pp. 1–8.
- [31] J. A. Reyer and P. Y. Papalambros, "Optimal Design and Control of an Electric DC Motor," ASME Design Engineering Technical Conferences, pp. 1– 12, 1999.
- [32] M. Moradi, M. Naraghi, and A. Kamali Eigoli, "Optimal codesign of controller and linear plants with input saturation: The sensitivity Lyapunov approach," *Optimal Control Applications and Methods*, vol. 39, no. 2, pp. 622– 637, 2018.
- [33] P. V. Chanekar, N. Chopra, and S. Azarm, "Co-design of linear systems using Generalized Benders Decomposition," *Automatica*, vol. 89, pp. 180– 193, 2018.
- [34] H. Fathy, P. Papalambros, A. Ulsoy, and D. Hrovat, "Nested plant/controller optimization with application to combined passive/active automotive suspensions," *Proceedings of the 2003 American Control Conference, 2003.*, vol. 4, pp. 3375–3380, 2003.
- [35] L. P. Diane, K. Katsuo, Y. P. Panos, and A. G. Ulsoy, "Co-Design of a MEMS Actuator and its Controller Using Frequency Constraints," *Proceedings of the ASME 2008 Dynamic Systems and Control Conference. ASME* 2008 Dynamic Systems and Control Conference, Parts A and B., vol. 20-22, pp. 801–807, 2008.

- [36] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control Analysis and Design*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons, 2001.
- [37] Flanders Make, "SBO Project Application: Robust and Optimal Control for Systems of Interacting Subsystems," 2015.
- [38] E. Silvas, T. Hofman, N. Murgovski, L. F. P. Etman, and M. Steinbuch, "Review of Optimization Strategies for System-Level Design in Hybrid Electric Vehicles," *Ieee Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 57–70, 2017.
- [39] T. Baeck, D. Fogel, and Z. Michalewicz, Evolutionary Computation 1: Basic Algorithms and Operators, 1st ed. Boca Raton, FL, USA: CRC Press, 2000.
- [40] D. Simon and Cleveland State University, *Evolutionary optimization algorithms*, 1st ed. Cleveland, OH, USA: John Wiley & Sons, 2013.
- [41] M. Thone, M. Potters, and S. Baldi, "Control configurations in distillation columns: A comparative study," in 2016 European Control Conference, ECC 2016, Aalborg, Denkmark, 2016, pp. 37–42.
- [42] M. H. De Queiroz and J. E. Cury, "Modular control of composed systems," in *Proceedings of the American Control Conference*, vol. 6, Chicago, Illinois, 2000, pp. 4051–4055.
- [43] J. Dong, X. Yang, Q. Liu, Z. Wang, and T. Wang, "Design and implementation of CNC controllers using reconfigurable hardware," in 2009 IEEE International Conference on Control and Automation, ICCA 2009, Christchurch, New Zealang, 2009, pp. 1481–1486.
- [44] G. Yong, S. Yushan, M. Yufeng, and W. Lei, "Design of Semi Physical Motion Simulation System of Underwater Robot," in *Proceedings of the* 25th chinese control conference, Harbin, 2007, pp. 1601–1604.
- [45] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., P. Glynn and S. M. Robinson, Eds. New York, NY, USA: Springer, 1999.
- [46] K. Deep, K. P. Singh, M. L. Kansal, and C. Mohan, "A real coded genetic algorithm for solving integer and mixed integer optimization problems," *Applied Mathematics and Computation*, vol. 212, no. 2, pp. 505–518, 2009.
- [47] J. Clausen, "Branch and Bound Algorithms Principles and Examples," Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, Tech. Rep., 1999.

- [48] J. Reyer, "Combined Embodiment Design and Control Optimization: Effects of Cross-Disciplinary Coupling," Ph.D. dissertation, University of Michigan, 2000.
- [49] H. K. Fathy, J. A. Reyer, P. Y. Papalambros, and A. G. Ulsoy, "On the coupling between the plant and controller optimization problems," *Proceedings* of the American Control Conference, vol. 3, pp. 1864–1869, 2001.
- [50] M. J. Smith, K. M. Grigoriadis, and R. E. Skelton, "Optimal Mix of Active and Passive Control in Structures," *Journal of Guidance, Control, and Dynamics*, vol. 15, pp. 912–919, 1992.
- [51] R. Patil, Z. Filipi, and H. Fathy, "Computationally Efficient Combined Design and Control Optimization using a Coupling Measure," *IFAC Proceedings Volumes*, vol. 43, no. 18, pp. 144–151, 2010.
- [52] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., M. J. Horton, A. Gilfillan, A. Dworkin, and S. Disanno, Eds. Upper Saddle River, NJ, USA: Prentice-Hall, 2011.
- [53] D. E. Rivera, "Internal model control: a comprehensive view," Tempe, Arizona, USA, 1999.
- [54] S. Zhao, R. Cajo, R. D. Keyser, and C. M. Ionescu, "The potential of fractional order distributed MPC applied to steam/water loop in large scale ships," *Processes*, vol. 8, no. 4, 2020.
- [55] C. M. Ionescu, E. H. Dulf, M. Ghita, and C. I. Muresan, "Robust controller design: Recent emerging concepts for control of mechatronic systems," *Journal of the Franklin Institute*, vol. 357, no. 12, pp. 7818–7844, 2020.
- [56] B. Wu, C. Liu, X. Song, and X. Wang, "Design and implementation of the inverted pendulum optimal controller based on hybrid genetic algorithm," 2015 International Conference on Automation, Mechanical Control and Computational Engineering, pp. 1480–1486, 2015.
- [57] C. Wongsathan and C. Sirima, "Application of GA to design LQR controller for an inverted pendulum system," 2008 IEEE International Conference on Robotics and Biomimetics, ROBIO 2008, no. 2, pp. 951–954, 2008.
- [58] M. Nagarkar and G. J. Vikhe Patil, "Optimization of the linear quadratic regulator (LQR) control quarter car suspension system using genetic algorithm," *Revista Ingenieria E Investigacion*, vol. 36, no. 1, pp. 23–30, 2016.
- [59] M. Haris, I.-U.-H. Shaikh, and H. Shoaib, "Genetic Algorithm Based LQR Control Of Hovercraft," in 2016 International Conference on Intelligent Systems Engineering (ICISE). Islamabad, Pakistan: IEEE, 2016, pp. 335–339.

- [60] I. Robandi, K. Nishimori, R. Nishimura, and N. Ishihara, "Optimal feedback control design using genetic algorithm in multimachine power system," *International Journal of Electrical Power and Energy Systems*, vol. 23, no. 4, pp. 263–271, 2001.
- [61] H. Du, J. Lam, and K. Y. Sze, "Non-fragile output feedback Hinfinity vehicle suspension control using genetic algorithm," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 7-8, pp. 667–680, 2003.
- [62] G. Duc, "Designing a Low Order Robust Controller for an Active Suspension System Thank LMI, Genetic Algorithm and Gradient Search," *European Journal of Control*, vol. 9, no. 1, pp. 29–38, 2003.
- [63] N. R. Raju and P. L. Reddy, "Optimal Tuning of Fractional Order PID Controller for Automatic Voltage Regulator System through Genetic Algorithm," vol. 8, no. 3, pp. 922–927, 2016.
- [64] M. P. Nagarkar, M. A. El-Gohary, Y. J. Bhalerao, G. J. Vikhe Patil, and R. N. Zaware Patil, "Artificial neural network predication and validation of optimum suspension parameters of a passive suspension system," *SN Applied Sciences*, vol. 1, no. 6, 2019.
- [65] A. Baumal, J. McPhee, and P. Calamai, "Application of genetic algorithms to the design optimization of an active vehicle suspension system," *Computer Methods in Applied Mechanics and Engineering*, vol. 163, no. 1-4, pp. 87–94, 1998.
- [66] Z. Affi, B. EL-Kribi, and L. Romdhane, "Advanced mechatronic design using a multi-objective genetic algorithm optimization of a motor-driven fourbar system," *Mechatronics*, vol. 17, no. 9, pp. 489–500, 2007.
- [67] J. N. Martinez-Castelan and M. G. Villarreal-Cervantes, "Frontal-Sagittal Dynamic Coupling in the Optimal Design of a Passive Bipedal Walker," *IEEE Access*, vol. 7, pp. 427–449, 2019.
- [68] P. V. Chanekar, N. Chopra, and S. Azarm, "Optimal actuator placement for linear systems with limited number of actuators," *Proceedings of the American Control Conference*, pp. 334–339, 2017.
- [69] M. Babazadeh, "Robust controllability assessment and optimal actuator placement in dynamic networks," *Systems and Control Letters*, vol. 133, 2019.
- [70] R. Semaan, "Optimal sensor placement using machine learning," *Computers & Fluids*, vol. 159, pp. 167–176, 2017.

- [71] B. Rakhim, A. Zhakatayev, O. Adiyatov, and H. A. Varol, "Optimal Sensor Placement of Variable Impedance Actuated Robots," *Proceedings of the 2019 IEEE/SICE International Symposium on System Integration, SII 2019*, pp. 141–146, 2019.
- [72] A. Nandy, D. Chakraborty, and M. S. Shah, "Optimal Sensors/Actuators Placement in Smart Structure Using Island Model Parallel Genetic Algorithm," *International Journal of Computational Methods*, vol. 16, no. 6, 2019.
- [73] S. K. Ahmed, J. K. Peng, and D. J. Chmielewski, "Covariance-based hardware selection-part III: Distributed parameter systems," *AIChE Journal*, vol. 58, no. 9, pp. 2705–2713, 2012.
- [74] T.-H. Yan and R.-M. Lin, "General optimization of sizes or placement for various sensors/actuators in structure testing and control," *Smart Mater. Struct.*, vol. 15, no. 3, pp. 724–736, 2006.
- [75] Y.-P. Yang and Y.-A. Chen, "Multiobjective optimization of hard disk suspension assemblies: Part II - integrated structure and control design," *Computers & Structures*, vol. 59, no. 4, pp. 771–782, 1996.
- [76] M. G. Villarreal-Cervantes, "Approximate and Widespread Pareto Solutions in the Structure-Control Design of Mechatronic Systems," *Journal of Optimization Theory and Applications*, vol. 173, no. 2, pp. 628–657, 2017.
- [77] J.-H. Park and H. Asada, "Concurrent design optimization of mechanical structure and control for high speed robots," *American Control Conference*, pp. 2673–2679, 1993.
- [78] A. Baheri and C. Vermillion, "Combined Plant and Controller Design Using Batch Bayesian Optimization: A Case Study in Airborne Wind Energy Systems," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 141, no. 9, 2019.
- [79] Y. S. Wang and Y. Wang, "A gradient-based approach for optimal plant controller co-design," *Proceedings of the American Control Conference*, no. July, pp. 3249–3254, 2015.
- [80] J. Ma, S. L. Chen, C. S. Teo, A. Tay, A. Al Mamun, and K. K. Tan, "Parameter space optimization towards integrated mechatronic design for uncertain systems with generalized feedback constraints," *Automatica*, vol. 105, pp. 149–158, 2019.
- [81] Q. Chen, T. Lin, and H. Ren, "Parameters optimization and control strategy of power train systems in hybrid hydraulic excavators," *Mechatronics*, vol. 56, no. 668, pp. 16–25, 2018.

- [82] W. Yang, H. Yang, and S. Tang, "Optimization and control application of sensor placement in aeroservoelastic of UAV," *Aerospace Science and Technology*, vol. 85, pp. 61–74, 2019.
- [83] F. Lin and V. Adetola, "Co-design of sparse output feedback and row/column-sparse output matrix," *Proceedings of the American Control Conference*, pp. 4359–4364, 2017.
- [84] X. Chen, "Joint Actuator-Sensor Design for Stochastic Linear Systems," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2018, no. December, pp. 6668–6673, 2019.
- [85] I. Bruant and L. Proslier, "Optimal location and gains of sensors and actuators for feedback vibroacoustic control of plates," *Journal of Intelligent Material Systems and Structures*, vol. 16, no. 3, pp. 197–206, 2005.
- [86] C. Y. Chang, S. Martínez, and J. Cortés, "Co-Optimization of Control and Actuator Selection for Cyber-Physical Systems," *IFAC-PapersOnLine*, vol. 51, no. 23, pp. 118–123, 2018.
- [87] B. Majid and S. Majeed, "Genetic Sensor Placement in Active Control of a Robotic Arm," *Proceedings - International Conference on Developments in eSystems Engineering, DeSE*, vol. 06, pp. 279–284, 2017.
- [88] R. Bernhard, E. Michael, R. Achim, and B. Richard, "The shuttle radar topography mission - a new class of digital elevation models acquired by spaceborne radar," *ISPRS journal of photogrammetry and remote sensing*, vol. 57, no. 4, pp. 241–262, 2003.
- [89] D. Li, W. Liu, J. Jiang, and R. Xu, "Placement optimization of actuator and sensor and decentralized adaptive fuzzy vibration control for large space intelligent truss structure," *Science China Technological Sciences*, vol. 54, no. 4, pp. 853–861, 2011.
- [90] B. Xu and J. S. Jiang, "Integrated optimization of structure and control for piezoelectric intelligent trusses with uncertain placement of actuators and sensors," *Computational Mechanics*, vol. 33, no. 5, pp. 406–412, 2004.
- [91] V. Bottega, A. Molter, O. A. A. Da Silveira, and J. S. O. Fonseca, "Simultaneous piezoelectric actuator and sensor placement optimization and control design of manipulators with flexible links using SDRE method," *Mathematical Problems in Engineering*, p. 362437, 2010.
- [92] W. P. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to eventtriggered and self-triggered control," *Proceedings of the IEEE Conference* on Decision and Control, pp. 3270–3285, 2012.

- [93] X. M. Zhang, Q. L. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng, "Networked control systems: A survey of trends and techniques," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 1–17, 2020.
- [94] L. U. Zibao and G. U. O. Ge, "Co-design of Controller and Communication Sequences," in 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, China, 2016, pp. 278–283.
- [95] W. Chen and L. Qiu, "A channel/controller co-design approach for infinitehorizon LQR problem with random input gains," 2013 IEEE International Conference on Information and Automation, ICIA 2013, pp. 844–849, 2013.
- [96] S. M. Noorbakhsh and J. Ghaisari, "Event-Based Consensus Controller for Linear Multi-Agent Systems Over Directed Communication Topologies: A Co-Design Approach," *Asian Journal of Control*, vol. 18, no. 5, pp. 1934– 1939, 2016.
- [97] D. Ma, J. Han, D. Zhang, and Y. Liu, "Co-Design of Event Generator and Dynamic Output Feedback Controller for LTI Systems," *Mathematical Problems in Engineering*, vol. 2015, pp. 7–14, 2015.
- [98] D. Ye and S. Luo, "A co-design methodology for cyber-physical systems under actuator fault and cyber attack," *Journal of the Franklin Institute*, vol. 356, no. 4, pp. 1856–1879, 2019.
- [99] T. Shi, T. Tang, and J. Bai, "Distributed event-triggered control co-design for large-scale systems via static output feedback," *Journal of the Franklin Institute*, vol. 356, no. 17, pp. 10393–10404, 2019.
- [100] E. Polak, Optimization: Algorithms and Consistent Approximations, 1st ed. New York, NY, USA: Springer-Verlag, 1997.
- [101] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, 1944.
- [102] J. Huang, P. Hu, K. Wu, and M. Zeng, "Optimal time-jerk trajectory planning for industrial robots," *Mechanism and Machine Theory*, vol. 121, pp. 530–544, 2018.
- [103] R. Shankar, J. Marco, and F. Assadian, "The novel application of optimization and charge blended energy management control for component downsizing within a plug-in hybrid electric vehicle," *Energies*, vol. 5, no. 12, pp. 4892–4923, 2012.
- [104] D. G. Li, Y. Zou, X. S. Hu, and F. C. Sun, "Optimal sizing and control strategy design for heavy hybrid electric truck," 2012 IEEE Vehicle Power and Propulsion Conference, VPPC 2012, pp. 1100–1106, 2012.

- [105] N. Murgovski, L. Johannesson, J. Sjöberg, and B. Egardt, "Component sizing of a plug-in hybrid electric powertrain via convex optimization," *Mechatronics*, vol. 22, no. 1, pp. 106–120, 2012.
- [106] M. V. Klibanov, A. E. Kolesov, and D. L. Nguyen, "Convexification method for an inverse scattering problem and its performance for experimental backscatter data for buried targets," *SIAM Journal on Imaging Sciences*, vol. 12, no. 1, pp. 576–603, 2019.
- [107] N. Van Oosterwyck, A. B. Yahya, A. Cuyt, and S. Derammelaere, "CAD based trajectory optimization of PTP motions using chebyshev polynomials," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 2020-July, pp. 403–408, 2020.
- [108] Frontline Systems, "Optimization Problem Types," 2018. [Online]. Available: https://www.solver.com/nonsmooth-optimization
- [109] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, mps-siam b ed. Philadelphia: SIAM, 2009.
- [110] C. Audet and M. Kokkolaras, "Blackbox and derivative-free optimization: theory, algorithms and applications," *Optimization and Engineering*, vol. 17, no. 1, pp. 1–2, 2016.
- [111] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, "Direct Multisearch for Multiobjective Optimization," *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 1109–1140, 2011.
- [112] W. Gao and S. K. Porandla, "Design optimization of a parallel hybrid electric powertrain," in 2005 IEEE Vehicle Power and Propulsion Conference, Chicago, IL, USA, 2005, p. 6.
- [113] M. J. Powell, "On search directions for minimization algorithms," *Mathe-matical Programming*, vol. 4, no. 1, pp. 193–201, 1973.
- [114] T. G. Kolda and V. J. Torczon, "On the convergence of asynchronous parallel pattern search," *SIAM Journal on Optimization*, vol. 14, no. 4, pp. 939–964, 2004.
- [115] R. Hooke and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," *Journal of the ACM (JACM)*, vol. 8, no. 2, pp. 212–229, 1961.
- [116] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, Massachusetts: Athena Scientific, 1976, vol. 9.
- [117] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

- [118] I. Loshchilov, M. Schoenauer, and M. Sebag, "Adaptive coordinate descent," *Genetic and Evolutionary Computation Conference, GECCO'11*, pp. 885– 892, 2011.
- [119] N. Hansen, "Adaptive encoding: how to render search coordinate system invariant," in *International Conference on Parallel Problem Solving from Nature*, Berlin, Germany, 2008, pp. 205–214.
- [120] J. Mockus, "On Bayesian Methods for Seeking the Extremum," in *Lecture Notes in Computer Science (LNCS)*. Berlin, Germany: Springer Berlin Heidelberg, 1974, pp. 400–404.
- [121] —, *Bayesian Approach to Global Optimization*. Dordrecht: Kluwer Academic Publications, 1989, vol. 37, no. D.
- [122] —, "The application of Bayesian methods for seeking the extremum," in *Towards Global Optimisation*, L. Dixon and G. Szego, Eds. Amsterdam, The Netherlands: North-Holland, 2014.
- [123] Y. Koyama, I. Sato, D. Sakamoto, and T. Igarashi, "Sequential line search for efficient visual design optimization by crowds," ACM Transactions on Graphics, vol. 36, no. 4, pp. 48:1–48:11, 2017.
- [124] R. Martinez-Cantin, N. De Freitas, E. Brochu, J. Castellanos, and A. Doucet, "A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot," *Autonomous Robots*, vol. 27, no. 2, pp. 93–103, 2009.
- [125] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 4, pp. 2951–2959, 2012.
- [126] A. D. Bull, "Convergence rates of efficient global optimization algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2879–2904, 2011.
- [127] R. Moriconi, M. P. Deisenroth, and K. S. Sesh Kumar, "High-dimensional Bayesian optimization using low-dimensional feature spaces," *Machine Learning*, vol. 109, no. 9-10, pp. 1925–1943, 2020.
- [128] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton, "High dimensional Bayesian optimization using dropout," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 0, pp. 2096–2102, 2017.
- [129] D. R. Jones, "Direct Global Optimization Algorithm," *Encyclopedia of Optimization*, pp. 725–735, 2008.
- [130] S. Kirkpatrick, C. D. J. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

- [131] P. van Laarhoven and E. Aarts, *Simulated Annealing: Theory and Applications*, 1st ed. Dordrecht, The Netherlands: Springer Netherlands, 1987.
- [132] H.-M. Gutmann, "A radial basis function method for global optimization," *Journal of Global Optimization*, no. 3, pp. 201–227, 2001.
- [133] Y. Wang and C. A. Shoemaker, "A General Stochastic Algorithmic Framework for Minimizing Expensive Black Box Objective Functions Based on Surrogate Models and Sensitivity Analysis," *arXiv: Machine Learning*, p. 31, 2014.
- [134] H.-P. Schwefel, Evolution and Optimum Seeking. New York, NY, USA: Wiley & Sons, 1995.
- [135] H.-G. Beyer, *The Theory of Evolution Strategies*, 1st ed. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2001.
- [136] H.-P. Schwefel and H.-G. Beyer, "Evolution Strategies: A Comprehensive Introduction," *Journal Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.
- [137] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942–1948.
- [138] M. Reza Bonyadi and Z. Michalewicz, "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review," *Evolutionary Computation (MIT)*, vol. 25, no. 1, pp. 1–54, 2017.
- [139] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed. Luniver Press, 2008.
- [140] M. Dorigo, T. Stützle, and M. Birattari, "Ant Colony Optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.
- [141] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Erciyes University, Kayseri, Tech. Rep., 2005.
- [142] G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, and J. Tan, "Modeling and Planning for Dual-objective Selective Disassembly Using AND/OR Graph and Discrete Artificial Bee Colony," *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–12, 2018.
- [143] V. H. Dang, N. A. Vien, and T. C. Chung, "A covariance matrix adaptation evolution strategy in reproducing kernel Hilbert space," *Genetic Programming and Evolvable Machines*, vol. 20, no. 4, pp. 479–501, 2019.

- [144] J. Sawicki, M. Łoś, M. Smołka, and J. Alvarez-Aramberri, "Using Covariance Matrix Adaptation Evolutionary Strategy to boost the search accuracy in hierarchic memetic computations," *Journal of Computational Science*, vol. 34, pp. 48–54, 2019.
- [145] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multiobjective optimization," *Evolutionary Computation*, vol. 15, no. 1, pp. 1–28, 2007.
- [146] G. Fujii, M. Takahashi, and Y. Akimoto, "CMA-ES-based structural topology optimization using a level set boundary expression - Application to optical and carpet cloaks," *Computer Methods in Applied Mechanics and Engineering*, vol. 332, pp. 624–643, 2018.
- [147] A. Conn, N. Gould, and P. Toint, "A Globally Convergent Augmented Lagrangian Barrier Algorithm for Optimization with General Inequality Constraints and Simple Bounds," *Mathematics of Computation*, vol. 66, no. 217, pp. 261–288, 1997.
- [148] T. Singh, J. Swevers, and G. Pipeleers, "Concurrent H2 /Hinfinity feedback control design with optimal sensor and actuator selection," *Proceedings -*2018 IEEE 15th International Workshop on Advanced Motion Control, AMC 2018, pp. 223–228, 2018.
- [149] S. F. Hwang and R. S. He, "A hybrid real-parameter genetic algorithm for function optimization," *Advanced Engineering Informatics*, vol. 20, no. 1, pp. 7–21, 2006.
- [150] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.
- [151] The MathWorks Inc., "MATLAB Global Optimization Toolbox User's Guide (Release 2018b)," Natick, MA, United States, p. 780, 2018.
- [152] D. Kalyanmoy, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 311–338, 2000.
- [153] J. A. Reyer, H. K. Fathy, P. Y. Papalambros, and A. G. Ulsoy, "Comparison of combined embodiment design and control optimization strategies using optimality conditions," *Proceedings of the ASME Design Engineering Technical Conference*, vol. 2, pp. 1023–1032, 2001.
- [154] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, 1st ed. New York, NY, USA: Wiley & Sons, 2001.

- [155] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, no. 4, pp. 41–59, 1977.
- [156] E. Goodarzi, M. Ziaei, and E. Z. Hosseinipour, Introduction to Optimization Analysis in Hydrosystem Engineering. Cham, Switzerland: Springer International Publishing, 2014.
- [157] A. Jahan, K. Edwards, and M. Bahraminasab, *Multi-criteria Decision Anal-ysis*, 2nd ed. Amsterdam: Elsevier, 2013.
- [158] M. Verhaegen, "Identification of the deterministic part of MIMO state space models given in innovations form from input-output data," *Automatica*, vol. 30, no. 1, pp. 61–74, 1994.
- [159] B. Friedland, Control System Design: An Introduction to State-Space Methods. Mineola, New York: Dover Publications, Inc., 1986.
- [160] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, "Control Systems Design," *International Journal of Adaptive Control and Signal Processing*, vol. 16, no. 2, pp. 173–174, 2002.
- [161] M. M. Share Pasand, "Luenberger-type cubic observers for state estimation of linear systems," *International Journal of Adaptive Control and Signal Processing*, vol. 34, no. 9, pp. 1148–1161, 2020.
- [162] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, 1960.
- [163] S. N. Nise, Control Systems Engineering, 6th ed. New York, USA: John Wiley & Sons, 2011.
- [164] J. Goos, "Modeling and identification of Linear Parameter-Varying systems," Ph.D. dissertation, Vrije Universieit Brussel, 2016.
- [165] J. S. Shamma and M. Athans, "Gain Scheduling: Potential Hazards and Possible Remedies," *IEEE Control Systems*, vol. 12, no. 3, pp. 101–107, 1992.
- [166] G. J. Balas, "Linear, Parameter-Varying Control And Its Application To Aerospace Systems," *Proceedings of the International Congress of the Aeronautical Sciences*, pp. 1–9, 2002.
- [167] J. Mohammadpour and C. W. Scherer, *Control of Linear Parameter Varying Systems with Applications*, 1st ed. New York, NY, USA: Springer-Verlag, 2012.

- [168] K. Havre and S. Skogestad, "Selection of Variables for Regulatory Control Using Pole Vectors," in *Proc. IFAC symposium DYCOPS-5*, vol. 31, no. 11, Corfu, Greece, 1998, pp. 614–619.
- [169] G. Mackiw, "A Note on the Equality of the Column and Row Rank of a Matrix," *Mathematics Magazine*, vol. 68, no. 4, p. 285, 1995.
- [170] B. Anderson and J. Moore, *Optimal Control: Linear Quadratic Methods*. Englewood Cliffs, NJ, USA: Prentice Hall, 1990.
- [171] W. Li, G. Wei, D. W. Ho, and D. Ding, "A Weightedly Uniform Detectability for Sensor Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5790–5796, 2018.
- [172] K. Hangos, J. Bokor, and G. Szederkenyi, Analysis and Control of Nonlinear Process Systems, 1st ed. London, UK: Springer-Verlag, 2004.
- [173] Y. Shen, W.-j. Cai, and S. Li, "Multivariable Process Control: Decentralized, Decoupling, or Sparse?" *Industrial and Engineering Chemistry Research*, vol. 49, no. 2, pp. 761–771, 2010.
- [174] C. C. Wo and Z. Q. Min, "Coupling and decoupling control study on aircraft (Airbus A320)," 2016 8th International Conference on Modelling, Identification and Control (ICMIC), pp. 29–36, 2016.
- [175] C. Yang, Q. Huang, and J. Han, "Decoupling control for spatial six-degreeof-freedom electro-hydraulic parallel robot," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 14–23, 2012.
- [176] R. Scattolini, "Architectures for distributed and hierarchical Model Predictive Control - A review," *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, 2009.
- [177] P. Massioni and M. Verhaegen, "Distributed control for identical dynamically coupled systems: A decomposition approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 124–135, 2009.
- [178] I. Tomi and G. D. Halikias, "Performance analysis of distributed control configurations in LQR multi-agent system design," in 2016 UKACC 11th International Conference on Control (CONTROL), Belfast, UK, 2016, pp. 1–6.
- [179] A. Maxim, D. Copot, R. De Keyser, and C. M. Ionescu, "An industrially relevant formulation of a distributed model predictive control algorithm based on minimal process information," *Journal of Process Control*, vol. 68, pp. 240–253, 2018.

- [180] H. R. Dharmayanda, K. Taesam, J. L. Young, and S. Sangkyung, "Motion stability of small scale helicopter using state feedback," *ICCAS 2007 - International Conference on Control, Automation and Systems*, pp. 1439–1444, 2007.
- [181] A. E. Bryson and Y.-C. Ho, Applied Optimal Control: Optimization, Estimation and Control, 1st ed. London, UK: Taylor & Francis Inc., 1975.
- [182] J.-L. Wu, "A Simultaneous Mixed LQR / Hinfinity Control Approach to the Design of Reliable Active Suspension Controllers," Asian Journal of Control, vol. 19, no. 2, pp. 415–427, 2017.
- [183] T. Singh, W. Decre, J. Swevers, and G. Pipeleers, "Concurrent design of an active vibration feedback controller and actuator/sensor selection for a composite plate," in *6th Indian Control Conference (ICC)*. Hyderabad, India: IEEE, 2019, pp. 146–151.
- [184] X. Shao and D. Sun, "Development of a new robot controller architecture with FPGA-based IC design for improved high-speed performance," in *IEEE Transactions on Industrial Informatics*, vol. 3, no. 4, 2007, pp. 312– 321.
- [185] Beckhoff Automation, "Beckhoff TC3 Controller Toolbox Manual," pp. 1– 180, 2019.
- [186] Siemens AG, "Siemens SINAMICS S150, Function Diagram, Control Version V2.6 SP1," 2009.
- [187] W. Bolton, *Instrumentation and Control Systems*, 2nd ed. Kidlington, Oxford, UK: Elsevier, 2015.
- [188] W. Chen, J. Liang, and T. Shi, "Speed Synchronous Control of Multiple Permanent Magnet Synchronous Motors Based on an Improved Cross-Coupling Structure," *Energies*, vol. 11, no. 2, pp. 282–298, 2018.
- [189] M. Haemers, S. Derammelaere, and K. Stockman, "Co-design of controller and setup configuration using Genetic Algorithm," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Limassol, Cyprus, 2017, pp. 1–5.
- [190] B. W. Bequette, Process Control: Modeling, Design and Simulation, 1st ed. London, UK: Pearson, 2002.
- [191] A. Musbah, A. Ahmad, and P. Mahanti, "Simulation of Digital Control Systems," Systems Analysis Modelling Simulation, vol. 42, no. 10, pp. 1419– 1428, 2002.

- [192] D. Hinrichsen and A. J. Pritchard, *Mathematical Systems Theory I: Modelling, State Space Analysis, Stability and Robustness*, 1st ed. Berlin, Germany: Springer-Verlag, 2005, vol. 48.
- [193] J. Bay, *Fundamentals of Linear State Space Systems*, 1st ed. New York, NY, USA: McGraw-Hill Science/Engineering/Math, 1999.
- [194] C. Briat, *Linear Parameter-Varying and Time-Delay Systems*, ser. Advances in Delays and Dynamics, S.-I. Niculescu, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, vol. 3.
- [195] K. Aström and B. Wittenmark, *Computer-controlled Systems: Theory and Design*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2012.
- [196] G. Franklin, D. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 3rd ed. London, UK: Pearson Education, 1998.
- [197] C. S. Burrus, Digital Signal Processing and Digital Filter Design, 1st ed. New York, NY, USA: Wiley & Sons, 1987.
- [198] B. M. Vinagre, Y. Q. Chen, and I. Petráš, "Two direct Tustin discretization methods for fractional-order differentiator/integrator," *Journal of the Franklin Institute*, vol. 340, no. 5, pp. 349–362, 2003.
- [199] G. P. Starr, *Introduction to Applied Digital Controls*, 1st ed. Cham, Switzerland: Springer International Publishing, 2020.
- [200] G. Stephanopoulos, *Chemical Process Control: An Introduction to Theory and Practice*, 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1983.
- [201] P. Bogacki and L. F. Shampine, "A 3(2) pair of Runge-Kutta formulas," *Applied Mathematics Letters*, vol. 2, no. 4, pp. 321–325, 1989.
- [202] M. Verbandt, L. Jacobs, D. Turk, T. Singh, J. Swevers, and G. Pipeleers, "Linear Control Toolbox - supporting B-splines in LPV control," *Mechatronics*, vol. 52, pp. 78–89, 2018.
- [203] I. Kucukkoc, A. D. Karaoglan, and R. Yaman, "Using response surface design to determine the optimal parameters of genetic algorithm and a case study," *International Journal of Production Research*, vol. 51, no. 17, pp. 5039–5054, 2013.
- [204] K. Deep and M. Thakur, "A new crossover operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 895–911, 2007.
- [205] —, "A new mutation operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 193, no. 1, pp. 211–230, 2007.

- [206] M. Gutowski, "Biology, Physics, Small Worlds and Genetic Algorithms," in *Leading Edge Computer Science Research*. Hauppauge, NY, USA: Nova Science Publishers, Inc., 2005, no. January, pp. 165–218.
- [207] M. Kishnani, S. Pareek, and R. Gupta, "Optimal tuning of DC motor via simulated annealing," 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), pp. 1–5, 2014.
- [208] R. Murray, "LQR Control," in *Control and Dynamical Systems*. Pasadena, CA, USA: California Institute of Technology, 2006, ch. Lecture 2, p. 14.
- [209] Z. Z. Dong, C. Faria, B. Pluymers, M. Hromčík, M. Šebek, and W. Desmet, "Structure-preserving low-order modeling approach of laminated composite plates integrated with macro-fiber composite transducers for dynamic applications," *Composite Structures*, vol. 208, pp. 287–297, 2019.
- [210] T. Singh, M. De Mauri, W. Decré, J. Swevers, and G. Pipeleers, "Feedback control of linear systems with optimal sensor and actuator selection," *Journal of Vibration and Control*, vol. July, 2020.
- [211] Z. Dong, C. Faria, M. Hromčik, B. Pluymers, M. Šebek, and W. Desmet, "Equivalent force modeling of macro fiber composite actuators integrated into non-homogeneous composite plates for dynamic applications," *Smart Materials and Structures*, vol. 26, no. 9, 2017.
- [212] J. N. Reza, *Vehicle Dynamics: Theory and Application*, 2nd ed. New York, NY, USA: Springer-Verlag, 2014.
- [213] H. Wang, G. I. Mustafa, and Y. Tian, "Model-free fractional-order sliding mode control for an active vehicle suspension system," *Advances in Engineering Software*, vol. 115, no. August 2017, pp. 452–461, 2018.
- [214] J. Glucker, "After 30 years, Bose-developed suspension tech will go into production," *Motor Authority*, 2018.
- [215] J. Cross, "The design, development and applications of MagneRide suspension," *Autocar*, 2014.
- [216] H. Tsuka, J. Nakano, Y. Yokoya, A. Fukami, and Y. Hirano, "A new electronic controlled suspension using piezo-electric ceramics," in *IEEE Workshop on Electronic Applications in Transportation*. Dearborn, MI, USA: IEEE, 1990, pp. 50–57.
- [217] H. Unuh, P. Muhamad, F. Yakub, M. A. Ismail, and Z. Tanasta, "Experimental validation to a prototype magnetorheological (MR) semi-active damper for C-Class vehicle," *International Journal of Automotive and Mechanical Engineering*, vol. 16, no. 3, pp. 7034–7047, 2019.

- [218] J. Poynor, "Innovative Designs for Magneto-Rheological Dampers," Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Tech. Rep., 2001.
- [219] A. M. Aly and R. E. Christenson, "On the evaluation of the efficacy of a smart damper: A new equivalent energy-based probabilistic approach," *Smart Materials and Structures*, vol. 17, no. 4, 2008.
- [220] B. Kong, "For Cadillac and Corvette Fans: Exploring the History of GM's Magnetic Ride Control," *Motortrend*, 2013.
- [221] J. Carlson, W. Matthis, and J. Toscano, "Smart Prosthetics Based on MR Fluids," in Smart Structures and Materials 2001: Industrial and Commercial Applications of Smart Structures Technologies, Newport Beach, CA, United States, 2001.
- [222] G. M. Kamath, N. M. Wereley, and M. R. Jolly, "Characterization of magnetorheological helicopter lag dampers," *Journal of the American Helicopter Society*, vol. 44, no. 3, pp. 234–248, 1999.
- [223] V. J. Moonjeli, "Analysis of Hydropneumatic Suspension," Amal Jyothi College of Engineering, Koovapally, Kanjirapally, Kerala, Tech. Rep., 2011.
- [224] M. Bobbitt, Citroen DS: Design Icon. Veloce Publishing Ltd., 2005.
- [225] K. Howard, "Active Suspension," Motor Sport Magazine, dec 2001.
- [226] G. Perlas, "How the Active Curve Tilting Feature of the S-Class Coupe Works," *BenzInsider.com*, 2014.
- [227] J. Yao, Z. Li, M. Wang, F. Yao, and Z. Tang, "Automobile active tilt control based on active suspension," *Advances in Mechanical Engineering*, vol. 10, no. 10, 2018.
- [228] J. Beno, D. Bresie, A. Guenin, D. Weeks, and W. Weldon, "Constant force suspension, near constant force suspension, and associated control algorithms," 1999.
- [229] A. Bryant, J. Beno, and D. Weeks, "Benefits of electronically controlled active electromechanical suspension systems (EMS) for mast mounted sensor packages on large off-road vehicles," SAE 2011 World Congress and Exhibition, vol. 01, no. 0269, p. 17, 2011.
- [230] M. Hanlon, "Bose Redefines Automobile Suspension Systems," New Atlas, sep 2004. [Online]. Available: https://newatlas.com/go/3259/
- [231] T. Higgins and L. Randall, "Motion Sick? This Tech Company Thinks It Might Have a Solution," *Wall Street Journal*, 2020.

- [232] I. Adcock, "Not fat, just big boned: new Audi A8's suspension and chassis explained," *Car Magazine*, 2017.
- [233] V. Vijayenthiran, "Audi reveals new A8's chassis technology," *Motor Authority*, 2017.
- [234] A. Alenezi, "Active Suspension Control based on a Full-Vehicle Model," *Journal of Electrical and Electronics Engineering (IOSR-JEEE)*, vol. 9, no. 2, pp. 6–18, 2014.
- [235] A. Kruczek and A. Stribrsky, "A full-car model for active suspension Some practical aspects," in *IEEE International Conference on Mechatronics*. Istanbul, Turkey: IEEE, 2004, pp. 41–45.
- [236] R. Darus and Y. M. Sam, "Modeling and control active suspension system for a full car model," 5th International Colloquium on Signal Processing and Its Applications, vol. 4, no. 7, pp. 13–18, 2009.
- [237] S. Sharma, V. Pare, M. Chouksey, and B. Rawal, "Numerical Studies Using Full Car Model for Combined Primary and Cabin Suspension," *Proceedia Technology*, vol. 23, pp. 171–178, 2016.
- [238] X.-J. Liu and J. Wang, Parallel Kinematics: Type, Kinematics, and Optimal Design, 1st ed. Berlin, Germany: Springer-Verlag, 2014.
- [239] J.-P. Merlet, *Parallel Robots*, 2nd ed. Amsterdam, The Netherlands: Springer Netherlands, 2006.
- [240] R. Ben-Horin, M. Shoham, and S. Djerassi, "Kinematics, dynamics and construction of a planarly actuated parallel robot," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 2, pp. 163–172, 1998.
- [241] M. Moradi and A. Fekih, "Adaptive PID-Sliding-Mode Fault-Tolerant Control Approach for Vehicle Suspension Systems Subject to Actuator Faults," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 3, pp. 1041–1054, 2014.
- [242] A. Ghosh, "Scaling Laws," in *Mechanics Over Micro and Nano Scales*. Springer, New York, NY, 2011, pp. 61–94.
- [243] L. Ljung, *Identification: Theory for the user*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.
- [244] ISO:8608-2016, *Mechanical vibration Road surface profiles Reporting of measured data*, 2nd ed. Vernier, Geneva, Switzerland: International Organization for Standardization, 2016.

- [245] F. Tyan, Y.-F. Hong, S.-H. Tu, and W. S. Jeng, "Generation of random road profiles," *Journal of Advanced Engineering*, vol. 4, no. 2, pp. 151–156, 2009.
- [246] M. Agostinacchio, D. Ciampa, and S. Olita, "The vibrations induced by surface irregularities in road pavements - a Matlab approach," *European Transport Research Review*, vol. 6, no. 3, pp. 267–275, 2014.
- [247] ISO:2631-2018, Mechanical vibration and shock Evaluation of human exposure to whole-body vibration Part 5: Method for evaluation of vibration containing multiple shocks, 2nd ed. Vernier, Geneva, Switzerland: International Organization for Standardization, 2018.
- [248] K. Strandemar, "On Objective Measures for Ride Comfort Evaluation," Department of Signals, Sensors and Systems, Stockholm, Sweden, Tech. Rep., 2005.
- [249] I. D. Diaz-Rodriguez and S. P. Bhattacharyya, "PI controller design in the achievable gain-phase margin plane," in 2016 IEEE 55th Conference on Decision and Control, CDC, Las Vegas, NV, USA, 2016, pp. 4919–4924.

