Doc ID	19072301R04
Doc Creation Date	19 Jul 2019
Doc Revision	R04
Doc Revision Date	16 Jan 2020
Doc Status	FINAL

Workpackage	Deliverable ID
WP4, Control Layer Dev Final Report	**D4.9 Control Layer Dev Final Report**

Summary

The document provides a comprehensive description of Control Layer building blocks, developed in the Work-package 4, including implementation aspects on relevant pilot applications, use cases and demonstrators.
In particular, the document is focusing on the description of the BB7 about “Vibration control in mechanically compliant systems”, BB8 about “Robust motion control strategies”, BB9 concerning “Iterative and repetitive control methods” and BB11 “Advanced motion control algorithms and software for predictable multi-many core platforms”.
The purpose of the document is to describe more in detail how the developed technologies fit with the requirements of the several mechatronics applications represented in pilots, use cases and demonstrators. The document provides a general technical introduction of the control strategy. This part of the document is based on the more complete description that have been released with the deliverables of the specific building-block (D4.3, D4.4, D.4.5 and D.4.6).
In addition, and more in detail, the document provides the information about the architecture of the system, i.e. describing input and output of each block, in order to allow the possibility to apply the technology also to an extended range of applications.

Author Davide Colombo

Keywords:
Building Blocks, Vibration control, Robust motion control, Iterative control, repetitive control, motion, multi-core, model-based design, control architecture, industrial application, tracking performance, accuracy, noise reduction.

Coordinator	Sioux CCM
Tel.	0031 (0)40.263.5000
E-Mail	info@i-mech.eu
Internet	www.i-mech.eu

Table of contents

Table of figures

Doc ID	19072301R04
Doc Creation Date	19 Jul 2019
Doc Revision	R04
Doc Revision Date	16 Jan 2020
Doc Status	FINAL

(Open) Issues & Actions

Open Issues (and related actions) that need central attention shall be part of a file called "IAL - Issues & Action List – Partners" which is can be found in the Goolge Drive Partner Zone.

ID	Description	Due date	Owner	IAL ID

Document Revision History

Revision	Status	Date	Author	Description of changes	IAL ID / Review ID
R01	Draft	22-JUL-19	D. Colombo (GEF) L. Simoni (UNIBS) M. Armendia (TEK) M. Goubej (ZAPUNI)	Initiate	
R02	Draft	23-DEC-19	N. Mooren (TUE)	Additional contribution	
R03	Draft	07-JAN-20	D. Colombo, P. Grande (GEF)	Revision before internal review	
R04	Final	16-JAN-20	D. Colombo, P. Grande (GEF)	Final Revision	

Contributors

Revision	Affiliation	Contributor	Description of work
R01	GEF	D. Colombo, P. Grande	Architecture of document, general introduction. Contributions in description of Task 4.4 BB7 and Task 4.6 BB9
	UNIBS	L. Simoni	Description of the implemented strategy on Task 4.4 BB7 and Task 4.6
	TEK	M. Armendia	BB7 Task 4.4 and Task 4.6
	ZAPUNI	M. Goubej	Description of controls task 4.4 and task 4.6
R02	TUE	N. Mooren	Description of controls task 4.5 and BB8
R03	GEF, UNIBS, TUE	D. Colombo, P. Grande, A. Visioli, L.Simoni, D.Goswami, N. Mooren	Internal Revision
R04	GEF	D. Colombo, P. Grande	Final Revision

Document control

	Status	Draft	Draft	Draft	Final				
	Revision	01	02	03	04				
Reviewer Name	**Role**	**Selection**							
Arend-Jan Beltman	Coordinator	X							
Davide Colombo	WP4 leader	X	X	X	X				

File Locations

Via URL with a name that is equal to the document ID, you shall introduce a link to the location (either in Partner Zone or CIRCABC)

URL	Filename	Date
		dd-MMM-yyyy

Literature

Ref	Name	Publisher	Year
[1]	D4.3 Vibration control module (BB 7)		2019
[2]	D4.4 Robust multivariable control module (BB 8)		2019
[3]	D4.5 Iterative and repetitive control module (BB 9)		2019
[4]	D4.6 Advanced software support for predictable motion control on multi/many-core platforms		2019
[5]	Full Bibliography in Section 8.		

Abbreviations & Definitions

Abbreviation	Description
BB	Building Block
WP	Work Package

Definition	Description

1 Introduction

This document provides a comprehensive description of the Building Blocks developed in the framework of the Work Package 4 (Control Layer design and development). In this document, the Control Layer Building Blocks (BB7, BB8, BB9, and BB11) are shown and their usage is briefly explained. Furthermore, some implementation aspects of the Control Layer Building Blocks on relevant pilots, use cases and demonstrators' applications are provided.

This document is not a complete and detailed documentation of the Control Layer Building Blocks, but, on the contrary, it has the aim to be a general explanation on how the Control Layer Building Blocks can be used and on how they can be implemented in some specific environments (that are the I-MECH Pilots, Use Cases and Demonstrators).
For a detailed explanation of the algorithms used inside the Control Layer Building Blocks and of the methodologies related to the usage of the Control Layer Building Blocks, or some parts of them, the reader has to consider the Building Blocks documentation and the specific deliverables of the building blocks.

The document provides a description of how the Building Blocks developed in the framework of the Work Package 4 (in the first two and a half year of the I-MECH project) can be used.
The Control Layer Building Blocks developed in the framework of WP4 and considered in this deliverable are:

- BB7 - Vibration control module
- BB8 - Robust model-based multivariable control
- BB9 - Iterative and repetitive control module
- BB11 - RTOS for multi-many core platform.

These Building Blocks are strictly connected to the tasks in which they have been developed, that are:

- BB7 - Task 4.4
- BB8 - Task 4.5
- BB9 - Task 4.6
- BB11 - Task 4.7

The Control Layer Building Blocks are mainly located in Layer 2 of the I-MECH platform (see Figure 1) and they can be used separately, or together, in order to obtain an enhanced performance of the mechatronic systems to control (namely the controlled plant).

In this document, a generic description of how the Control Layer Building Blocks have been developed is shown, and a brief description of the methodologies that have been adopted to implement the Control Layer Building Blocks is provided. Furthermore, a short summary of the usage of these Building Blocks is reported.
For all the Control Layer Building Block, a description of the used implementation method is given, followed by the description of the inputs, the outputs, the constants and the parameters that are mandatory to use the different parts of the Building Blocks.

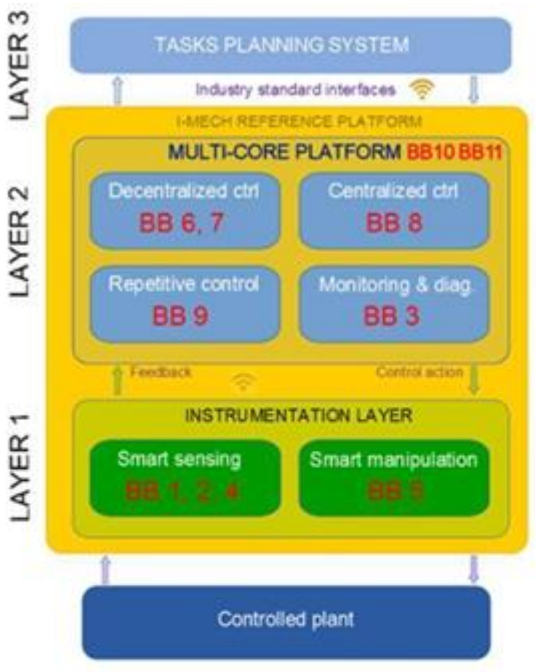


Figure 1: Building Blocks and Layers architecture

The steps that have been followed to develop the Control Layer Building Block (namely, MIL, PIL, SIL, HIL) before to test it on the Pilots, Use Cases and Demonstrators are then described.

2 System Behavior Design and Interfaces

The Building Blocks developed in the framework of the Work Package 4 are located in Layer 2 of the I-MECH architecture. It means that the System Behaviour Layer, that is the Layer 3 of the I-MECH architecture, is not involved in the BBs developed inside the Work Package 4. Nonetheless, the BBs of the WP4 have been developed in order to be integrated inside the I-MECH architecture, and it means that they comply with the architecture and the interfaces described in deliverable D5.5.

2.1 Typical building block component

In this deliverable, the behavior of the Building Blocks related to the Work Package 4 of the I-MECH project is explained, together with the interfaces that allow the Building Blocks to be configured and used properly.

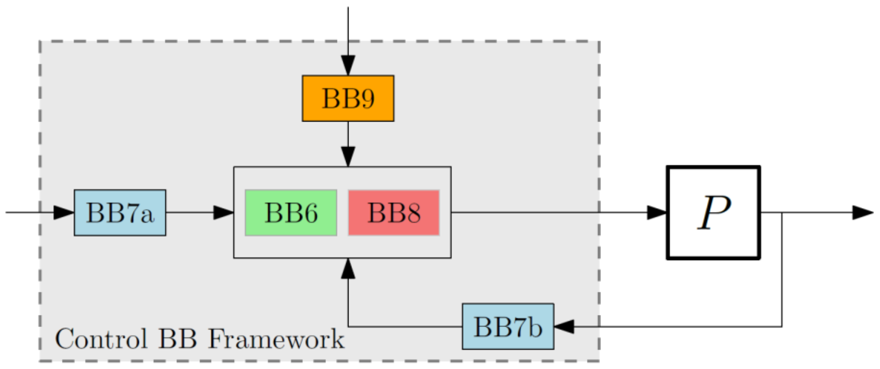


Figure 2: Control BBs interconnection

Figure 2 shows the interconnection between the control Building Blocks, where the Control BB Framework is represented by BB11.

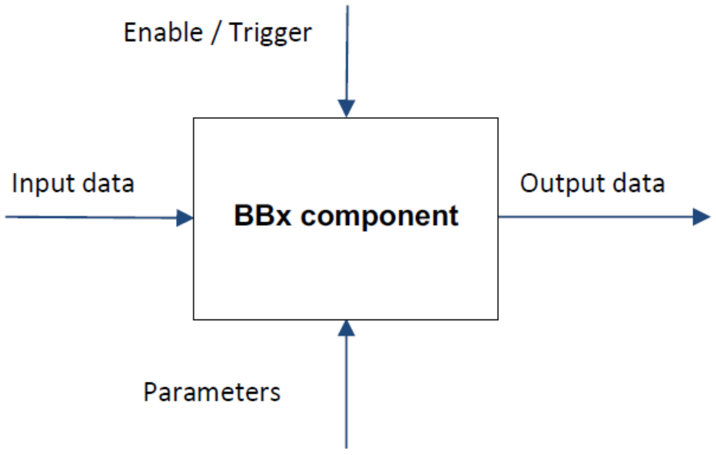


Figure 3: Structure of the algorithmic control BB (BB7, BB8, BB9)

With the exception of the BB11, the generic structure of the Building Blocks developed within the Work Package 4 is shown in Figure 3, where it is possible to see that the generic Building Block has 3 inputs data types (Enable/Trigger, Input Data, and Parameters) and 1 output data type (Output Data).

2.2 How to operate with blocks

The Building Blocks have been designed as standalone MATLAB/Simulink blocks, it means that each block can be used separately from the others. They can be translated into C/C++ functions or PLC standard function using MATLAB/Simulink coders. Thus, they can be used both in greenfield and brownfield contexts. The user can modify the I/O structure in Simulink according to implementation needs, nevertheless, it is strongly suggested to keep the developed I/O structure and to use interface blocks to adapt the needed structure to the developed ones.
In any case, a user’s guide will be available for every BBs. This guarantees the interoperability of the Building Blocks.

2.3 Communication Interfaces

The Building Blocks developed in the framework of the Work Package 4 are located in Layer 2 of the I-MECH architecture, so, they can communicate between all the Building Blocks of the same layer. The communication between layers is described in Deliverable 5.5 and the BBs have been built in order to comply with the interfaces implemented in the I-MECH platform.

3 BB7 - Task 4.4 Vibration control in mechanically compliant systems

3.1 Functionalities

3.1.1 Anti-sway techniques for overhead cranes - Input shaping techniques (UNIBS/GEF)

3.1.1.1 General description - algorithm theory

One of the most known open loop techniques for the anti-sway control of oscillation systems is called Input-Shaping (IS).
For the case of overhead cranes, IS control can be schematized as shown in Figure 4.

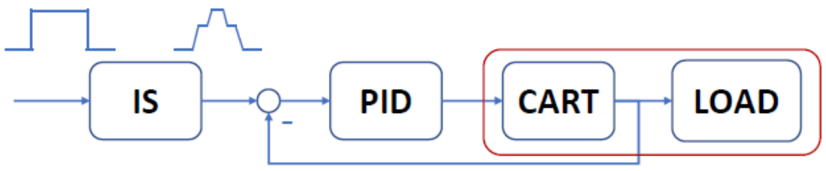


Figure 4: Input Shaping Control

Basically, the Input Shaping technique consists in a proper modification of a generic set point in order to obtain zero oscillations of the load.
In the framework of the I-MECH project, seven different Input Shaping techniques have been implemented and tested.
The seven different Input Shaping techniques are:

- ZV - zero vibrations,
- ZVD - zero vibrations and derivative,
- ZVDD - zero vibrations and second derivative,
- EI - extra insensitive,
- EI2H - extra insensitive 2 humps,
- UM - unitary module,
- PS - partial sum.

Each technique differs from the others in the degree of robustness and in the velocity of application.
It is important to remark that in the case of multiple pendulums the Input Shaping blocks can be connected in series to take into account all the different oscillation modes of the system.

3.1.1.2 Implementation in Matlab-Simulink C

The different Input-Shaping techniques have been implemented in Matlab/Simulink (see Figure 5).

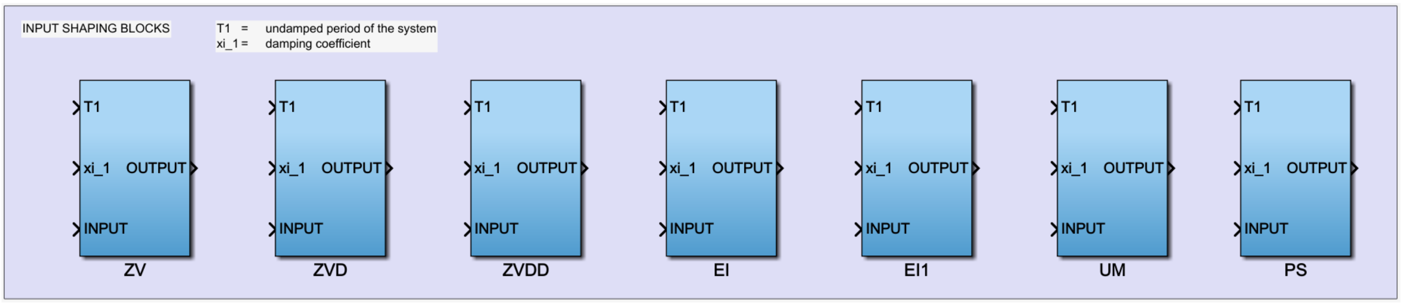

INPUT SHAPING BLOCKS
T1 = undamped period of the system
xi_1 = damping coefficient
T1
xi_1
OUTPUT
INPUT
ZV
ZVD
ZVDD
EI
EI1
UM
PS

Figure 5: Input Shaping Simulink Blocks

After the validation the blocks have been exported in FMU in order to be used in different simulation environments.

3.1.1.3 Inputs, outputs, parameters, constants

All the Input Shaping techniques share the same layout.
In this section a brief description on inputs, outputs, parameters and constants is shown. More details are available in the function help.

Input Shaping Blocks	
Inputs, Parameters and Constants	
Parameter	Description
T1	Undamped period of the system [s]
xi_1	Damping coefficient
Input	Input set point signal
Outputs	
Parameter	Description
Output	Output set point signal

3.1.1.4 Validation in MIL, PIL, HIL

The validation of the 7 Input Shaping techniques has been done in MIL by using Matlab/Simulink.
Then, the techniques have been validated by using Simulink-Multibody and in co-simulation Simulink-Amesim.
In Figure 6 the example of the Simulink scheme for the MIL validation of the Input Shaping ZV technique is shown.

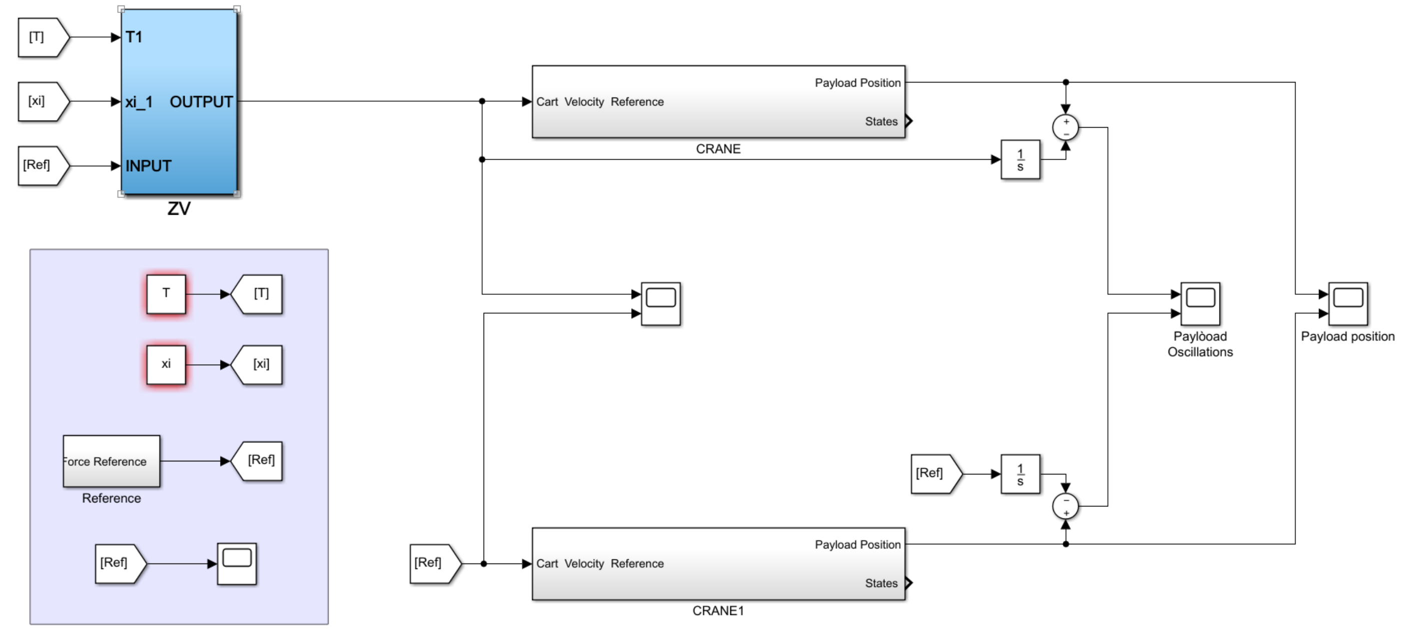


Figure 6: Input-Shaping MIL validation example

Each Input Shaping techniques have then been validated in PIL and HIL on the Use Case 1.1; details of the validation can be found in Deliverable 6.5.

3.1.2 Anti-sway techniques for overhead cranes - Input/Output inversion techniques (UNIBS/GEF)

3.1.2.1 General description - algorithm theory

An alternative open loop method for the anti-sway control of overhead crane is the so-called Input/Output Inversion technique, described in [1].
Basically, the Input/Output Inversion technique consists in the proper computation of a set point, based on the model inversion, in order to obtain zero oscillations of the load (see Figure 7). Differently from the IS technique, in the Input/Output Inversion technique the computation of the proper set point is based on the inversion of the model between the cart position and the load position.

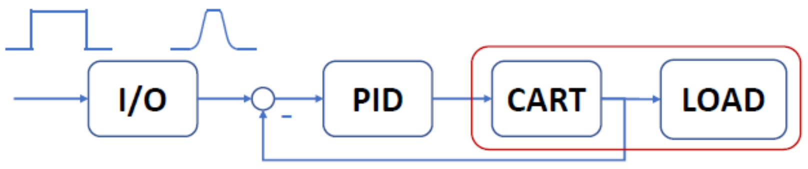


Figure 7: Input/Output inversion control.

3.1.2.2 Implementation in Matlab-Simulink C

The different Input/Output inversion techniques has been implemented in Matlab/Simulink (see Figure 8).

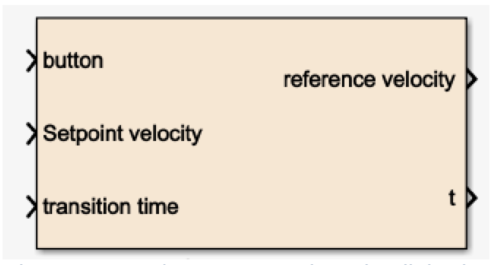


Figure 8: Input/Output Inversion Simulink Block.

After the validation the block have been exported in FMU in order to be used in different simulation environments.

3.1.2.3 Input, output, parameters, constants

In this section a brief description on inputs, outputs, parameters and constants is shown. More details are available in the function help..

Input/Output Inversion Blocks	
Inputs, Parameters and Constants	
Parameter	Description
Button	Start and stop of the motion [0-1]
Setpoint velocity	Maximum velocity setpoint
transition time	Maximum transition time to reach the setpoint velocity [s]
Outputs	
Parameter	Description
reference velocity	Output reference velocity
t	I/O inversion time [s] for debug

3.1.2.4 Validation in MIL, PIL, HIL

The validation of the Input/Output Inversion technique have been done in MIL by using Matlab/Simulink.
Then, the technique has been validated by using Simulink-Multibody and in co-simulation Simulink-Amesim.
Figure 9 shows the example of the Simulink scheme for the MIL validation of the Input/Output Inversion technique.

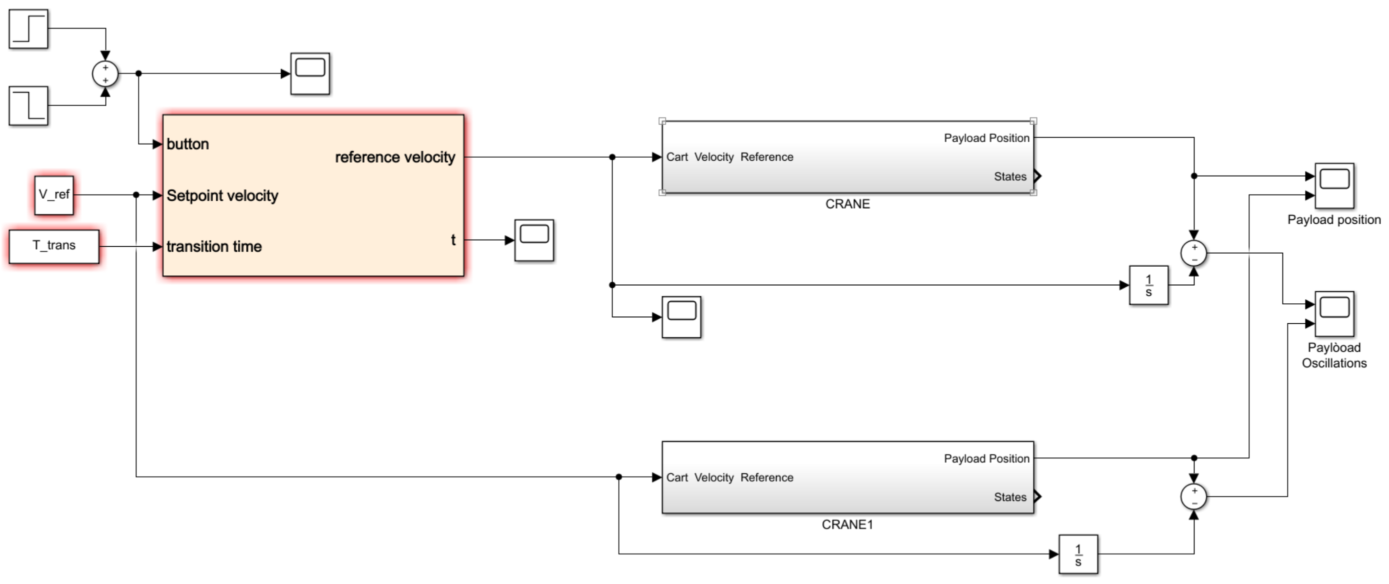


Figure 9: Input-Output inversion MIL validation example

The Input/Output Inversion technique has then validated in PIL and HIL on the Use Case 1.1.

3.1.3 Anti-sway techniques for overhead cranes - MPC based techniques (UNIBS/GEF)

3.1.3.1 General description - algorithm theory

Advanced closed loop methods for the anti-sway control of overhead crane are the so-called MPC based techniques, which can take into account the system constraints.
Standard linear MPC can give good results when applied to cranes, by taking into account the presence of the operator, as shown in [2].
A varying cable length, which is a common maneuver for industrial cranes, requires the development of more complex methods (see [3]).

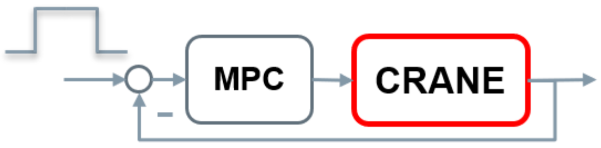


Figure 10: Model predictive control scheme.

Depending on the system to be controlled, (e.g. if a velocity control loop is already in place and the controller tuned), two different approaches can be used: controlling the crane by acting directly on the torque of the motor (see Figure 10) or by setting the cart velocity as the input of the system, relying on a closed velocity control loop for the trajectory tracking of the cart [4] (see Figure 11).

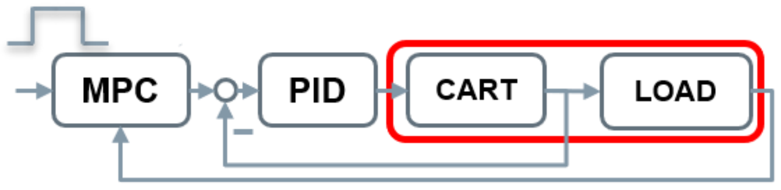


Figure 11: MPC-PID control scheme.

3.1.3.2 Implementation in Matlab-Simulink C

The different MPC techniques have been implemented in Matlab/Simulink, Figure 12 shows the Simulink blocks of the developed MPC based techniques.

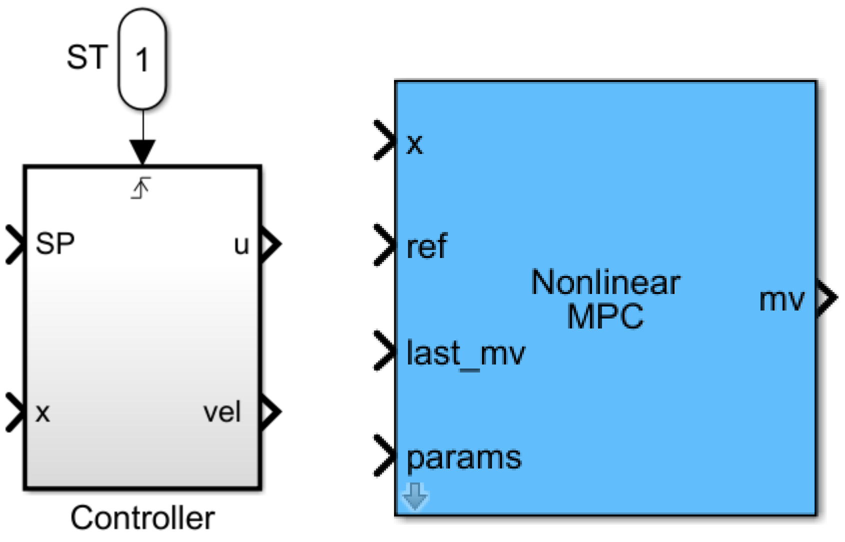


Figure 12: MPC Techniques Simulink Blocks

The blocks shown in Figure 12 are the linear and nonlinear MPC controllers respectively. After the validation the block have been exported in FMU in order to be used in different simulation environments.

3.1.3.3 Input output parameters, constants

In this section a brief description on inputs, outputs, parameters and constants is shown. More details are available in the function help.

Linear MPC Block	
Inputs, Parameters and Constants	
Parameter	Description
ST	Control sampling period
SP	Setpoint
x	States of the system
Outputs	
Parameter	Description
u	torque on the motor
vel	velocity of the cart, in case of MPC+PID approach

Nonlinear MPC Block	
Inputs, Parameters and Constants	
Parameter	Description
last MV	last MV of the block
ref	Setpoint
x	States of the system
params	parameters (e.g. control sampling time)
Outputs	
Parameter	Description
MV	torque on the motor

3.1.3.4 Validation in MIL, PIL, HIL

The validation of the MPC technique has been done in MIL by using Matlab/Simulink. Then, the technique has been validated by using Simulink-Multibody and in co-simulation Simulink-Amesim.
In Figure 13 the example of the Simulink scheme for the MIL validation of the Input/Output Inversion technique is shown.

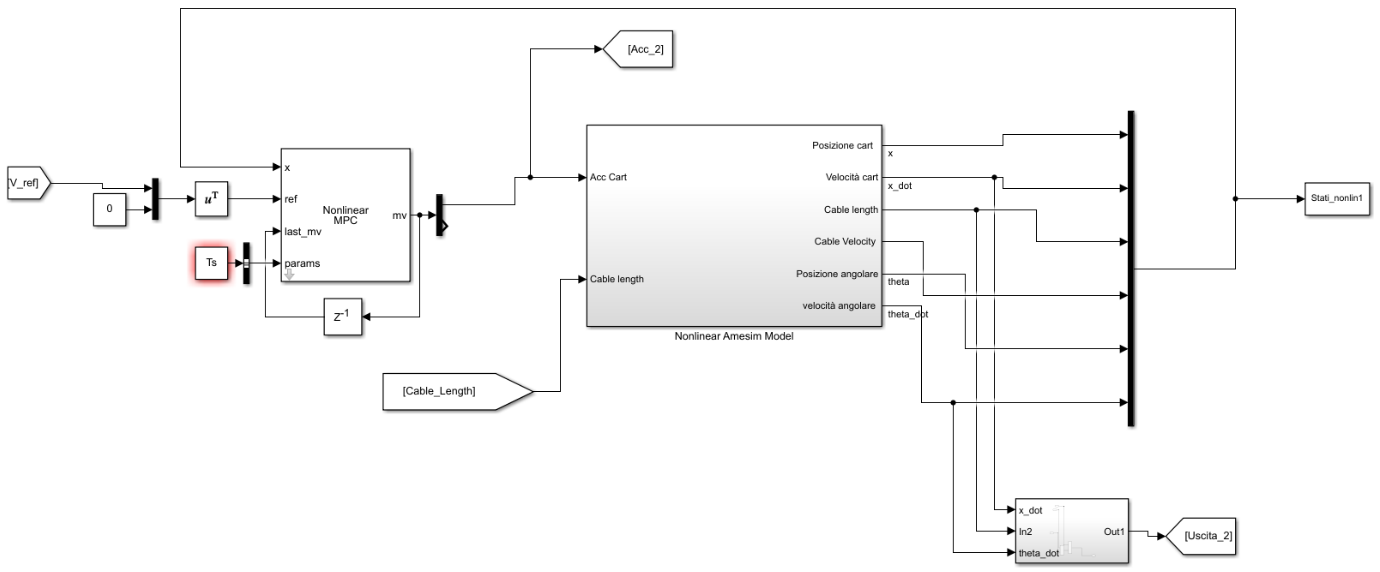


Figure 13: MPC technique MIL validation example

The MPC technique has then been validated in PIL and HIL on the Use Case 1.1.

3.1.4 New design methods for Zero Vibration input shapers (ZAPUNI)

3.1.4.1 General description - algorithm theory

3.1.4.1.1 Generic shaping/smoothing filter design tools

New design methods for optimal synthesis of input shaping filters were developed by ZAPUNI and implemented in a simple graphical user interface. It allows formulation of various design constraints, validation of important filter characteristics and combination of multiple filters using discrete convolution. Result of the design can be then exported to a .csv file in the form of filter impulse function or a C-code can be automatically generated from the functional block in the Matlab/Simulink environment. This allows its utilization in arbitrary real-time control or signal-processing platform.

The developed methods interfaced through the GUI can serve for the design of generic FIR type discrete-time filter with the transfer function in the Z-transform given in as:

$$H(z) = \frac{y(z)}{u(z)} = \sum_{i=1}^{n} A_i z^{-t_i/T_s},$$

where t denotes integer multiples of the sampling period T_s and the filter dynamics fully determined by its impulse function

$$\mathbf{h} \triangleq [A_1, A_2, \dots, A_n]^T.$$

The developed SW deals with the tasks of design constraints specification, shaping filter synthesis and implementation. It relies on the Zero Vibration damping techniques which can be used to eliminate or substantially reduce unwanted residual oscillations arising during highly-dynamic maneuvers with mechanically compliant systems. The shaping filter can be implemented as a reference shaping feedforward filter, or embedded in the feedback loop as shown in Figure 14.

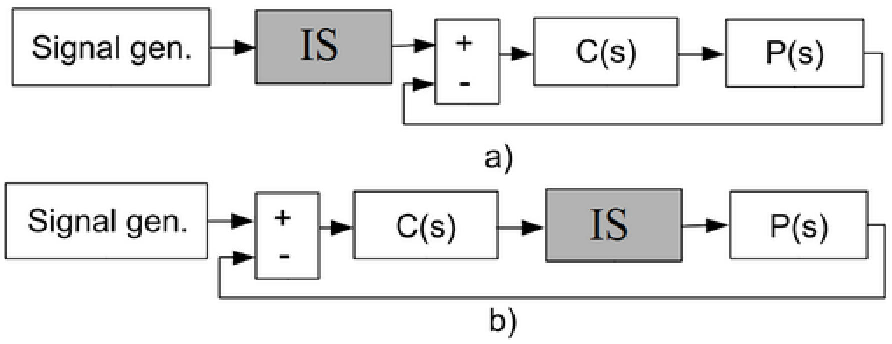
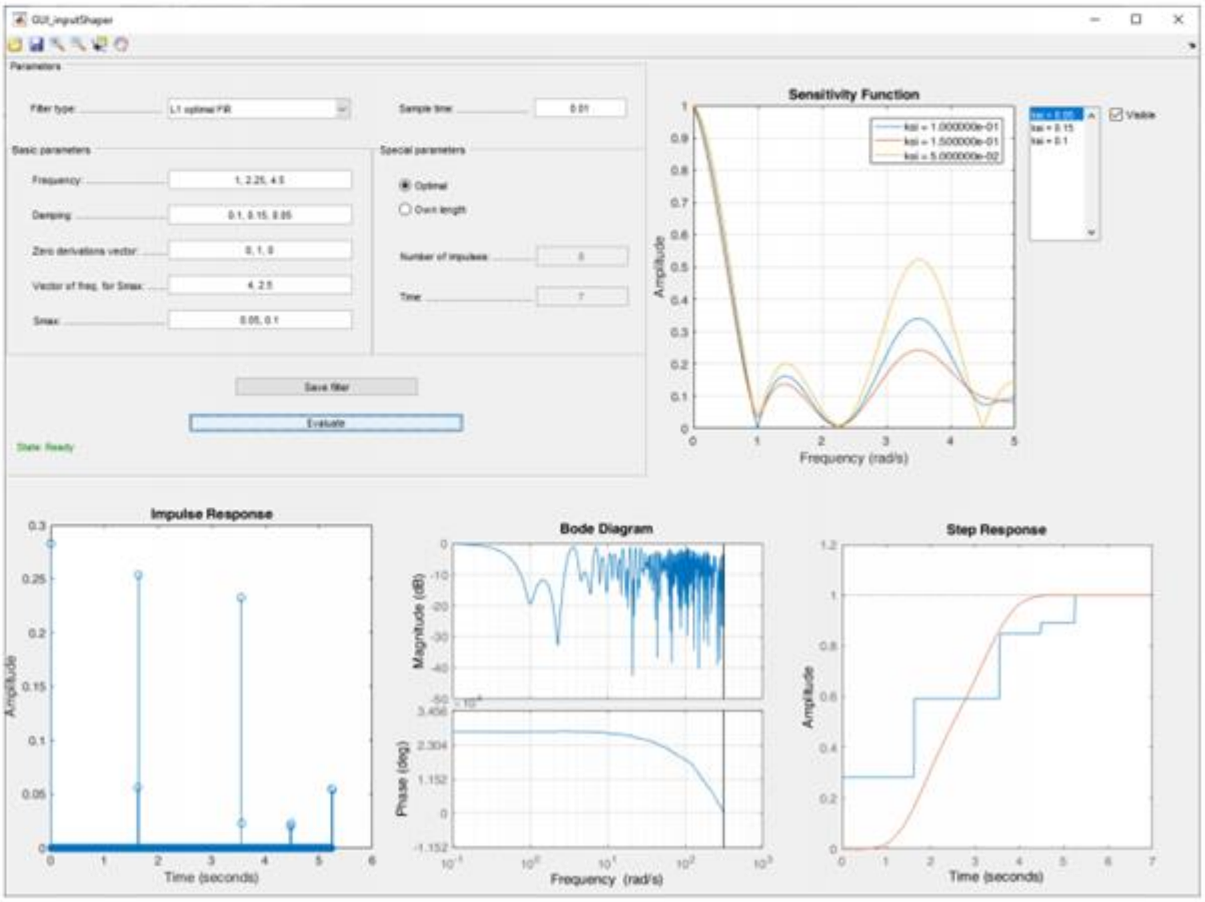


Figure 14: Possible ways of shaping filter integration, a) reference command shaper, b) feedback command shaper

The main advantage of Zero Vibration shapers is that they can be designed using a limited information about the plant dynamics model. Only approximate location of the resonance frequencies is necessary compared to full plant model usually required in traditional feedforward control strategies. Robust design methods are embedded to allow the user to directly specify the level of tolerance to modelling errors and choose a suitable robustness vs induced delay tradeoff.


GUI_inputShaper

Parameters

Filter type: L1 optimal FIR
Sample time: 0.01

Basic parameters
Frequency: 1, 2.25, 4.5
Damping: 0.1, 0.15, 0.05
Zero derivations vector: 0, 1, 0
Vector of freq. for Smax: 4, 2.5
Smax: 0.05, 0.1

Special parameters
[✓ Optimal]
Own length
Number of impulses: 5
Time: 7

State: Ready

ksi = 1.000000e-01
ksi = 1.500000e-01
ksi = 5.000000e-02

ksi = 0.05
ksi = 0.15
ksi = 0.1
[✓ Visible]

Figure 15: Zero Vibration Filter Design Tool, GUI for the shaping filter design methods

The graphical user interface (see Figure 15) consists of three main cooperating modules:

- **Graphical interface** – serves for definition of design constraints, results validation, import and export of user data
- **Computation core** – a set of design methods realizing the shaping filter synthesis algorithms based on the user-specified design constraints
- **Functional block** for a real-time implementation – generic discrete FIR filter in the form suitable for implementation in arbitrary sampled data control system

The graphical user interface was designed at ZAPUNI in former R&D projects. The application was substantially extended in terms of I-MECH by adding new optimization-based design methods allowing to synthesize new classes of input shaping filters. The common denominator of the implemented method is a robust design approach allowing the user to specify a desired level of tolerance to modelling errors. This is the main difference to conventional shaping filters known from the literature which typically do not allow fine-tuning of the filter characteristics. The implemented methods cover most of the known shapers for single resonance systems as a special case.

The list of implemented methods:

- ZV, ZVD, ZVDD, EI, THEI, MISZV – standard algorithms known from the literature for single dominant resonance systems
- ZV4IS – novel method developed at ZAPUNI allowing a generic parameterization of all input shaping filters for a single resonance system with a minimum nonredundant parameterization using two user-specified design parameters with a clear physical meaning. All the conventional filters listed above are particular cases of the given generic parameterization.
- L1 optimal filter – generic shaping filter suitable for arbitrary number of resonance modes. Optimization-based procedure searches for sparse-structure filters with a minimum number of nonzero elements which may be beneficial for real-time implementation.
- H2 optimal filter – novel method based on quadratic optimization allowing to deliver generic shaping/smoothing filters for arbitrary number of resonances. The chosen cost function leads to filters with a minimum energy of the impulse response, which is beneficial for the generation of smooth trajectories easily followable by feedback loops and tends to reduce energy consumption
- Weighted H2 optimal filter – novel method based on the modification approach by adding a user-specified weighting allowing to gain further control over the dynamics of the shaping filter, this can serve for the design of joint shaping & smoothing filters in an optimal manner, it is possible to e.g. enforce a defined high-frequency roll-off in the smoother transition band
- Generic FIR low-pass/high-pass/band-pass/band-stop filter – implementation of state-of-the-art methods for the synthesis of general-purpose filters which can be used separately or in conjunction with the above mentioned Zero vibration shaping filters
- Sequential multi-level filters – combination of various filter types to a single FIR system

3.1.4.1.2 Optimal input shaping for gantry cranes systems

Specific shaping filter designs were formulated in terms of the development of anti-sway system for human-operated gantry cranes in use-case UC1.1 (see Figure 16). The general idea is to use a shaping filter to modify the setpoint commands generated by the human operator and use this modified signal as a velocity or position reference for the existing motor feedback loops. Novel design method employs convex optimization to deliver shaping filters which combine robustness and simplicity of implementation known from the ZV filtering theory and precise constraints definition known from model-predictive control field. This allows utilizing the key advantages from both methods in terms of performance and robustness.

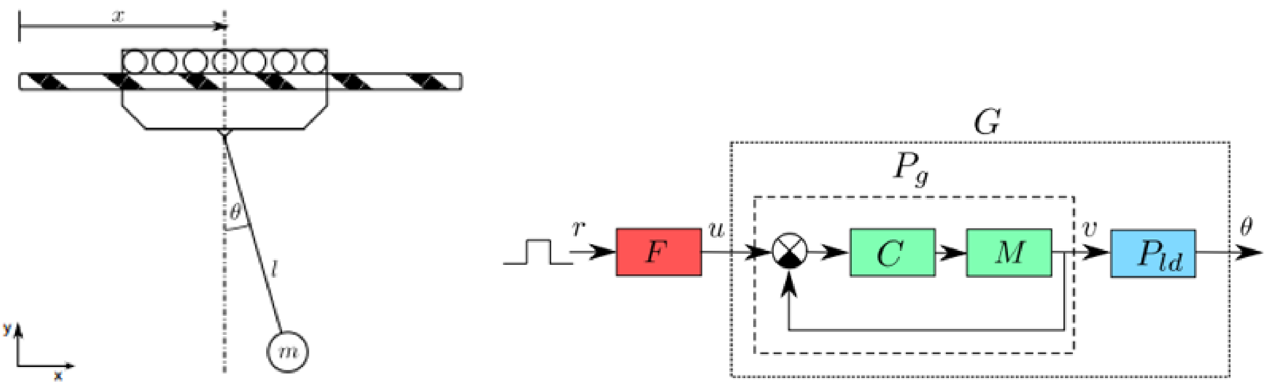


Figure 16: Optimal control for gantry crane systems – principle of operation

More details about the algorithms and theory behind the design methods can be found in the I-MECH deliverable D4.3. Novel theoretical results were presented in [5]-[9].

3.1.4.2 Implementation in Matlab-Simulink C

The shaping filter design methods were embedded in the developed graphical user interface. It is available in the form of a Matlab application or as a stand-alone software which can run on any PC with Windows operating system (see Figure 17). The designed shaping filters can be exported in the form of a .csv file containing resulting shaper impulse function. The user can use it to transfer the shaping filter on an arbitrary real-time control platform. Alternatively, C-code can be automatically generated from the Matlab/Simulink scheme containing the standard Discrete FIR filter functional block. The individual filter design algorithms are also available in the form of C-MEX functions for Matlab environment which can be accessed from the command line without the necessity of utilization of the graphical user interface.

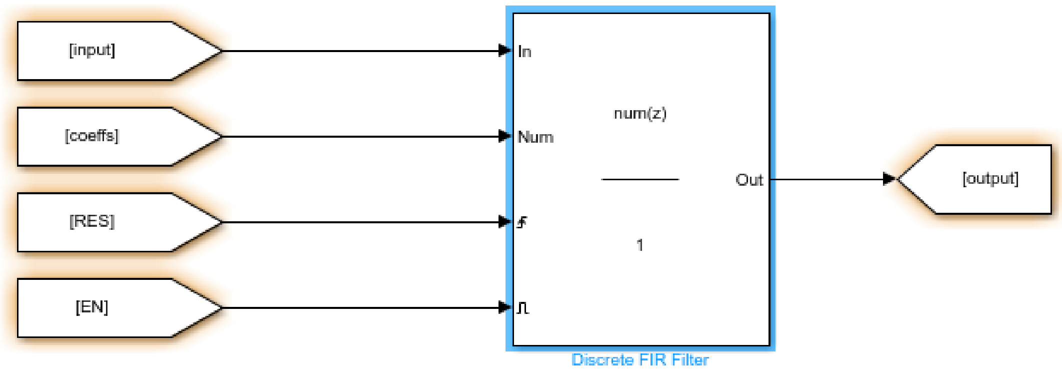


Figure 17: Shaping Filter Simulink Block.

3.1.4.3 Inputs, outputs, parameters, constants

General purpose/shaping/smoothing FIR filter	
Inputs, Parameters and Constants	
Parameter	Description
In	input signal to be processed by the filter
Num	shaper impuse function, derived from the design methods using GUI/command line interface
RES	External reset of the filter on the rising edge
EN	Enable filter
Outputs	
Parameter	Description
Out	Output filtered signal

Finite horizon optimal command shaping for gantry crane control	
Syntax: coeffs=fhocs(om,ksi,G,amax,vmax,jmax,phimax,len,vcom,Q,R)	
Inputs, Parameters and Constants	
Parameter	Description
G	Structure describing the dynamics of the internal gantry velocity control loop
om	Vector of eigen-frequencies of the load subsystem
ksi	Vector of relative damping coefficients corresponding to the frequencies in om
amax	(optional) Maximum allowed gantry acceleration
vmax	(optional) Maximum allowed gantry velocity
jmax	(optional) Maximum allowed gantry jerk
phimax	(optional) Maximum allowed load-sway angle during transients
len	Length of the filter in seconds, setting len=0 leads to a search for a time-optimal solution fulfilling all the defined constraints
vcom	Amplitude of the velocity command provided by the human operator
Q	(optional) User specified quadratic weighting of the tracking error during transient maneuvre

R	(optional) User specified quadratic weighting of the control effort
Outputs	
Parameter	Description
coeffs	Impulse response coefficients of the synthesized shaping filter

3.1.4.4 Validation in MIL, PIL, HIL

Validation has been done in HIL on UC1.1. Validation on UC1.3 is ongoing. The proposed methods can be used in various mechatronic applications, typical application domains include

- Robotic manipulators
- Machine tools
- Single-purpose machines
- High-precision servo systems
- Gantry crane systems.

3.1.5 Acceleration Feedback (FAG/TEK)

3.1.5.1 General description - algorithm theory

The acceleration feedback strategy is an additional feedback loop which modifies the current setpoint with the signal provided by the accelerometer as shown in Figure 18. Since the accelerometer is placed in the TCP, the main aim should not be to cancel the acceleration of this point but to cancel the difference of accelerations between the acceleration desired according to the reference and the acceleration measured. The main advantage of this strategy is that the acceleration measured by the accelerometer can be used without any transformation. Just a low pass filter may be used depending on the level of noise introduced in the measure [10].

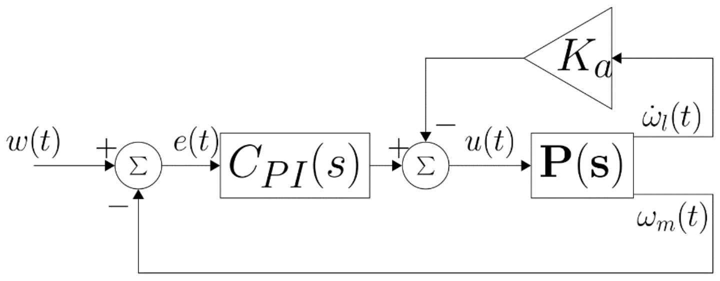


Figure 18: Acceleration Feedback control scheme.

3.1.5.2 Implementation in Matlab-Simulink C

The acceleration feedback technique has been implemented in Matlab/Simulink (see Figure 19).

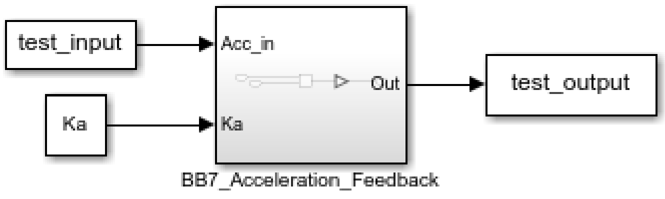


Figure 19: Acceleration Feedback Simulink Block.

After the validation the block have been exported in FMU in order to be used in different simulation environments.

3.1.5.3 Inputs, outputs, parameters, constants

Inputs, outputs, parameters and constants of the acceleration feedback technique.
In this section a brief description on inputs, outputs, parameters and constants is shown. More details are available in the function help.

MPC Blocks	
Inputs, Parameters and Constants	
Parameter	Description
Acc_in	Acceleration signal
Ka	Acceleration feedback gain
Outputs	
Parameter	Description
Fout	Force signal (to be fed in the torque FFW input of reference controller)

3.1.5.4 Validation in MIL, PIL, HIL

The validation of the acceleration feedback techniques has been done in MIL by using Matlab/Simulink and considering Use Case 1.2 as a reference.

Figure 20 presents the general view of the developed model. It must be highlighted that two versions of the mechanical section of the model has been tested, the first one using Simulink-Multibody (Simscape) and the second one using Amesim.

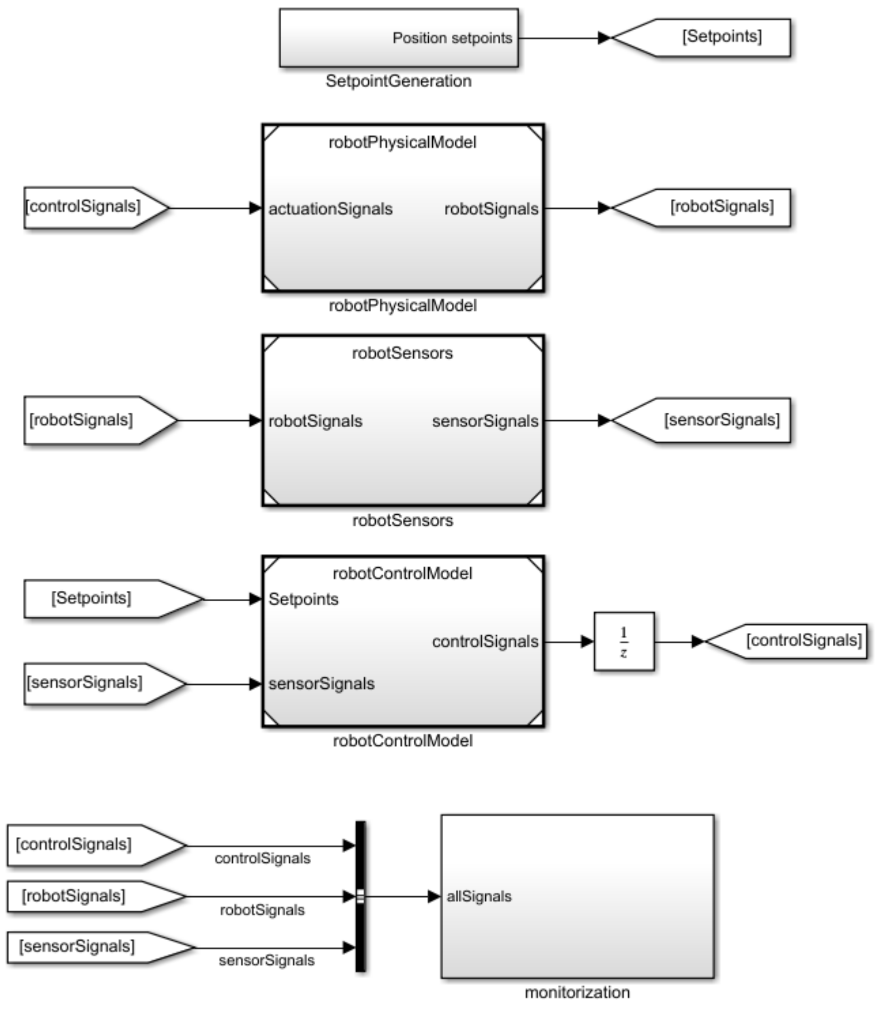


Figure 20: Acceleration Feedback MIL validation example.

Figure 21 presents the content of the control block of the upper diagram. There, it can be observed how the outputs of the acceleration feedback blocks are directly fed to the correspondent input of the reference controllers of the respective joint of the robot.

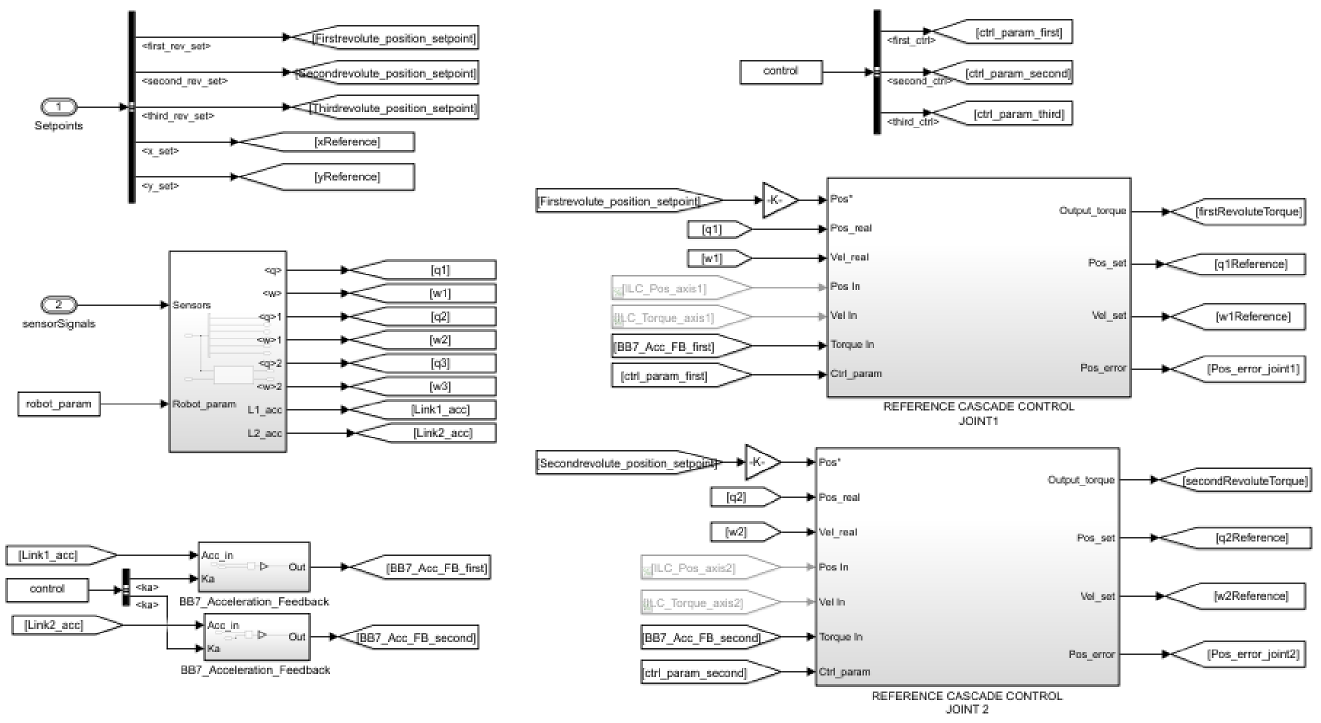

Setpoints
sensorSignals
robot_param
control
REFERENCE CASCADE CONTROL JOINT1
REFERENCE CASCADE CONTROL JOINT 2
BB7_Acceleration_Feedback

Figure 21: Acceleration Feedback MIL detailed validation example.

Validation has been done in HIL on UC1.2.

3.1.6 H-infinity optimization approach (ZAPUNI)

3.1.6.1 General description - algorithm theory

This functionality provides a design method for an optimal coordinated synthesis of the acceleration feedback structure shown in the previous section. The goal is to derive suitable parameters for both velocity/position feedback controller which is assumed to be of a PID type and the auxiliary acceleration feedback taking the measurement from the load-attached accelerometer providing additional plant output. This strategy allows to significantly improve the achievable quality of control for mechanically compliant systems which are difficult to control using a conventional non-collocated feedback solely from the motor side.

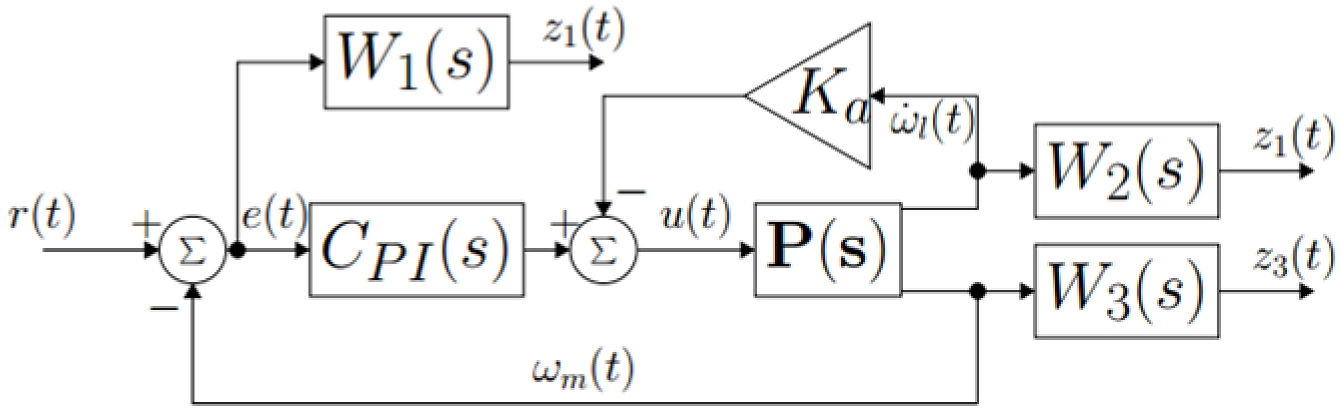


Figure 22: Weighted H-infinity optimization for the synthesis of both feedback control and auxiliary acceleration feedback

The optimization problem is formed as a weighted H-infinity minimization

$$\|H\|_\infty = \left\| \begin{bmatrix} W_1(s)S_2^{ma}(s) \\ W_2(s)\cdot s\cdot T_2^{la}(s) \\ W_3(s)T_2^{ma}(s) \end{bmatrix} \right\|_\infty \leq \gamma,$$

with the user-specified weights W1, W2 and W3 allowing to formulate loop-shaping inequalities based on the frequency domain requirements (see Figure 22).
Details about the design methods and used algorithms can be found in the I-MECH deliverable D4.3. Novel theoretical findings and experimental results are presented in [11], [12].

3.1.6.2 Implementation in Matlab-Simulink C

The design method is accessible in the form of C-MEX function accessible from the Matlab environment. The result of the controller synthesis are the parameters of the PID feedback controller and the acceleration feedback. They can be used on a target platform to parameterize an existing control structure. Alternatively, C-code can be generated directly from the Matlab-Simulink environment using the Simulink Coder.

3.1.6.3 Inputs, outputs, parameters, constants

Optimal design of Acceleration feedback-based controller	
Syntax: [C,A,F]=pidacc(P,W1,W2,W3,afbmode,ka,abw)	
Inputs, Parameters and Constants	
Parameter	Description
P	Structure containing identified plant model, generally a single-input-two-outputs system describing the dynamics from the motor torque/force to motor-side velocity/position and load-side acceleration
W1	User-specified weighting function for the definition of motor-side closed-loop performance in terms of desired bandwidth, disturbance rejection in terms of low-frequency attenuation and robustness in stability in terms of maximum sensitivity peak
W2	User-specified weighting function for the definition of the load-side closed-loop performance in terms of the maximum load-side complementary sensitivity peak

W3	User-specified weighting function for the definition of closed-loop bandwidth. It can be used to penalize high control activity to find a suitable performance trade-off
afbmode	(optional) Mode of acceleration feedback tuning, 0 – automatic acceleration gain derivation, 1 – manual user-specified gain for controller fine-tuning
ka	(optional) Manual specification of the acceleration gain for afbmode=0
lpf	(optional) Low-pass shaping filter selection flag, 0 – tune PI(D) gains only, 1 – include additional low-pass filter to improve high-frequency roll-off
Outputs	
Parameter	Description
C	PI(D) controller in the motor loop
A	Acceleration feedback part of the controller
F	Additional low pass shaping filter

3.1.6.4 Validation in MIL, PIL, HIL

Validation has been done in HIL on ZAPUNI Flexible manipulator setup. Detailed explanation of the proposed approach can be found in I-MECH deliverable D4.3. New theoretical findings and experimental results are presented in [13].

3.2 Implementation aspects

3.2.1 Use Case 1.1

3.2.1.1 Anti-sway techniques for overhead cranes - Input shaping techniques (UNIBS/GEF)

The input shaping filters have been implemented in UC1.1 in ST language (IEC 61131.3 Structured Text). The Function Block containing the input shaping filters can be easily added to already existing Gefran control structures. Due to the small amount of memory, the Input shaping length has been kept fixed and equal to 200 samples so that the user just need to select the desired shaper filter, and to insert the proper IS inputs (Undamped period of the system [s], Damping coefficient and Input set point signal).

3.2.1.2 Anti-sway techniques for overhead cranes - Input/Output inversion techniques (UNIBS/GEF)

The input/output inversion technique has been implemented in UC1.1 in ST language (IEC 61131.3 Structured Text) The input/output inversion technique does not present memory limitation on Gefran Drives. The user must insert just the inputs and parameters described in the Input/Output inversion techniques section of this deliverable.

3.2.1.3 Anti-sway techniques for overhead cranes - MPC Based techniques (UNIBS/GEF)

The MPC Based techniques have been implemented in UC1.1 in ST language (IEC 61131.3 Structured Text).
The sampling time for the MPC strategy considered in the Gefran Drive is 24/32 [ms]. At the moment of this Deliverable Contribution, the MPC technique has been tested on the HIL at the University of Brescia. Due to the fact that the Drives used in the HIL are the same as the ones used in the UC1.1, the implementation aspects do not change.

3.2.2 Use Case 1.2

3.2.2.1 Acceleration Feedback (FAG/TEK)

The acceleration feedback functionality has been implemented in UC1.2 in C-code, thanks to a special option of FAGOR CNCs that allows including additional features to the control loops. The algorithm runs at position loop sampling rate, in this case 2 ms.
Accelerometer signal is directly read as an internal variable by the FAGOR CNC thanks to the inputs of the used drives.

3.2.3 Use Case 1.3

3.2.3.1 New design methods for Zero Vibration input shapers (ZAPUNI)

The input shaping filters designed using the developed SW tools were implemented in a functional blocks realizing a generic discrete FIR filter in the REXYGEN framework which is compatible with the PLC controllers produced by TECO company. They can be executed with an arbitrary update rate with typical values ranging from 100 microseconds to several milliseconds based on a particular dynamics of the controlled plant. They can serve as a reference shaping filter in the feedforward manner or inserted directly in the feedback loop to extend conventional velocity or position controller.

3.2.3.2 H-infinity optimization approach (ZAPUNI)

The proposed control structure with the augmented controller containing the acceleration feedback part was implemented in a functional blocks realizing the whole control structure in the REXYGEN framework which is compatible with the PLC controllers produced by TECO company. As in the previous case, they can be executed with an arbitrary sampling rate. The actual value of acceleration feedback is provided by a MEMS sensor connected to analog input module of the controller. The signal is fed to the controller structure after A/D conversion and proper preprocessing.

4 BB8 - Task 4.5 Robust motion control strategies

4.1 Functionalities

In this section a brief description of the general theory and most important tuning knobs behind functionalities in BB8 are provided, refer to D4.4 for more details. Finally, the validation of the functionalities on either the pilots or other benchmark systems is provided.
In the remainder of this section the functionalities are outlined. Firstly, system identification is described that forms the basis for all controller design procedures. Second, controller design methods that allow to design a controller on the basis of a measured frequency response function are discussion. Third are the controller design that require a parametric model to synthesize a controller. Note that all of these methods do not have a specific plant to be following and require a lot of engineering insight to work properly. Hence, guidelines will be provided to simplify the tuning of the algorithm and help the user to successfully apply them.

4.1.1 Data-driven system identification

One of the functionalities within BB8 is system identification on the basis of measurement data. Note that system identification is also discussed within T4.3, the big difference is that T4.3 used FEM analyses to create a model instead of data. Data-driven system identification allows to generate a frequency response function, or a parametric model, one of which is required to tune a controller.

4.1.1.1 Open loop vs. closed loop measurement

In general one can make a distinction between two cases, one where the system is operating in open loop and the second one where the system is operating in closed loop. The most simple case to recover the plant is the open loop case. By exciting the system through the input and measuring the output one can determined the system dynamics as follows

$$\hat{G}(j\,\omega) = Y(k)\,U(k)^{-1}$$

which is the simplest case.

In the case where the system itself is unstable or has to follow a specific path, the system can alternatively be identified in closed loop (see Figure 23). To recover the plan dynamics in this situation an additional step is required. First excite the system via d and measure u and e, secondly determine the Process Sensitivity and Input Sensitivity

$\widehat{PS}(j\,\omega) = D(k)\,E(k)^{-1}$, and

$$\hat{S}(j\,\omega) = D(k)\,U(k)^{-1}$$

and finally determined the plant dynamics as follows
$\hat{G}(j\,\omega) = \widehat{PS} \cdot \hat{S}^{-1}$.

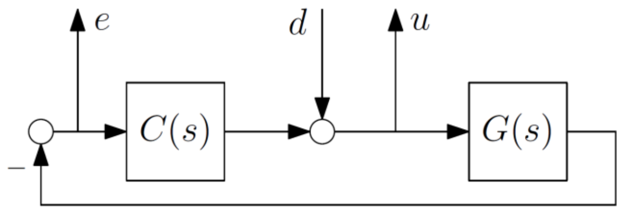


Figure 23: Generic closed-loop identification control scheme.

4.1.1.2 Excitation signal

The excitation signal is an important factor to determine the quality of the identified frequency response function. The most easy option is to excite the system equally and an all frequencies by applying a white noise signal. An alternative solution is to design a multi-sine, that allows to excite the system at specific frequencies and determine the amplitude spectrum as desired. Note that if $d(t)$is a white noise signal with a flat spectrum, then the actual excitation of the plant u becomes $u = S\,d$, hence the amplitude spectrum of u is determined by the sensitivity function. If one desires to excite the system equally for all frequencies in a closed loop setting, the excitation signal must have an amplitude spectrum that approximates the inverse of the sensitivity function.

4.1.1.3 Averaging

Finally, an important factor in system identification is time, more specifically, a longer measurement gives more information and allows to obtain a better result. Here, better result refers to having a lover variance of the identified frequency response function. A measured data set will always contain measurement noise that can be minimized by using averaging. Essentially this implies cutting the data set in multiple frames that are averages such that noise will be mitigated. Depending on the type of excitation signal, periodic or non-periodic, one may require a windowing function to avoid aliasing (see Figure 24).

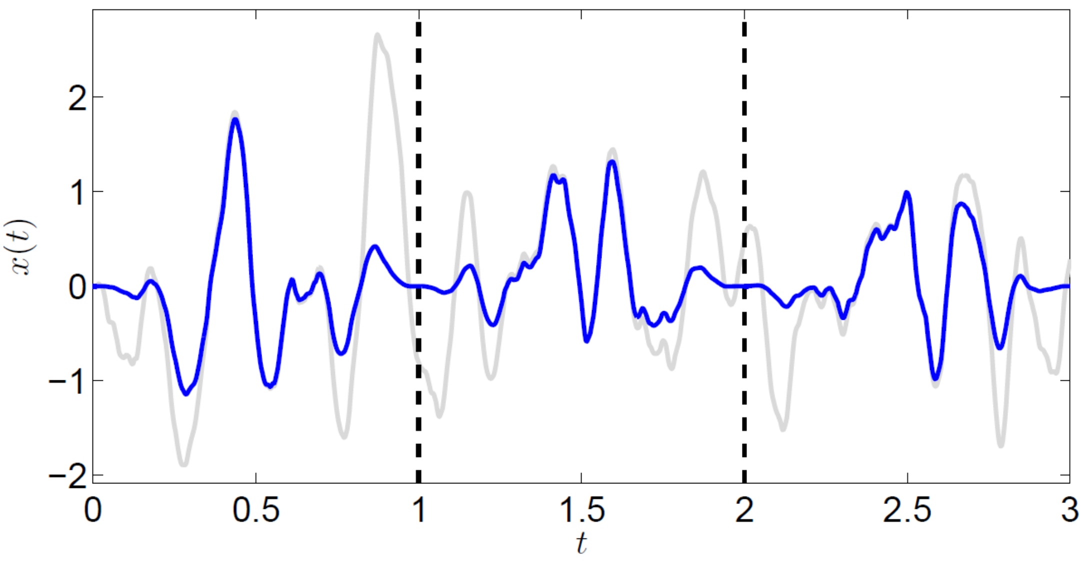


Figure 24: Original measured signal (gray) and three separate frames where a window is applied (blue).

4.1.1.4 Implementation (Matlab function)

The above mentioned cases (open-loop and closed loop) can simply be implemented in Matlab code using the following code:
Open loop case:

```
[G,~] = tfestimate(U,Y,hanning(nfft),overlap,nfft,Fs);
```

Closed loop case:

```
[S,~] = tfestimate(D,U,window,overlap,nfft,Fs);
[PS,~] = tfestimate(D,E,window,overlap,nfft,Fs);
 G = PS./S;  % computes the system FRF
```

With the following parameters.

Parameter	Interpretation	unit
nfft	frame size	[samples]
Fs	sample frequency	[Hz]
res	resolution of the frf (= nfft/Fs)	
overlap	Overlap of individual frames, e.g., 0.5*nfft for half a frame of overlap	[samples]
window	windowing function, e.g., hanning(nfft)	

(For more advanced methods, such as and background refer to Deliverable 4.4.)

To generate a multisine with random phase and specific amplitude spectrum, the following matlab code can be used;

```
% multisine generation by creating the spectrum and using the inverse Fourier transform
N = 1024;
nr_of_periods = 8;
f_min = 1;
f_max = N/3;
for k = 1:N
  if k >= f_min && k <= f_max
     a = rand(1)*2*pi;
     % Complex number with random phase and unit magnitude
     Y(k) = cos(a) + 1i*sin(a);
  else
     % unexcited frequencies
     Y(k) = 0;
  end
end
% Convert generated spectrum to the time-domain
u_period = 2*real(ifft(Y));
u_period = u_period/rms(u_period);

uu = repmat(u_period, nr_of_periods);
u = uu(1,:);          % multisine input signal with 8 periods
```

Parameter	Interpretation	unit
N	number of frequencies	
nr_of_period	number of periods in total	
f_min	frequency from which the spectrum of the multisine is non-zero	[hz]
f_max	frequency until which the spectrum of the multisine is non-zero	[hz]
u	multi-sine	

4.1.1.5 Coupling analyses of the basis of measured FRF

Given a measured frequency response function or transfer function matrix of a system, the amount of coupling between inputs and outputs can be investigated using the relative gain array. If the relative gain array approximates a diagonal identity matrix, then the system is decoupled (until that frequency). Often mechatronic systems are (or can be) well decoupled until the first resonance mode, this is referred to as rigid body decoupled.

4.1.1.6 Implementation (Matlab function)

The relative gain array can simple be computed as follows in Matlab.

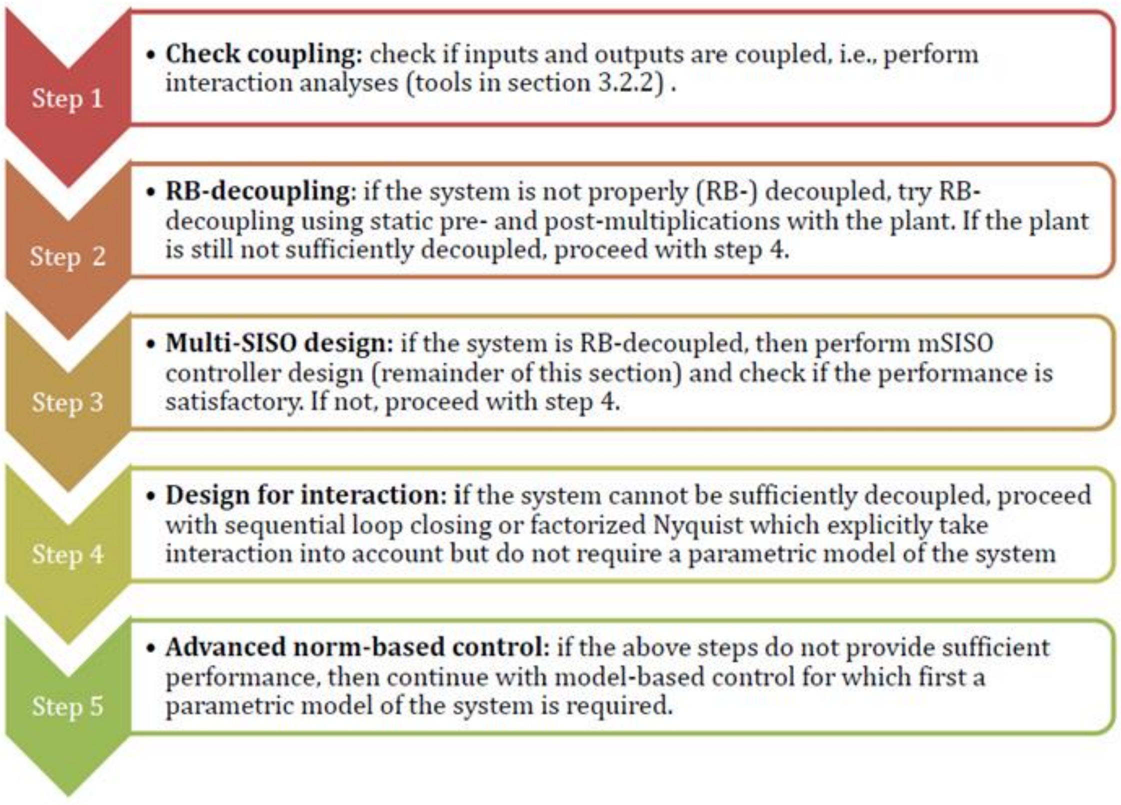
```
RGA = G.*pinv(G).';
```

where G is a complex valued matrix containing the frequency response function of the system.

4.1.2 Multivariable controller design

This document contains several methods that allow to design a controller for a multivariable system on the basis of either a measured frequency response function or a parametric model. A stepwise approach to multivariable controller design that is applicable to motion system is given in Figure 25.

Note: these methods still require engineering insight to successfully interpret the results and tune the controller. The aim of this section is to provide the general reasoning behind the methods and to get started with a simple script that can be extended to the specific needs of the application.



Step 1
- **Check coupling:** check if inputs and outputs are coupled, i.e., perform interaction analyses (tools in section 3.2.2) .

Step 2
- **RB-decoupling:** if the system is not properly (RB-) decoupled, try RB-decoupling using static pre- and post-multiplications with the plant. If the plant is still not sufficiently decoupled, proceed with step 4.

Step 3
- **Multi-SISO design:** if the system is RB-decoupled, then perform mSISO controller design (remainder of this section) and check if the performance is satisfactory. If not, proceed with step 4.

Step 4
- **Design for interaction:** if the system cannot be sufficiently decoupled, proceed with sequential loop closing or factorized Nyquist which explicitly take interaction into account but do not require a parametric model of the system

Step 5
- **Advanced norm-based control:** if the above steps do not provide sufficient performance, then continue with model-based control for which first a parametric model of the system is required.

Figure 25: MIMO controller design steps.

4.1.2.1 Decoupling

The first step to design a controller for a multivariable system is to check whether the system is decoupled. This can be done using the RGA as discussed before. If the system is not decoupled, one can proceed with rigid-body decoupling based on the geometric information of the system. Unfortunately, there is not generic approach for decoupling that works for all systems based on a measured FRF. Essentially, decoupling implies that one must find a static matrix T_u and $T_y$such that a new system

$$G_{dec} = T_y G T_u$$

is decoupled. How to design the decoupling matriced is application specific.

Once a system is (rigid-body) decoupled, a simple multi-SISO controller can be designed, i.e., design a controller for each loop individually (see Figure 26). For motion systems the following rules of thumb can be used as a starting point for the SISO controller.

$$C_i(s) = K_p \cdot C_{lead}(s) \cdot C_{lp}(s) = K_p \cdot \frac{s + BW/3}{s + 3 * BW} \cdot \frac{1}{s + BW * 10}$$

where BW is the desired bandwidth.

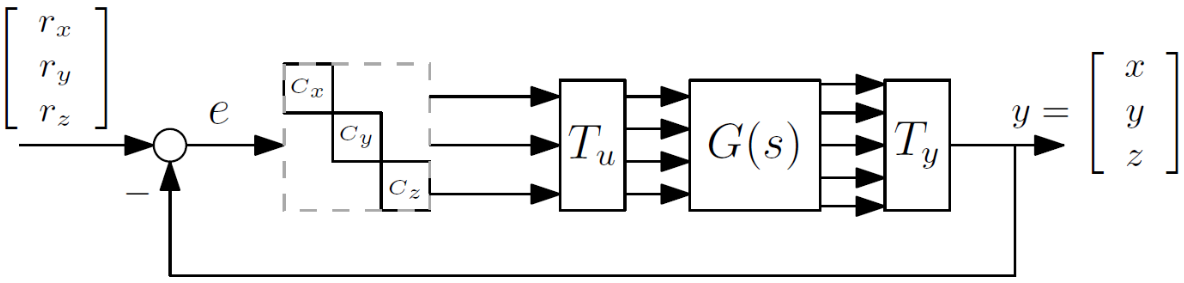

$\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$
e
$-$
C_x
C_y
C_z
T_u
$G(s)$
T_y
$y = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

Figure 26: Decoupled MIMO control scheme.

4.1.2.2 Sequential loop closing

In the case that a system is not decoupled, or it is not desired/possible to decouple the system or performance is not as desired, then sequential loop closing can be applied directly. Consider the following example (see Figure 27) on a 2x2 system to present the sequential loop closing principle.

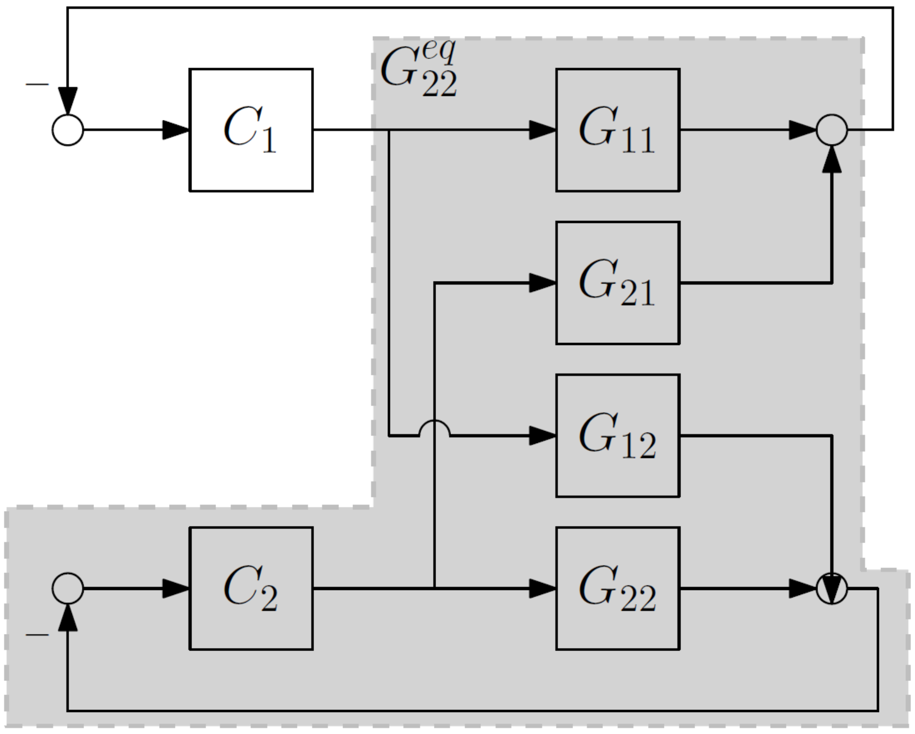


Figure 27: Non-decoupled MIMO control scheme.

step 1: Find a stabilizing controller $C_2(s)$ for the plant $G_{22}(s)$
step 2: compute the equivalent plant containing also the interaction $G_{22}{}^{eq} = G_{22} - \frac{G_{12}C_1G_{21}}{1+G_{11}C_1}$
step 3: find a stabilizing controller $C_1(s)$ for the plant $G_{22}{}^{eq}$

If the performance is not satisfactory then one can close the loops in a different sequence. Furthermore, it is essential that the loops are closed in the same sequence as used to design the controller. Also note that stability of previously closed loops is guaranteed, however, robustness margins of previously closed loop may change if new loops are closed. Hence one should always check the robustness margins for all loops after designing the full MIMO controller.

4.1.2.3 Factorized Nyquist

Factorized Nyquist is a tool to analyze stability of a MIMO system where interaction is present. Assume that the system $G$with a given controller K is stable if the following closed loop transfer function $S = (I + GK)^{-1}$is stable. This can alternatively be written as

$$(I + GK)^{-1} = (I + \tilde{G}K)^{-1}(I + E\tilde{T})^{-1}$$

where $\tilde{G}$can be seen as the diagonal elements of the system G and Erepresents the interaction. Then the following test can be used to check stability of the interaction (assuming that K is a stable controller for $\tilde{G}$).
$1)\ \rho(E\tilde{T}) < 1\ \forall\,\omega$ or $2)\ \bar{\sigma}(\tilde{T}) < \mu_{\tilde{T}}(E)\ \forall\,\omega$

where $\rho(E\tilde{T})$ and $\bar{\sigma}(\tilde{T})$ can be computed using the following scripts.

```
function rho = Compute_spectral_radius(ET,f)
```

```
% Computes the spectral radius of ET which is a transfer function and f is
% the frequency vector.

        ET_freq = freqresp(ET,f);

        for k = 1:1:length(f)
          rho(k) = max(abs(eig(ET_freq(:,:,k))));
        end
end

function sigma_bar = Compute_max_singular_value(ET,f)
% Computes the maximum singular value of ET which is a transfer function and f is
% the frequency vector.

ET_freq = freqresp(ET,f);

for k = 1:1:length(f)
  [~,S,~] = svd(ET_freq(:,:,k));
  sigma_bar(k) = max(S);
end

end
```

finally, to compute the structured singular value$\mu_{\tilde{T}}$the Matlab function mussv can be used.

4.1.2.4 Robust model-based controller synthesis

To perform model-based controller synthesis, a parametric model of the system is required. For motion systems in general, a 4-block problem approach is applicable to shape four closed-loop transfer function. This can be done according to the following steps:

- First define the performance variables and generalized disturbances (see Figure 28)

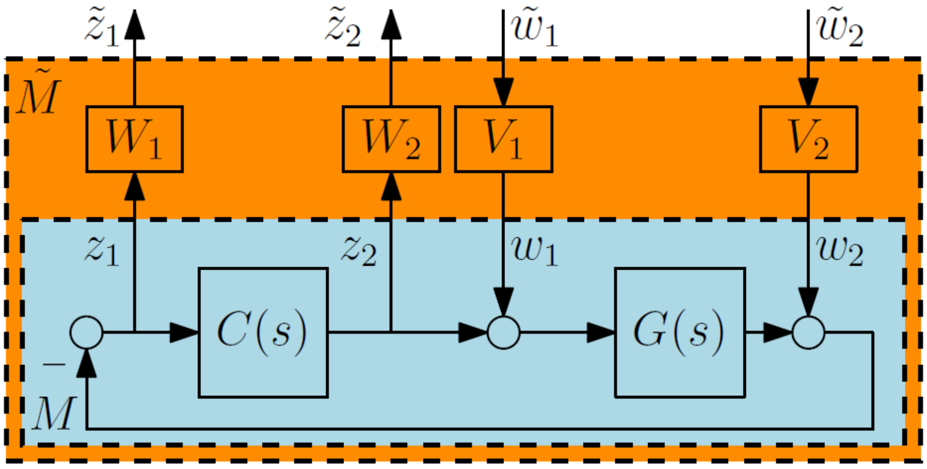


Figure 28: Performance Variables and Generalized Disturbances scheme.

$z = [z_1\, z_2\,]^{\mathsf{T}}$ and $w = [w_1\, w_2\,]^{\mathsf{T}}$. The transfer function matrix between the generalized performance variables and generalized disturbances is given by $M(s)$.

$$\underbrace{\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}}_{z} = -\underbrace{\begin{bmatrix} S & SG \\ KS & KSG \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{w}$$

Note that $M$is given by the lower linear fractional transformation (LFT) which can be computed easily in Matlab using the LFT(sys1,sys2,nu,ny)command.

- Next, the performance weights must be included in the form of weighting filters. On the input side the filters V_1 and V_2 are included and at the performance channels w_1 and w_2, which leads to the following weighted 4-block problem $\widetilde{M}$.

$$\underbrace{\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}}_{\tilde{z}} = -\underbrace{\begin{bmatrix} W_1SV_1 & W_1SGV_2 \\ W_2KSV_1 & W_2KSGV_2 \end{bmatrix}}_{\widetilde{M}} \underbrace{\begin{bmatrix} \widetilde{w}_1 \\ \widetilde{w}_2 \end{bmatrix}}_{\widetilde{w}}$$

In matlab:
V = blkdiag(V1,V2); W = blkdiag(W1,W2); Mtilde = W*M*V;
the design of shaping filters depends on the performance variables for that specific application. In general they can be seen as (inverse) of upper-bound for the four closed-loop transfer function. As an example, if one desires to have high-frequency roll-off in the control sensitivity, the filter W_2 should be chosen such that it is large in the high-frequency range. This leads to a large weight on the higher frequencies, such that the control sensitivity will become small.

- The last step is to minimize the H_∞-norm of the closed-loop system $\widetilde{M}$ over all possible stabilizing controllers, where the H_∞-norm of $\widetilde{M}$ is given by the maximum singular value over all frequencies of $\widetilde{M}$

$$k_\infty = \arg \min_{stab.\ K} \|M(j\omega)\|_\infty$$

$$\|M(j\omega)\|_\infty = \sup_\omega \bar{\sigma}(\tilde{M})$$

in Matlab this can be implemented using the following command:
[K_infty,CL,gamma] = hinfsyn(Mtilde,nu,ny)

Parameter	Interpretation
K_infty	Obtained stabilizing H-infinity controller (K inself can be unstable)
CL	Closed loop system $\tilde{M}$with designed controller K_infty
gamma	Obtained value for $\|\|M\|\|_\infty$
Mtilde	weighted closed loop system to be minimized. With inputs $[w\ y_k]$ and outputs $[z\ u_k]$.
nu	Number of controller inputs, i.e., dimension of u_k
ny	Number of controller outputs, i.e., dimension of y_k

where K_infty is the obtained H-infinity controller, CL is the system Mtilde after optimization and gamma is the obtained value for the H-infinity norm of $\tilde{M}$. If gamma < 1, then all the performance specs are obtained, if this is not the case then not all performance specifications, i.e., weighting filter, are satisfied.

4.1.3 Beyond state-of-the-art

In this section, two methods are described that are subject of current research. The first method is about online tuning of feedforward parameters on the basis of data. The second methods aim at tuning the PID parameters of the feedback controller automatically, i.e., similar to H-infinity control but with

4.1.3.1 Data-driven online feedforward parameter tuning

The principle behind this method is depicted below (see Figure 29). The idea is to generate a feedforward controller which is the inverse of the actual plant such that the tracking error $e(t)$is minimized.

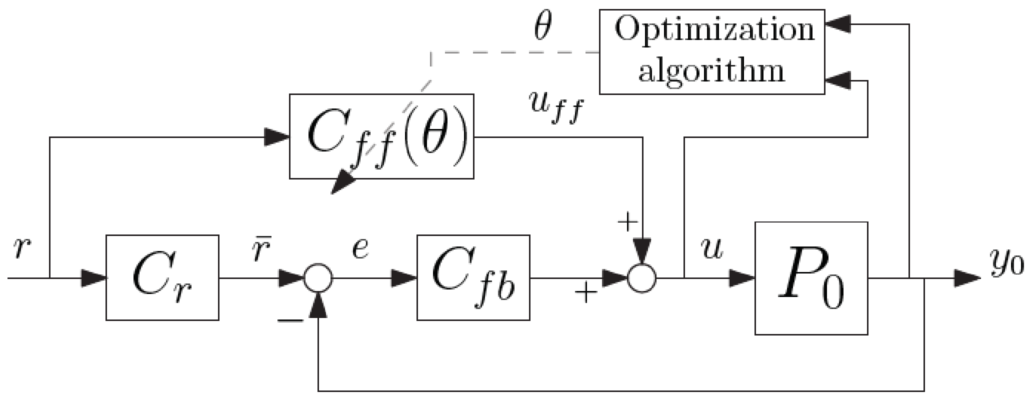


Figure 29: Data driven online feedforward parameter tuning control scheme.

The following steps can be followed in order to apply this method.

- Create a low order parametric model that approximates the true plant, consider for example:
$$P(s) = \frac{ds + k}{ms^2 + ds + k}$$
and discretize this model to obtain the discrete time description.
- The second step is to select basis functions that can represent the plant dynamics as follows,
$$\hat{P}(z) = \frac{\psi_1(q)\,\theta_1 + \ldots + \psi_n(q)\theta_n}{\psi_{n+1}(q)\,\theta_{n+1} + \ldots + \psi_m(q)\,\theta_m}$$
such that in the subsequent steps the parameters corresponding to the basis functions can be optimized. The following basis functions are widely used,
$$\psi(z) = \left(\frac{z-1}{T_s\, z}\right)^n$$
which can be seen as the discrete time equivalent of the differential operator.
- For further details on the optimization refer to [1] or [2].

The following tuning knobs are available:

Parameter	Description
$P(t_0)$	Initial 'covariance' matrix. Influences the convergence rate of the recursive least squares algorithm
$\theta(t_0)$	Initial condition for the RLS algorithm to start with, choose this based on prior knowledge or initialize this as zero.
f_c	Cut-off frequency for the low-pass filter F in [1]. Start with a low value for f_c and increase if possible. Note that a low value for f_c limits learning beyond that frequency, hence for performance it should be high and for robustness low.

4.1.3.2 Implementation

An implementation example for data-driven feedforward tuning can be found in [1] and fundamental limitations in presence of measurement noise can be found in [2].

4.1.3.3 PI(D) controller tuning using H_infinity regions approach

A new design method specifically tailored to PID controllers was developed in terms of I-MECH BB8 functionalities. It is primarily intended for PI(D) controllers and simple fixed structure feedback control algorithms with two or three parameters (lead-lag compensators, low-order static feedback etc.).
The basic features are summarized as follows:

- Formulation of the design specifications in the frequency domain by imposing arbitrary closed-loop weighted sensitivity inequalities
- Possibility of automatic calculations for the auto-tuning purposes
- Generic method for an arbitrary LTI system described by a rational transfer function + time delay
- Analytical method for the computation of the admissible set of controllers, no performance losses due to approximations, model reduction or non-convex numerical optimization

The performance specification can be expressed by means of a loopshaping inequalities

$$\|H(s,k)\|_\infty < \gamma,$$

where H corresponds to an arbitrary closed-loop transfer function given in the form of:

$$H(s,k) \triangleq W(s)S_*(s)$$

The transfer function S(s) denotes one the closed-loop sensitivity functions (sensitivity, complementary-, input- and controller-sensitivity (see Figure 30)) and W(s) introduces a user-defined frequency-dependent scaling

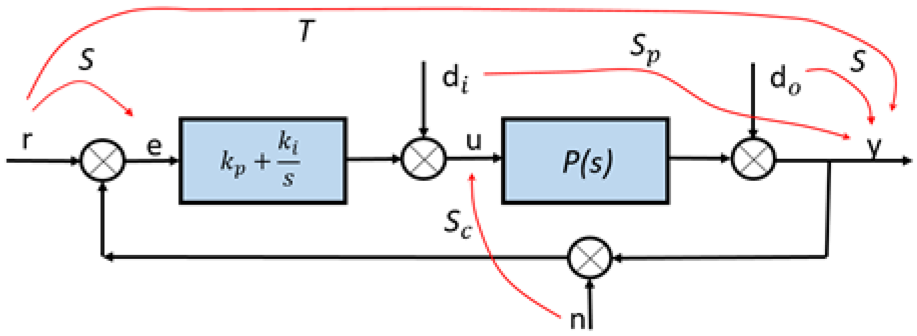


Figure 30: Transfer function representation on a standard closed loop control scheme.

The goal is to find a controller C(s,k) which together with the given H(s) fulfill the following three conditions

1. C(s,k) internally stabilizes the closed-loop
2. H(s,k) used in the design criterion is stable
3. The H-infinity norm condition holds $\|H(s,k)\|_\infty \le \gamma$

More thorough explanation of the algorithm is given in the I-MECH D4.4 deliverable.
Novel theoretical findings and applications of this method in motion control design problems were reported in [14]-[16].

4.1.3.4 Implementation

The proposed design method was implemented in Matlab environment. It can work also as a standalone application running independently from Matlab SW by generating an executable using the Matlab Compiler code generation function.
The algorithm is summarized in the following steps:
1. Derive the plant model P(s) from a mathematical modelling or experimental identification
2. Formulate arbitrary number of design constraints in the frequency domain by shaping the closed-loop sensitivity functions or their mixed-sensitivity combination
3. Compute the H-infinity region for the individual loopshaping inequalities from the 2nd step.
4. Derive the intersection of the individual H-infinity regions. The resulting set in the parametric plane defines the set of admissible controllers.
5. Select one particular controller from the admissible set. A suitable candidate is the one with the maximum integral gain in the case of PI(D) compensator. A method of gridding may be used to fulfill some additional design requirements which could not be addressed directly in step 2.

H-infinity design of PI(D) controllers	
Inputs, Parameters and Constants	
Parameter	Description
P	LTI Plant model in the form of transfer function
Wi	Constant or frequency dependent weighting function for the i-th design requirement
SFi	Closed-loop sensitivity funtion for the i-th design requirement
gami	Scalar parameter for the i-th design requirement formulated as a loop-shaping inequality ||Wi*Sfi||<gami

Outputs	
Hinfreg.fig	Matlab figure with plotted H-infinity regions defining the admissible set of controllers
Kopt	Optimal PI(D) controller minimizing the IE criterion
Regs	Vector containing the samples of the derived H-infinity region in the parametric plane of the controller which designates the whole set of admissible controllers (if non-empty). This can be used for a subsequent fine-tuning by selecting a particular controller for the admissible set using a secondary design objective.

The implemented function can be called form the command line. Alternatively, a simple graphical user interface designed in Matlab (see Figure 31) can be used to formulate the design requirements and evaluate the design results interactively

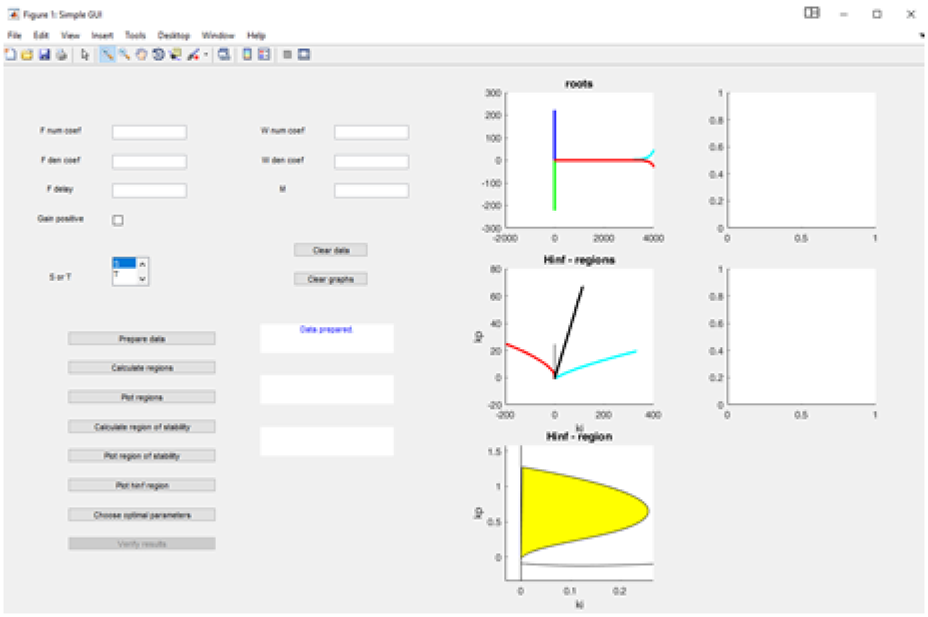


Figure 31: Simple GUI for the implemented H-infinity design method.

4.2 Implementation aspects

4.2.1 Pilot 1

System identification and coupling analyses is successfully applied to Pilot 1, the main outcome is that interaction in the system is limited and therefore no MIMO control techniques are required and multi-SISIO with rigid body decoupling is sufficient. To apply sequential loop closing for validation, the decoupling matrices have been perturbed (to simulate interaction) and a controller is designed using SLC.

The above described BB8 functionalities have been implemented in Matlab/Simulink in Pilot 1.

4.2.2 Pilot 2

Similar to Pilot 1, system identification and decoupling analyses have been performed. Note that Pilot 2 is a brownfield application [17], [18].

5 BB9 - Task 4.6 Iterative and repetitive control methods

5.1 Functionalities

5.1.1 Repetitive Control for repetitive disturbance compensation (UNIBS/GEF)

5.1.1.1 General description - algorithm theory

In this section a repetitive control (RC) strategy for the compensation of periodic and/or repetitive disturbance is presented.
The generic structure of a Repetitive Controller for the compensation of periodic and/or repetitive disturbance is shown in Figure 32.
The Repetitive Controller, in order to be applied to the control structure, needs the knowledge of the closed loop transfer function of the controlled system. It is in fact possible to say that the Repetitive Controller is based on the model of the system to control and on the closed loop controller applied.

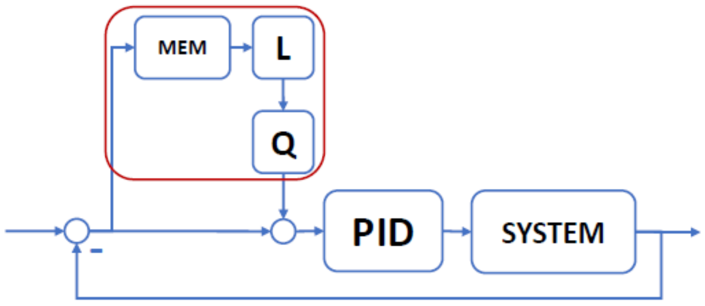


Figure 32: Repetitive control scheme.

5.1.1.2 Implementation in Matlab-Simulink C

The RC technique has been implemented in Matlab/Simulink (see Figure 33).

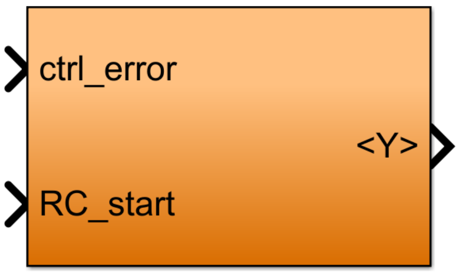


Figure 33: Repetitive control Simulink Block.

After the validation the block has been exported in FMU to be used in different simulation environments.

5.1.1.3 Inputs, outputs, parameters, constants

Inputs, outputs, parameters and constants of the RC technique.
In this section a brief description on inputs, outputs, parameters and constants is shown. Please, refer to the function help for more detailed information.

RC Block	
Inputs, Parameters and Constants	
Parameter	Description
ctrl_error	Control error
RC_start	Enable of the RC block
Kr	RC gain
Ts	Sampling time [s]
d	Delay correction constant
Td	Disturbance period [s]
FIR_cutoff_f	Stability filter cutoff frequency [Hz]
numGz	Numerator of the closed loop transfer function
denGz	Denominator of the closed loop transfer function
Outputs	
Parameter	Description
Y	RC output to sum to the control error

5.1.1.4 Validation in MIL, PIL, HIL

The validation of the RC technique has been done in MIL by using Matlab/Simulink.
Then, always in MIL, the technique has been validated by using Simulink-Multibody and in co-simulation Simulink-Amesim.
In Figure 34 the example of the Simulink scheme for the MIL validation of the Repetitive Control technique is shown.

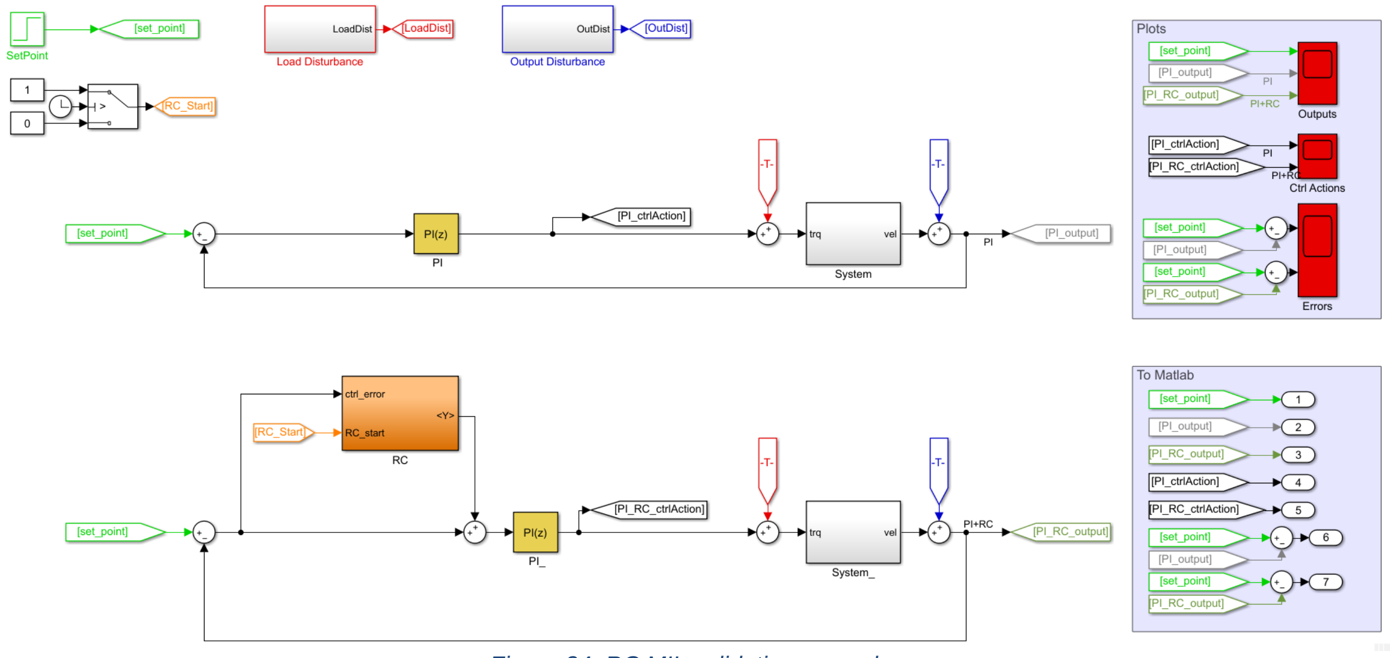


Figure 34: RC MIL validation example.

The Repetitive Control technique has then been validated in PIL and HIL on the Use Case 1.1.

5.1.2 Iterative learning Control for repetitive disturbance compensation (UNIBS/GEF)

5.1.2.1 General description - algorithm theory

In this section an Iterative Learning Control (ILC) strategy for the compensation of periodic and/or repetitive disturbance is presented. The generic structure of an Iterative Learning Controller for the compensation of periodic and/or repetitive disturbance is shown in Figure 35.
The Iterative Learning Controller, differently from the Repetitive Controller, in order to be applied to the control structure, doesn't need the knowledge of the closed loop transfer function of the controlled system. It is in fact possible to say that the Iterative Learning Controller is NOT based on the model of the system to control and on the closed loop controller applied.

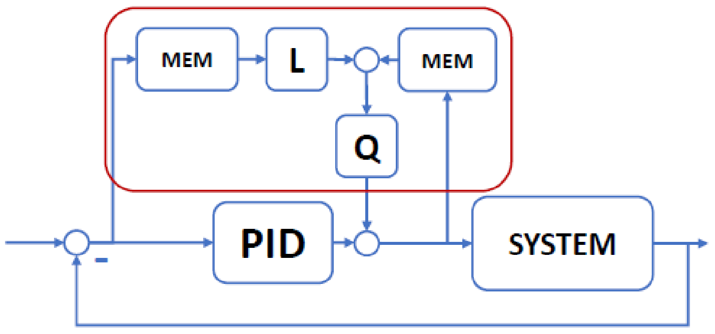


Figure 35: Iterative learning control scheme.

5.1.2.2 Implementation in Matlab-Simulink C

The ILC technique has been implemented in Matlab/Simulink (see Figure 36).

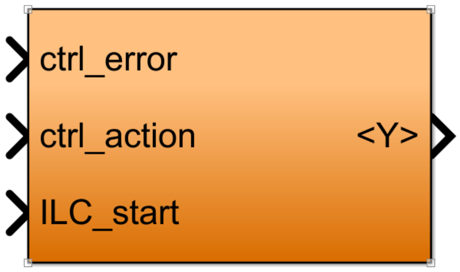


Figure 36: Iterative learning control Simulink Block.

After the validation the block has been exported in FMU to be used in different simulation environments.

5.1.2.3 Inputs, outputs, parameters, constants

5.1.2.3.1 Inputs, outputs, parameters and constants of the ILC technique.

In this section a brief description on inputs, outputs, parameters and constants is shown. Please, refer to the function help for more detailed information.

ILC Block	
Inputs, Parameters and Constants	
Parameter	Description
ctrl_error	Control error
ctrl_action	Control action
ILC_start	Enable of the ILC block
L_gain	ILC gain
Ts	Sampling time [s]
d	Delay correction constant
Td	Disturbance period [s]
FIR_cutoff_f	Stability filter cutoff frequency [Hz]
Outputs	
Parameter	Description
Y	ILC output to sum to the control action

5.1.2.3.2 Validation in MIL, PIL, HIL

The validation of the ILC technique has been done in MIL by using Matlab/Simulink, then, always in MIL, the technique has been validated by using Simulink-Multibody and in co-simulation Simulink-Amesim.
In Figure 37 the example of the Simulink scheme for the MIL validation of the Iterative Learning Control technique is shown.

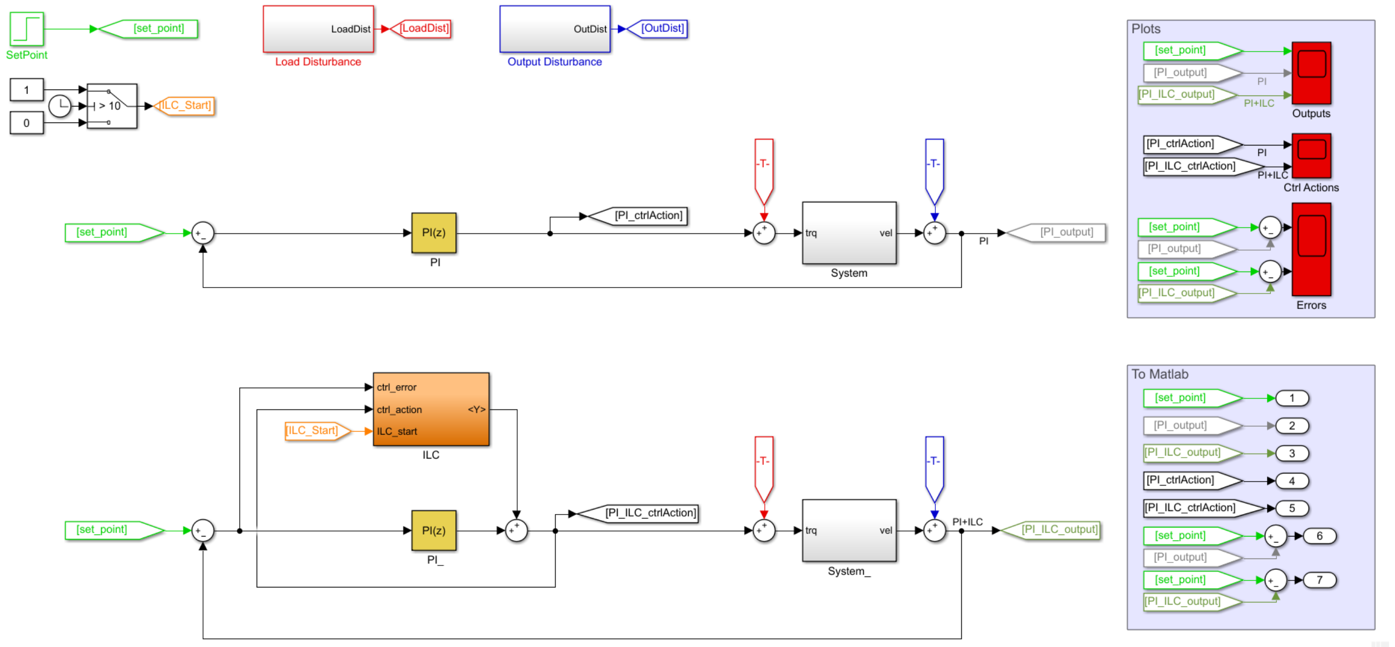


Figure 37: ILC MIL validation example.

The Iterative Learning Control technique has then been validated in PIL and HIL on the Use Case 1.1.

5.1.3 Anticipatory ILC (TEK)

5.1.3.1 General description - algorithm theory

Anticipative ILC can help to push the capacity of the machine beyond these limitations in both complex trajectory machining and in rigid tapping. It consists in adapting the results of the proportional ILC by accounting for the settling time of the controlled system in closed loop [19].

5.1.3.2 Implementation in Matlab-Simulink C

The anticipatory ILC techniques has been implemented in Matlab/Simulink (see Figure 38).

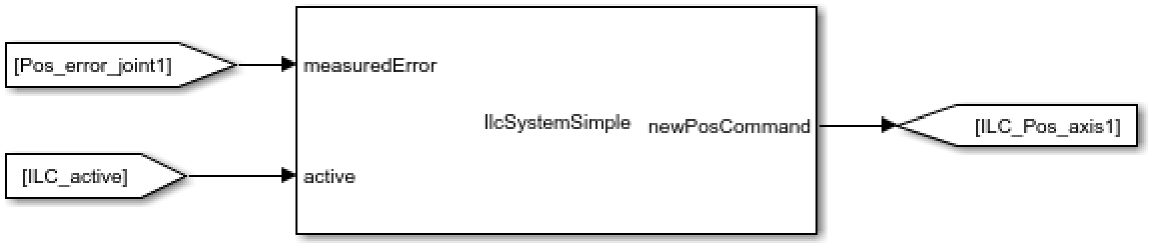


Figure 38: Anticipatory ILC Simulink Block.

After the validation the block have been exported in FMU in order to be used in different simulation environments.

5.1.3.3 Inputs, outputs, parameters, constants

Inputs, outputs, parameters and constants of the anticipatory ILC technique.
In this section a brief description on inputs, outputs, parameters and constants is shown. More details are available in the function help.

MPC Blocks	
Inputs, Parameters and Constants	
Parameter	Description
measuredError	Buffer containing the positioning error in the previous iteration
active	Operating mode (0: calculation for the next iteration is performed; 1: input/output buffers load/download)
Outputs	
Parameter	Description
newPosCommand	ILC output buffer to be fed in position FFW input of reference controller

5.1.3.4 Validation in MIL, PIL, HIL

The validation of the anticipatory ILC technique has been done in MIL by using Matlab/Simulink and considering Use Case 1.2 as a reference.

Figure 39 presents the general view of the developed model. It must be highlighted that two versions of the mechanical section of the model has been tested, the first one using Simulink-Multibody (Simscape) and the second one using Amesim.

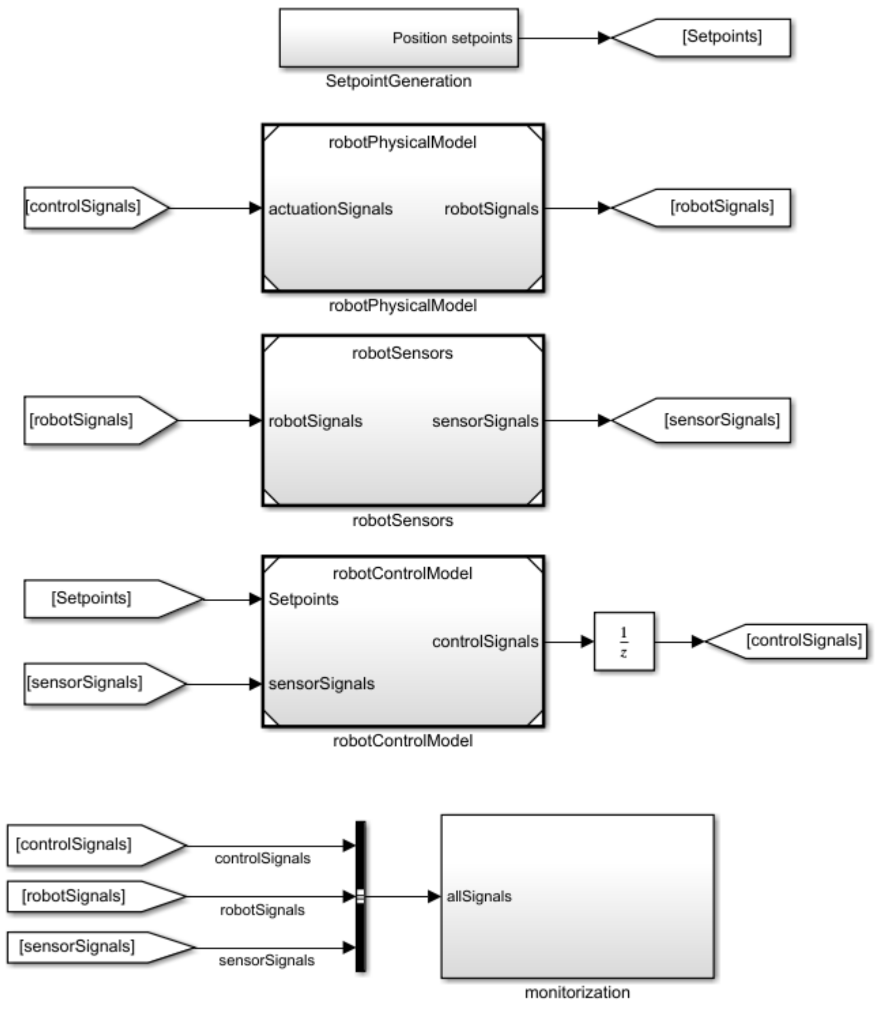


Figure 39: Anticipatory ILC MIL validation example.

Figure 40 presents the content of the control block of the upper diagram. There, it can be observed how the outputs of the anticipatory ILC blocks are directly fed to the corresponding input of the reference controllers of the respective joint of the robot.

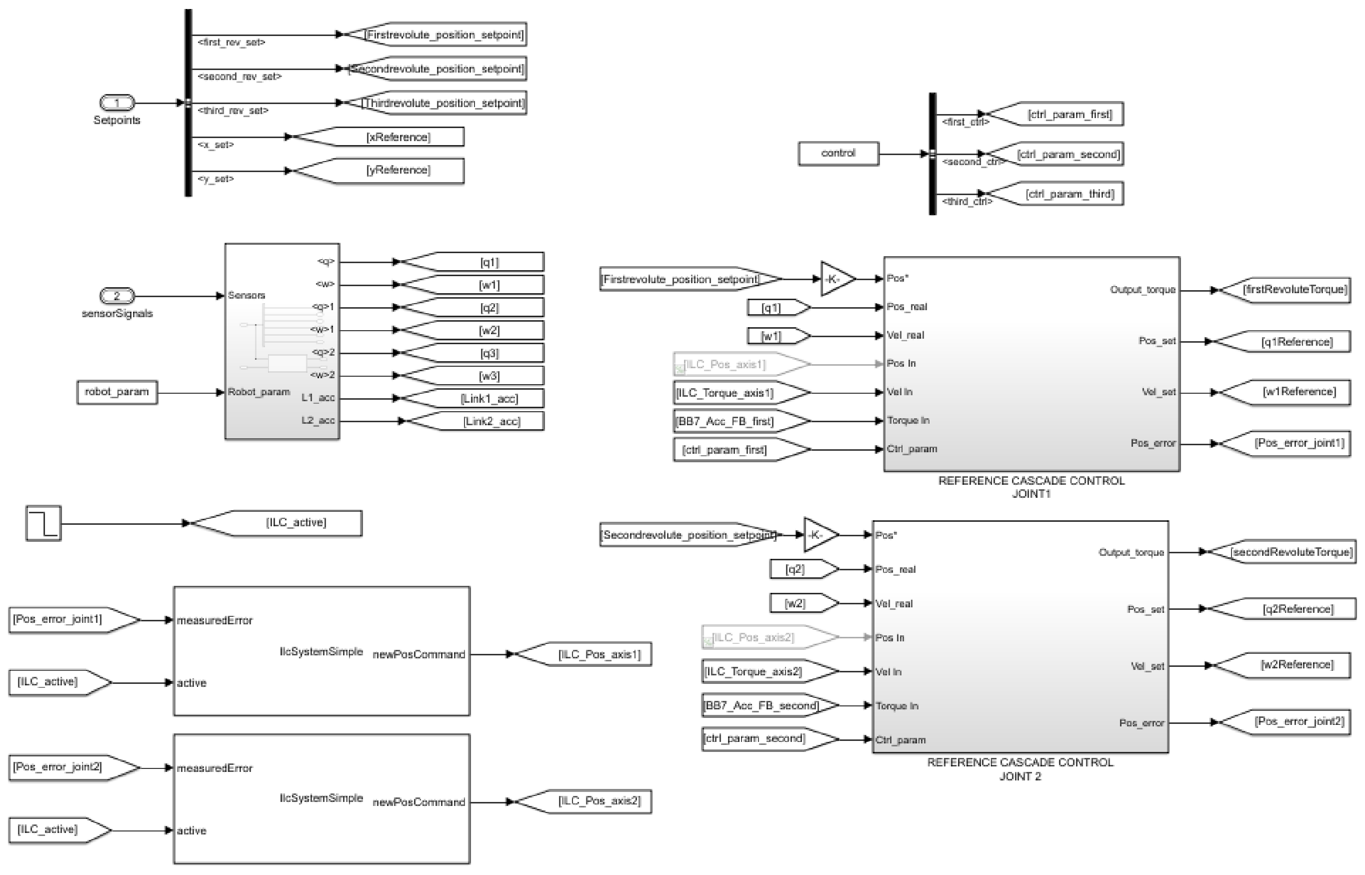


Figure 40: Anticipatory ILC MIL expanded validation example.

Validation has been done in HIL on UC1.2.

5.1.4 PI(D) Repetitive controller (ZAPUNI)

This functionality provides a design method for the synthesis of plug-in type PID + repetitive controller for the reference architecture shown in Figure 41. The goal is to parameterize the feedback controller, which assumed to be of a PI(D) type, and the repetitive control block R which can be appended when the compensation of a periodic disturbance or tracking of a periodic reference is required with a high fidelity. Conventional methods, known from the literature, assume the feedback controller C to be known in advance, usually assumed to be tuned for the nominal plant P without the repetitive control part R. The advantage of the novel design method is an algorithm which allows a concurrent design of both the feedback controller and the repetitive control module while guaranteeing stability and performance specifications for both nominal (w/o RC) and RC modes of operation.
The theoretical development of the design method is explained in [20], [21].

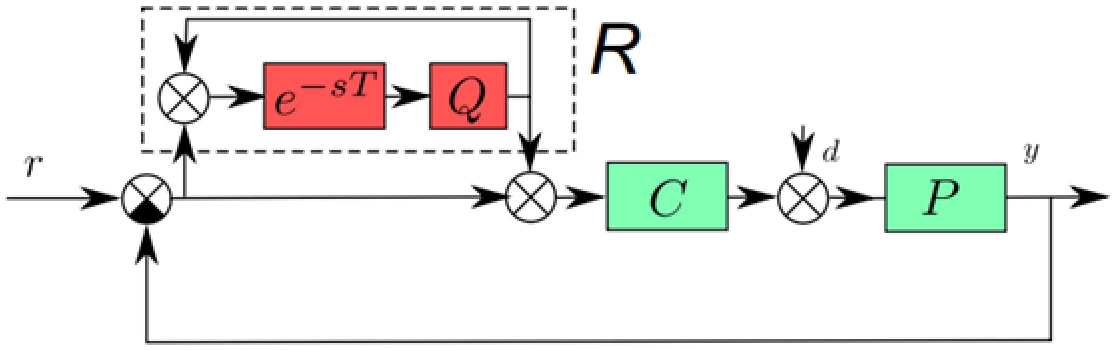


Figure 41: Assumed repetitive control setup, P – controlled plant, C – feedback compensator, R – plug-in repetitive control block, Q – robustness filter, T – period of repetitive disturbance in r/d

5.1.4.1 Implementation aspects

The design method is accessible in the form of C-MEX function accessible from the Matlab environment. The result of the controller synthesis are the parameters of the PID feedback controller and the repetitive control block. They can be used on a target platform to parameterize an existing control structure. Alternatively, C-code can be generated directly from the Matlab-Simulink environment using the Simulink Coder.

PID + RC design	
Syntax: [C,Q]=pidrc(P,WS,WT,WSrc,Tper,wq)	
Inputs, Parameters and Constants	
Parameter	Description
P	Plant transfer function model
WS	Weighted sensitivity loop-shaping filter for the nominal stability and performance specification
WT	Weighted complementary sensitivity loop-shaping filter for the nominal stability and performance specification
WSrc	Robustness and performance specification for the RC regime of operation
Tper	Period of the reference/disturbance for the RC part
wq	Assumed disturbance bandwidth [Hz]
Outputs	
Parameter	Description
C	PID controller
Q	Robustness filter

The proposed design method was validated experimentally using ZAPUNI Flexible arm motion stage. The results are summarized in the above mentioned references. More detailed explanation is available in the D4.5 deliverable.

5.1.5 Model-Based ILC strategies (TUE)

Iterative learning control can be used for systems performing repeating motion tasks (see Figure 42 and Figure 43) to remove the repeating parts of the error. The main assumption is that the initial condition at the start of each motion task is equivalent to the previous tasks. The main idea is to compute a feedforward signal u_{ff} for the next task, indicated with $j+1$, on the basis of the error data from the current task, a learning filter and a robustness filter.

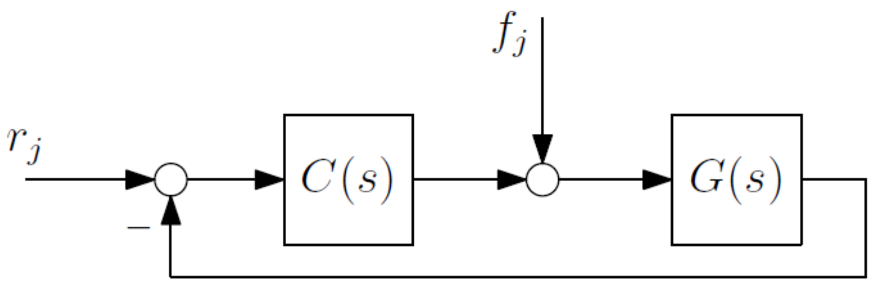


Figure 42: Generic closed-loop control scheme.

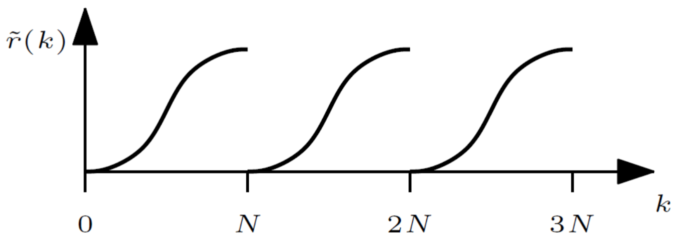


Figure 43: Generic repetitive task set point signal.

The ILC update is given by

$$f_{j+1} = Q(f_j + \alpha\, Le_j)$$

with the following parameters.

Parameter	Description
f_j, f_{j+1}	Feedforward signal for the j^{th} task, vector with length equal to the task length
e_j	Error corresponding to the j^{th} taks, vector with length equal to the task length
L	Learning filter, ideally $L = (\widehat{PS})^{-1}$, where $\widehat{PS}$is a parametric model of the process sensitivity.
Q	Robustness filter. Design Q such that the stability condition $\lvert Q(1+GSL)\rvert < 1$holds for all frequencies. For performance Q is preferably close 1 and learning is limited for frequencies where $Q < 1$.
α	Learning gain. If trial varying disturbances are present it is not prefered to learn there each task. Hence a learning gain $\alpha < 1$ can be used to reduce the learning per trial and hence be robust against trial varying disturbances.

If the process sensitivity contains non-minimum phase zeros, then the following matlab function can be used to compute a stable but potentially non-causal inverse.

```
[Lnum, Lden, phd] = zpetc(PS.a,PS.b,PS.c,PS.d,rho);
Lc = tf(Lnum, Lden,Ts);             % causal part of the inverse
z = tf('z', Ts);
L = Lc*z^phd;                       % non-causal inverse
```

Parameter	Description
PS.a, PS.b, PS.c, PS.d	state-space matrices of the process sensitivity
rho	Stability radius of in the complex plane for discrete-time systems. (default = 1, is the unit circle)
Lnum, Lden	Numerator and denominator coefficients of the causal part of L

phd	the number of non-causal samples in the learning filter, i.e., $L_c = z^{phd} \cdot L_{nc}$
Ts	Sample time

The following matlab example can be used to compute the feedforward update:

```
function [f_next] = ILC_update(error,f_prev,L,Q,alpha,phd)

        % Filter error with learning filter L
        L_out = alpha.*filter(L.num{1},L.den{1},error);

        % Phase compensation, shirt feedforward back of phd samples
        L_out_shift = [L_out(phd+1:end); zeros(phd,1)];

        % Feedforward signal without Q filter
        f_next_noQ = f_prev + L_out_shift;

        % Anti-causal filtering of f_next to implement Q filter
        f_next = filtfilt(Q.num{1},Q.den{1},f_next_noQ);

end
```

Parameter	Description
Q	Q-filter transfer function
L	L-filter transfer function
error	Error obtained from the previous trial
f_prev	Previous feedforward signal
Alpha	learning gain
phd	number of non-causal samples is Q (see also the zpetc function)
f_next	feedforward signal for the next trial

Note that alternative implementations are available if the reference does change from one task to the next. For more details check for example [22]-[25].

5.2 Implementation aspects

5.2.1 Pilot 1

Repetitive control is applied to Pilot 1. The specific form of RC that has been implemented is with a buffer in the position domain. This allows to reject disturbances with varying frequency, whereas standard RC can only deal with disturbances that have a fixed period. For confidentiality reasons the details cannot be included in this document, however, to learn more on this topic refer to [22].

5.2.2 Pilot 2

Standard ILC has been implemented on Pilot 2. Note that Pilot 2 is a brownfield application, hence the code that is used to implement ILC is in principle the same as shown in section 7.1.5 but it is not given here.

5.2.3 Use Case 1.1

5.2.3.1 Repetitive Control for repetitive disturbance compensation (UNIBS/GEF)

The Repetitive control for repetitive disturbance compensation has been implemented in the HIL setup in Brescia. The HIL setup uses the same Gefran ADV 200 drives that are mounted on the UC 1.1. The Control algorithm has been written in IEC61131.3 standard structured text language. The algorithm runs at 1 ms sampling rate.

5.2.3.2 Iterative Learning Control for repetitive disturbance compensation (UNIBS/GEF)

The Iterative Learning Control for repetitive disturbance compensation has been implemented in the HIL setup in Brescia. The HIL setup uses the same Gefran ADV 200 drives that are mounted on the UC 1.1. The Control algorithm has been written in IEC61131.3 standard structured text language. The algorithm runs at 1 ms sampling rate.

5.2.4 Use Case 1.2

5.2.4.1 Anticipatory ILC (FAG/TEK)

The anticipatory ILC functionality has been implemented in UC1.2 in C-code, thanks to a special option of FAGOR CNCs that allows including additional features to the control loops. The algorithm runs at position loop sampling rate, in this case 2 ms.
For the memory management, which is critical for ILC, FAGOR's Data Logger tool has been used. This tool is able to read any of the CNC variables at position loop sample time and store it in a csv file.
The ILC algorithm reads the csv file generated in the previous iteration and gets the needed information to calculate the inputs for the next iteration [26].

6 BB11 - Task 4.7 Advanced motion control algorithms and software for predictable multi-many core platforms

6.1 FPGA platform

6.1.1 Functionalities

BB10/BB11 are specifically concerned with creating PIL and HIL configurations to test and debug using Simulink environment. The testing, debugging and performance analysis of control building blocks (BB6-BB9) may be possible using the above environments. In these environments, the following design questions are answered.

- Controller code execution is firm real-time on the target platform and get the timing numbers such as execution times.
- Feasible sampling period.
- Memory requirement to execute the control codes.
- Functional verification of the overall implementation.
- Verification of performance (e.g., response time).
- Logging and scoping of signal and errors.
- Data-type support and conversions validation.

As explained in D2.4, Maltab® Simulink® will be used as a model-based design environment for controller design, testing and testing. The design process will start with full MIL testing in the Simulink environment. Under I-MECH, we use Simulink Embedded Coder to generate code for the considered target platforms – i.e. COTS platform (with a dedicated RTOS) or FPGA platform, depending on the platform choice in BB10. The code is further compiled using target specific compilers (i.e. GCC for both platforms) to generate the binary and download to the target platform. This is depicted in Figure 44.

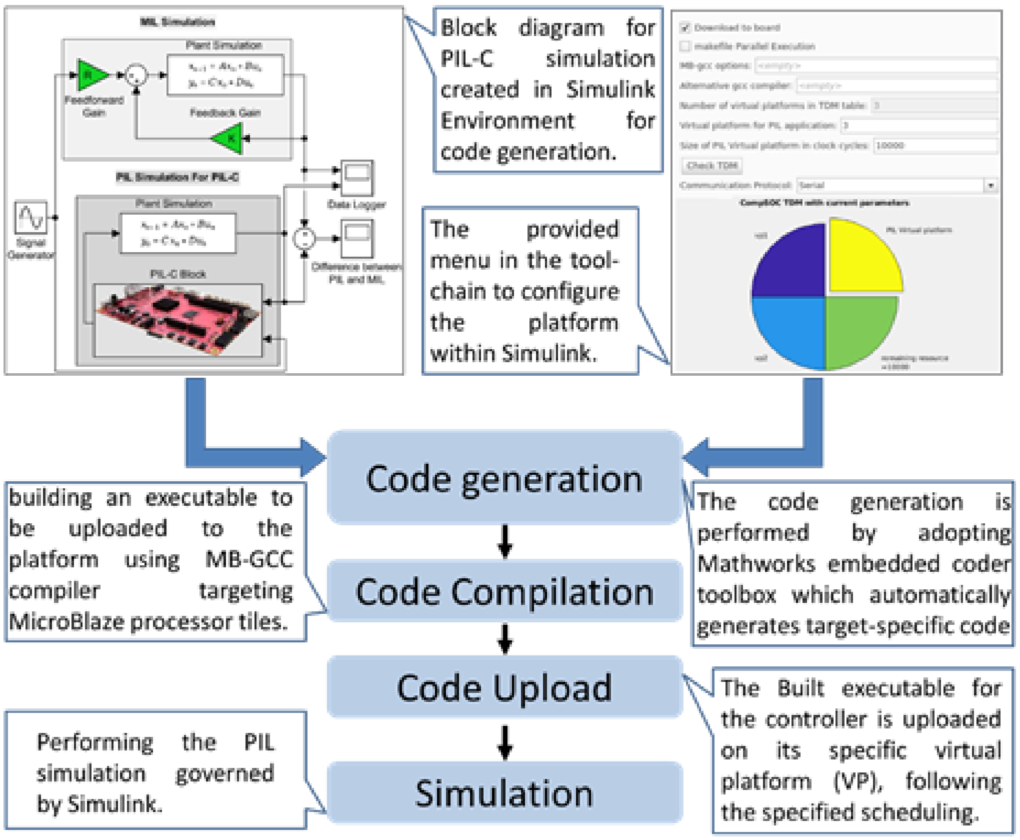

MIL Simulation
Plant Simulation
Feedforward Gain
Feedback Gain
PIL Simulation For PIL-C
Signal Generator
Data Logger
Difference between PIL and MIL
PIL-C Block
Block diagram for PIL-C simulation created in Simulink Environment for code generation.
The provided menu in the tool-chain to configure the platform within Simulink.
Code generation
Code Compilation
Code Upload
Simulation
building an executable to be uploaded to the platform using MB-GCC compiler targeting MicroBlaze processor tiles.
The code generation is performed by adopting Mathworks embedded coder toolbox which automatically generates target-specific code
The Built executable for the controller is uploaded on its specific virtual platform (VP), following the specified scheduling.
Performing the PIL simulation governed by Simulink.

Figure 44: Model-based design methodology adopted for BB11.

In PIL configuration setup Maltab® Simulink® drives the overall system. This allows after each simulation step to inspect the internal state. Therefore, signals inside the controller and plant can be traced and viewed from inside Matlab® Simulink® and compared to Maltab® Simulink® simulation.
The PIL framework is developed as a *system target file* for a custom target. When a model is provided in Simulink and the MIL simulations are successful, the next step is to perform the PIL simulations to achieve:

- Memory requirement to execute the control codes.
- Functional verification of the overall implementation.
- Verification of performance (e.g., response time).
- Data-type support and conversions validation.

In the following, the steps of performing PIL simulation using the provided framework is demonstrated.

6.1.1.1 PIL Simulations in Simulink

To perform a PIL simulation, the user first chooses the developed *system target file (STF)* as the code generation tool to generate the target specific code (in this case the FPGA platform). In a Simulink model (which is already used for MIL simulations), the user can choose the custom *system target file*(which is called CompSOC_ec.tlc) in: **Configuration Parameters> Code Generation>System Target File> CompSOC_ec.tlc.**
When the STF is chosen, a menu called **CompSOC_Options** is added to the code generation bar where the user can modify the scheduling of the targeted FPGA platform. This menu can be seen in the top right of the previous Figure.

6.1.1.1.1 Code generation report and code profiling

As discussed the PIL simulation provides the information about the memory requirements and the response time of the simulation on the platform.
The *code generation report* is a report created from the code generation, where the user can trace the size of each of the generated code for each part of the model. The user can enable the code generation report through: **Configuration Parameters> Code Generation> Report> Enable code generation report.**
The developed STF is able to report the execution time of the model on the target platform. This is provided to the user as a pop-up report at the end of the simulation. The user can enable the execution time measurement through: **Configuration Parameters> Code Generation> Verification> Measure execution times.**

6.1.1.1.2 Performing PIL simulation

Now that all the parameters are set, the user can perform a PIL simulation on the target platform. To do this, changes the simulation mode to *processor in the loop (PIL)*, and hit the run button. In the background (where the steps can be traced through *diagnostic viewer*) the STF first generates the code for the model with the usage of Mathworks embedded coder. When the code is generated, the STF continues the process by compiling the generated code and building the executable. At the end of build, the STF reports the size of the executable and in the case the size of the executable is bigger than the target memory, is halts the simulation and gives an error, reporting the issue.
If the executable fits in the target memory, the STF uploads the executable on the target platform and starts the simulation. Within the simulation the user can track the output signals (via simulation sink blocks such as *Scope*).
When the simulation is finished, the code profiling report pops up, and the user can view the measured execution time for the different parts of the model.

6.1.1.1.3 PIL simulation for a part of the model

The STF is able to perform the PIL simulation for a part of the model (instead the whole model). This means that a part of the simulation stays in Simulink environment and is simulated by the Simulink engine, while another part is simulated on the target platform. As discussed in section 6.1.1.1, Simulink is the overall system in PIL simulations and perform an I/O exchange with the platform for any simulation step.
To perform the PIL simulation for a part of the model, this part is first encapsulated in a subsystem. The user then generates the code for the aforementioned subsystem. To do this, the user first inform the STF that the code generation for the block should target PIL simulation, through: **Configuration Parameters> Code Generation> Advanced Parameters> Create Block> PIL.** Next, the user builds the subsystem (by right click on the subsystem> C/C++ code> Build this subsystem). The STF starts generating the code for the subsystem and results a new PIL block. The initial subsystem is then replaced by the resulted PIL block. The next steps of performing the PIL simulation is the same as the Section 6.1.1.2 with the only difference that in this case, the simulation mode changes to *normal*.

6.1.1.2 HIL Simulations in Simulink

Once both PIL configurations are used to validate the setup, it can be validated against the actual hardware, or an accurate model, using the Hardware in the Loop (HIL) setup. This step is used to check the drivers, the timing of the drivers and timing of communicating with the actuators and the sensors. It is also used to validate the correct working of the controller against the actual hardware. This setup will run at the actual sample frequency.
In this configuration, Matlab® Simulink® does not drive the configuration, the setup runs standalone and self-timed. Controlling and logging is done via the High-level user interface (gRPC).
The configuration is meant to address the following questions:

- Controller code execution is firm real-time on the target platform.
- Functional verification
- Logging of signal and errors
- Tuning the simulation parameters via High-level user interface

6.1.1.2.1 External mode

To demonstrate the HIL abilities of the STF, we perform *external mode* simulation in Simulink. The external mode simulation is similar to PIL simulation. The difference is that in *external mode* Simulink® does not drive the simulation and the executable generated out of the model runs as a standalone on the target platform.
To perform the external mode simulation, the user first chooses the provided STF using the similar instruction described in Section 6.1.1. Next, the user should specify that code generation should target external mode. To do this the external mode should be enabled through: **Configuration Parameters> Code generation> Interface> External mode**.
The external mode enables the user to tune the model parameters while the simulation is running on the target platform. To enable this, the parameters behavior should change from inlined (default) to tunable. This can be done through: **Configuration Parameters> Code generation> Optimization> Default parameter behavior> Tunable**.

6.1.1.2.2 Performing external mode simulation

Now that all the parameters are set, the user can perform an external mode simulation on the target platform. To do this, the user changes the simulation mode to *external mode*, and hit the build button. In the background (where the steps can be traced through *diagnostic viewer*) the STF first generates the code for the model with the usage of Mathworks embedded coder. When the code is generated, the STF continues the process by compiling the generated code and building the executable. At the end of build, the STF reports the size of the executable and in the case the size of the executable is bigger than the target memory, is halts the simulation and gives an error, reporting the issue.
If the executable fits in the target memory, the STF uploads the executable on the target platform and starts the simulation.
In order to connect to the board to trace the data and tune the parameter, the user hit the connect button (which replaced the simulation run button). Doing this, commands the STF to start the communication with the hardware through an initial handshaking. When the handshaking is successful, the executable running on the board and Simulink are connected. Within the simulation, the user can track the output signals (via simulation sink blocks such as *Scope*) as well as tuning the parameters (via blocks parameters).
To stop the simulation on the target, the user hits the stop button.

6.2 Implementation aspects

6.2.1 Pilot 1

The implementation targeted with BB11 is Pilot 1. The implementation is defined in an integration process where the BB functionalities are presented to the pilot owner and validated on the pilot.
At the time of writing this report, the integration of the BB with Pilot 1 is ongoing. Through the integration meeting, the BB functionalities initially were presented to the pilot owner. The implementation of the BB on Pilot 1 was then decided to perform the steps proposed in Figure 45.

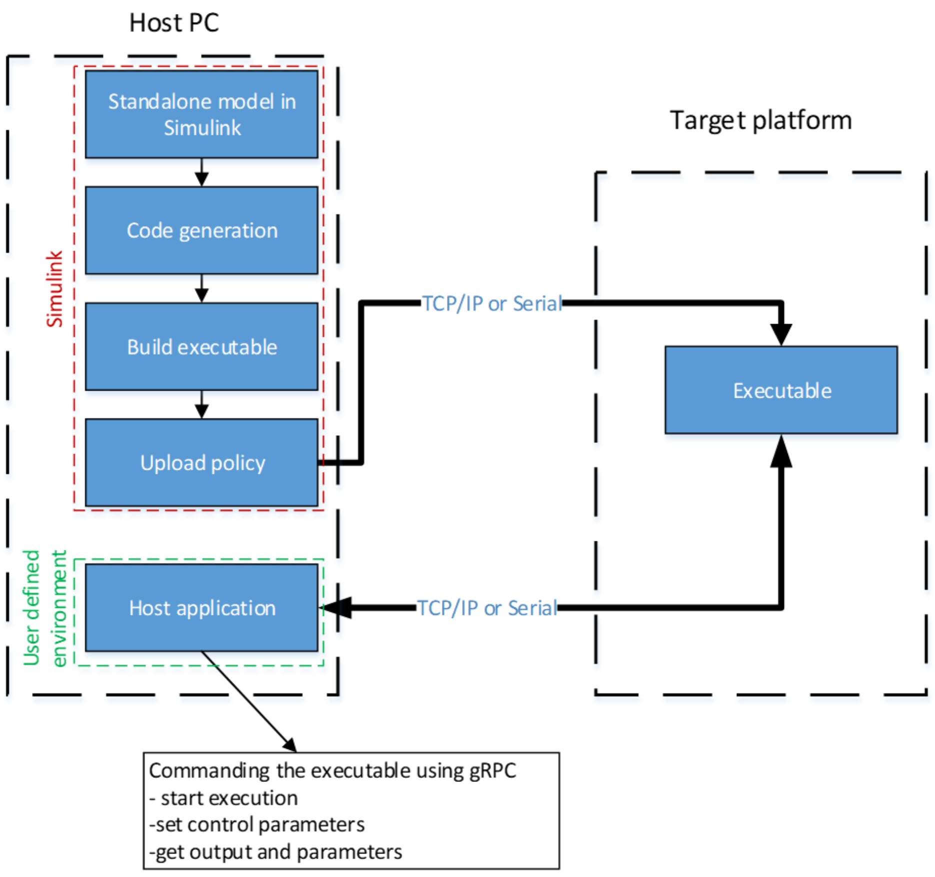


Figure 45: The integration steps of BB11 with Pilot 1.

The integration in the Code generation, Build executable, upload policy, and executing on the platform are done. The remaining step is "host application" where the user can use the data tracing and parameter tuning via a host application else than Simulink. By achieving this step the integration of the BB with layer 3 (gRPC framework) is validated.

6.3 COTS platform

6.3.1 Functionalities

BB11 for COTS platform consisted in the design/development of a hypervisor technology allowing the concurrent execution of multiple operating systems ("Multi-OS) with different levels of criticality (see Figure 46).

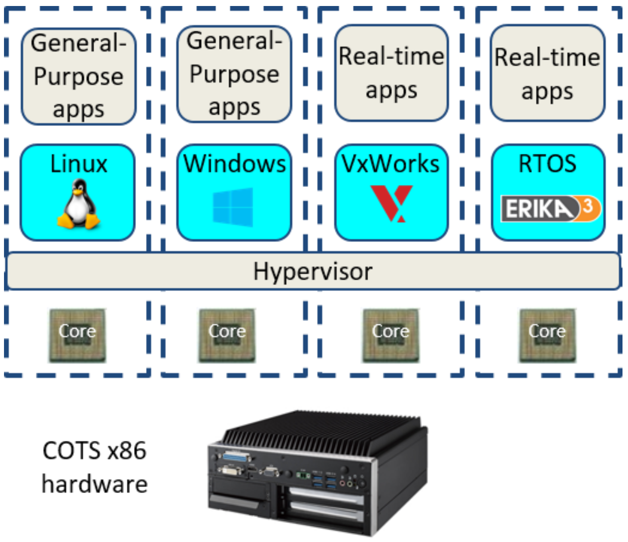


Figure 46: Hypervisor technology scheme.

In particular, the work aimed at running both the traditional general-purpose operating systems with model-based design tools (e.g. Windows or Linux) and real-time operating systems (e.g. VxWorks, ERIKA Enterprise) on the same platform at the same time. The challenge has been the design of proper isolation for preventing the former operating systems from interfering in the real-time control performance.
The features provided by the implementation are:

- Running on COTS x86 multi-core hardware (i.e. no specific hardware requested)
- Concurrent execution of multiple operating systems with different criticality through hypervisor technology
- Real-time performance of the control part
- Open and modular API (based on Google's gRPC) for interfacing with the system
- Integration with model-based design tools (i.e. Simulink) for automatic code generation

6.3.2 Implementation aspects

At the lowest level, the software architecture consists of a hypervisor for partitioning the hardware resources among the different operating systems. The selection of the specific hypervisor has been topic of a careful investigation. Among the possible open-source hypervisors (namely: Jailhouse[1], Xen[2], KVM[3]), BB11 selected Xen for the following reasons:

- It allows to run unmodified operating systems (e.g. Windows)
- The latest versions feature the "null scheduler" for real-time performance.

The hypervisor provided a high degree of flexibility for adapting this solution through different configurations. The first configuration (shown in the next Figure) consisted of the Windows operating system running the Simulink design tool, and the ERIKA Enterprise RTOS[4] running the control code automatically generated from Simulink. To reach this objective, the following components have been developed (see Figure 47):

- API based on Google's gRPC technology[5] for interfacing the ERIKA part with the Simulink tool (or with ad-hoc tools for inspecting and monitoring the execution)

[1] https://github.com/siemens/jailhouse
[2] https://xenproject.org/
[3] https://www.linux-kvm.org/page/Main_Page
[4] https://www.erika-enterprise.com/
[5] https://grpc.io/

- Driver for Intel i210 Ethernet network card for ERIKA Enterprise
- SOEM open-source EtherCAT master stack ported on ERIKA for controlling EtherCAT slaves
- Simulink support for automatic code generation and deployment through the gRPC API

A demo of this configuration is available on YouTube: http://y2u.be/jtVdTYgxZU4.
Proper installation instructions about how to configure the system have been made available on ERIKA's wiki pages: http://www.erika-enterprise.com/wiki/index.php?title=EtherCAT_Master

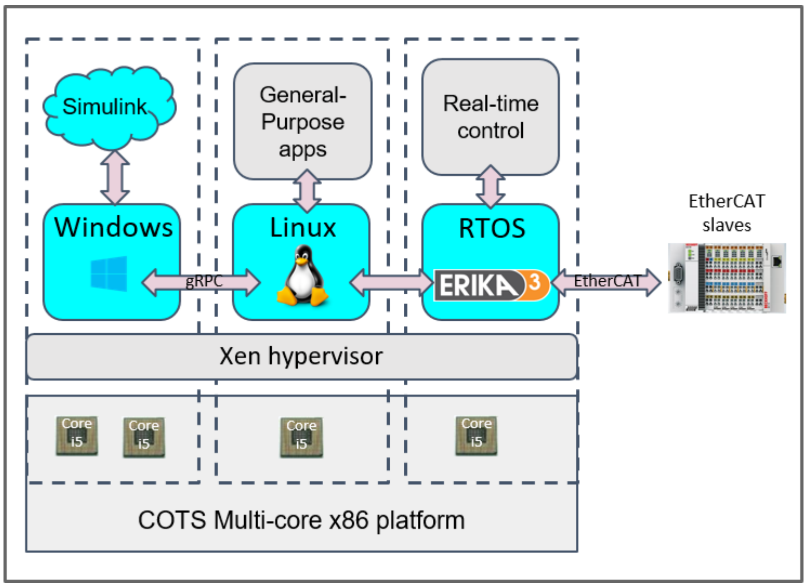


Figure 47: First configuration of the considered hypervisor technology.

The second configuration (shown in Figure 48) was developed for supporting Pilot 3 by IMA, and it consists of multiple instances of Windriver VxWorks 6.9 executing concurrently on the hypervisor. The work consisted in porting the VxWorks BSP on top of the machine exposed by the Xen hypervisor.

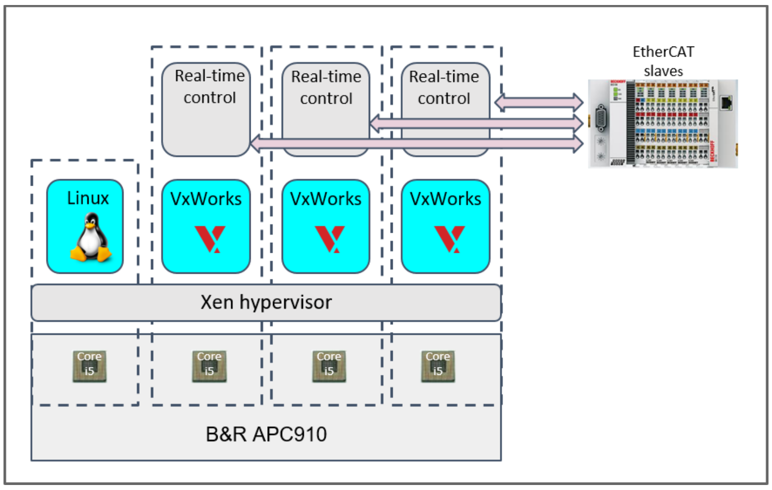


Figure 48: Second configuration of the considered hypervisor technology.

7 Conclusions

7.1 General conclusion remarks

The I-MECH project aims at the development of smart mechatronics systems by means of hardware and software Building Blocks. The Building Blocks have been developed to work within the I-MECH platform, but most of them can be easily adapted to brown field applications. The Building Blocks developed in the framework of the Work Package 4 are all located in the control layer, that is the layer 2, of the I-MECH platform, and they can help in the development of smart controllers for new, or already existing, mechatronic systems.
These Building Blocks can work as standalone blocks or joint to other Building Blocks developed in the I-MECH project. They have been developed in MATLAB/Simulink, but FMUs as well as C/C++ blocks can be easily obtained by means of Code Generation applications.
The implementation and integration of the Building Blocks developed in WP4 in Pilots, Use cases and Demonstrators is in progress and this activity will continue until the end of the project

7.2 Contribution beyond the state of the art

The contribution beyond the state of the art of the Building Blocks developed inside the Work Package 4 has been shown in this document. The BB7 (Vibration control in mechanically compliant systems) that has been developed in Task 4.4 of WP4, brings innovative solution in the control of mechatronics systems affected by vibrations and/or oscillations (example of these systems are over-head cranes, ,etc.). The BB8 (Robust motion control strategies) that has been developed in Task 4.5 of WP4, brings solution from the university world to the industrial field related to the robust tuning strategies for MIMO systems. The BB9 (Iterative and repetitive control methods) that has been developed in Task 4.6 of WP4, brings new solution related to the control of mechatronics systems affected by repetitive and/or periodic disturbances, as well as systems that must perform repetitive tasks. Finally, BB11 (Advanced motion control algorithms and software for predictable multi-many core platforms), that has been developed in Task 4.7 of WP4, brings up-to-date solutions for the management and control of systems by using multi-many cores platforms.
The Building Blocks mentioned above have been validated by following a procedure defined inside the I-MECH project that consists of a series of steps to reach the goal of a performing BB. The validation process starts from a MIL step to reach a HIL final step (intermediate steps can be SIL and PIL) before to be integrated in Pilots, Demonstrators or Use Cases.

7.3 Dissemination and exploitation

The work that has been done in Work Package 4 brought to several scientific publications that are listed in this document. Videos prepared by BB7, BB8, BB9 and BB11 owners have been shown during the ECSEL Symposium in June 2019. The I-MECH Newsletter (https://www.i-mech.eu/publications/dissemination-material/) is released every three months and it shows the status of the work for each BB. Seminars and Webinars will be done before the end of the project to show the techniques developed in the framework of the I.MECH project. The dissemination and exploitation work will work until the end of the project and it is managed by WP8.

8 Bibliography

Number	Reference
[1]	M. Giacomelli, F. Padula, L. Simoni, A. Visioli, "Simplified input-output inversion control of a double pendulum overhead crane for residual oscillations reduction", Mechatronics, Vol. 56, pp. 37-47, 2018.
[2]	M. Giacomelli, M. Faroni, D. Gorni, A. Marini, L. Simoni, A. Visioli, “Model predictive control for operator-in-the-loop overhead cranes”, 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 589-596, 2018.
[3]	M. Giacomelli, D. Colombo, M. Faroni, O. Schmidt, L. Simoni, A. Visioli, “Comparison of linear and nonlinear MPC on Operator-In-the-Loop overhead cranes”, 7th International Conference on Control, Mechatronics and Automation, Delft (NL), 2019.
[4]	M. Giacomelli, M. Faroni, D. Gorni, A. Marini, L. Simoni, A. Visioli, “MPC-PID control of operator-in-the-loop overhead cranes: a practical approach”, 7th International Conference on Systems and Control, Valencia (E), 2018.
[5]	M. Schlegel, M. Goubej, “Feature-based parameterization of input shaping filters with time delays”, 9th IFAC Workshop on Time Delay Systems, 2010
[6]	M. Goubej, “Robust control of flexible electromechanical systems”, PhD thesis, University of West Bohemia, 2014
[7]	M. Goubej, “Fundamental performance limitations in PID controlled elastic drive systems”, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2016.
[8]	M. Goubej, V. Helma, “Vibration damping in gantry crane systems: Finite horizon optimal control approach”, IEEE International Conference on Emerging Technologies and Factory Automation, 2019.
[9]	M. Goubej, T. Vyhlidal, M. Schlegel, “Frequency-weighted H2 optimization of multi-mode input shaper”, submitted to Automatica, 2019
[10]	I.R. de Argandona, M. Zatarain, A. Illarramendi, J.L. Azpeitia, “Improvement of the performance in machine tools by means of state space control strategies”, IEEE Conference on Decision and Control, 2005.
[11]	M. Goubej, Fundamental performance limitations in PID controlled elastic drive systems”, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2016.
[12]	V. Helma, M. Goubej, O. Ježek, “Acceleration Feedback in PID controlled elastic drive systems”, IFAC Conference on Advances in PID Control, 2018.
[13]	V. Helma, M. Goubej, “Acceleration Feedback in PID Controlled Elastic Drive Systems”, 3rd IFAC Conference on Advances in PID Control, 2018.
[14]	M. Schlegel, P. Medvecová, “Design of PI Controllers: H-infinity region approach“, 15th IFAC Conference on Programmable devices and Embedded Systems, 2018.
[15]	M. Goubej, M. Schlegel, “PID plus Repetitive Control Design: H-infinity regions approach“, 22nd IEEE Conference on Process Control, 2019.
[16]	M. Goubej, M. Schlegel, T. Vyhlídal, “Robust controller design for feedback architectures with signal shapers“, submitted to IFAC World Congress 2020.
[17]	N. Mooren, G. Witvoet, T. Oomen, “From batch-to-batch to online learning control: Experimental motion control case study”, Joint Conference 8th IFAC Symposium on Mechatronic Systems and 11th IFAC Symposium on Nonlinear Control Systems, 2019.
[18]	N. Mooren, G. Witvoet, T. Oomen, “Feedforward motion control: From batch-to-batch to online parameter estimation”, American Control Conference, pp. 947-952, 2019.
[19]	J. Madariaga, L. G. Uriarte, I. Ruíz de Argandoña, J. L. Azpeitia and J. C. Rodríguez de Yurre, “Machine tool tracking error reduction in complex trajectories through anticipatory ILC”, Spring ASPE Topical Meeting: Control of Precision Systems, 2010.
[20]	M. Goubej, M. Schlegel, “PI plus RC design: H-infinity regions approach”, IEEE International Conference on Process Control, 2019.

[21]	M. Schlegel, P. Medvecová, “Design of PID Controllers: H-infinity regions approach”, 15th IFAC Conference on Programmable Devices and Embedded Systems, 2018.
[22]	D. A. Bristow, M. Tharayil, A. G. Alleyne, “A survey of iterative learning control”, IEEE control systems magazine, Vol. 26, pp. 96-114, 2006.
[23]	J. Bolder, T. Oomen, “Rational basis functions in iterative learning control - With experimental verification on a motion system”, IEEE Transactions on Control Systems Technology, Vol. 23, 722-729, 2015.
[24]	J. van de Wijdeven, O. H. Bosgra, “Using basis functions in iterative learning control: analysis and design theory”, International Journal of Control, Vol. 83, 661-675, 2010.
[25]	M. Goubej, S. Meeusen, N. Mooren, T. Oomen, “Iterative Learning Control in high-performance motion systems: From theory to implementation”, 24th IEEE Conference on Emerging Technologies and Factory Automation, 2019.
[26]	N. Mooren, G. Witvoet, I. Acan, J. Kooijman, T. Oomen, “Suppressing Position-Dependent disturbances in repetitive control: With applications to a substrate carrier system”, Submitted.

Acknowledgement

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737453. This Joint Undertaking receives support from the European Union’s Horizon 2020 research and innovation program and Netherlands, Czech Republic, Latvia, Spain, Greece, Portugal, Belgium, Italy, France, Ireland