Universidad de Oviedo

Department of Electrical, Electronics,
Communications and Systems Engineering

Ph.D. Thesis
PhD Program in Energy and Process Control

"Mechatronic co-simulation and implementation of control strategies to assure the stability of a small footprint mobile robot system"

Aishe Toledo Fuentes
Born in El Espinal, Oaxaca, Mexico

Universidad de Oviedo

Department of Electrical, Electronics,
Communications and Systems Engineering

Ph.D. Thesis dissertation

"Mechatronic co-simulation and implementation of control strategies to assure the stability of a small footprint mobile robot system"

A dissertation submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in the PhD program in Energy and Process Control of the University of Oviedo, with International Mention

Aishe Toledo Fuentes

Directors:
Prof. Dr.-Ing. Martin Kipfmüller
Karlsruhe University of Applied Sciences, Germany

Prof. Dr. Miguel Ángel José Prieto
University of Oviedo, Gijón, Spain

Royal Oak, United States of America, November 2022.

Universidad de Oviedo

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: **Co-simulación mecatrónica e implementación de estrategias de control que aseguren la estabilidad de un sistema robótico móvil compacto**	Inglés: **Mechatronic co-simulation and implementation of control strategies to assure the stability of a small footprint mobile robot system**

2.- Autor	
Nombre: **Aishe Toledo Fuentes**	
Programa de Doctorado: **Energía y Control de Procesos**	
Órgano responsable: **Centro Internacional de Postgrado**	

RESUMEN (en español)

Los manipuladores móviles de estructura ligera ofrecen gran flexibilidad, agilidad y maniobrabilidad, no obstante, tienden a volcarse, sobre todo durante procesos bruscos de frenado.
El presente trabajo trata dos enfoques diferentes que ayudan a solucionar dicho problema de inestabilidad, los cuales difieren entre sí en el tipo de sistema al que serán aplicados: manipuladores móviles con sistema operativos de código cerrado o de código abierto, respectivamente.

En el primer enfoque se describen tres estrategias de estabilización que compensan momentos de inestabilidad con la ayuda de mecanismos de actuadores externos.
El primero de los mecanismos propuestos se compone de actuadores lineales en configuración delta, integrados entre la plataforma móvil y el robot manipulador. En la estrategia de estabilización "inclinación", el robot manipulador es inclinado en la dirección opuesta al desplazamiento de la plataforma móvil antes de que comience el proceso de frenado, con el fin de desplazar hacia atrás el centro de gravedad del robot manipulador. Por el contrario, la estrategia de estabilización "conservación del momento angular" impulsa al robot manipulador en la misma dirección del desplazamiento de la plataforma móvil durante el proceso de frenado, generando momentos angulares de compensación.
El segundo mecanismo consigue un efecto similar, pero esta vez adoptando un estabilizador giroscópico como actuador.
Para el dimensionamiento y evaluación de las tres estrategias de estabilización se ha propuesto un entorno de co-simulaciones mecatrónicas, basados en modelos de simulación multicuerpo (MBS). Para ello, se emplean resultados obtenidos de análisis modales experimentales (EMA) de la plataforma móvil y el robot manipulador real para parametrizar y validar sus modelos MBS, de forma que éstos reproduzcan su comportamiento dinámico real.
El principal reto del modelado MBS es estimar los coeficientes de rigidez y amortiguamiento de los elementos de unión del sistema, necesarios para obtener los parámetros modales reales deseados. El largo ajuste iterativo manual de dichos coeficientes fue mejorado mediante un algoritmo de parametrización automatizado.
Este primer enfoque fue evaluado en un manipulador móvil real con sistema operativo de código cerrado. El procedimiento presentado proporciona una de las primeras investigaciones sobre el modelado completo de co-simulaciones mecatrónicas de manipuladores robóticos y plataformas móviles: desde la identificación de sus parámetros modales mediante EMAs, hasta

su modelado y parametrización como sistemas MBS.
Las co-simulaciones mecatrónicas demostraron que las estrategias propuestas mejoran la estabilidad del manipulador móvil, incluso cuando grandes aceleraciones y desaceleraciones afectan al sistema. Cabe destacar que la técnica "inclinación" implica la predicción del perfil de frenado, lo que dificulta su implementación en un bucle de control cerrado. Por otro lado, el método basado en la “conservación del momento angular" puede integrarse en un bucle de control cerrado, pero su impacto en la mejora de estabilidad del sistema ha sido menor. Por último, el estabilizador giroscópico mostró gran potencial al generar la mayor compensación contra las inestabilidades.
Aunque el presente estudio se implementó en robots particulares, ofrece una visión útil de la metodología para otros sistemas complejos similares.

El segundo enfoque comprende la optimización del espacio de trabajo del manipulador móvil y el reposicionamiento de las articulaciones del robot manipulador.
El valor de la estabilidad dinámica para la detección del vuelco se estimó mediante el método *Moment-Height Stability Measurement* (MHS), que considera el efecto de las fuerzas internas del robot manipulador y proporciona información sobre el grado de estabilidad.
Con el fin de restringir el volumen máximo en el que el manipulador móvil puede operar sin riesgos de inestabilidad, el espacio de trabajo teórico del robot manipulador se optimizó en base al valor de estabilidad. Adicionalmente, se concibió un algoritmo que reposiciona los eslabones del robot manipulador cuando el valor de estabilidad está por debajo del valor crítico definido. Dicho algoritmo determina la nueva configuración del robot manipulador mediante un método de gradiente, también dependiente del valor de estabilidad.
Este enfoque se aplicó a un segundo manipulador móvil con sistema operativo de código abierto. Los algoritmos se implementaron en el entorno ROS y se validaron en el sistema real. Pruebas realizadas con el manipulador móvil (real y en simulaciones) indicaron que la estrategia de estabilización es capaz de evitar su vuelco bajo diferentes escenarios.
Quizás el valor añadido más significativo de esta estrategia de estabilización respecto a trabajos anteriores, es que se ejecuta en tiempo real y sin manipular el control de movimiento de los robots, reduciendo así la complejidad para su implementación y aumentando su flexibilidad como solución universal.

RESUMEN (en inglés)

Lightweight mobile manipulators offer great flexibility, agility and maneuverability, however, they tend to tip over especially during abrupt braking processes.
This work describes two different approaches to deal with this instability problem, differing from each other in the system on which they will be implemented: mobile manipulators with closed-source or open-source operating systems, respectively.

The first approach presents three stabilization strategies that compensate instability moments with the help of external actuator mechanisms.
The first of the proposed mechanisms consists of linear actuators in delta configuration, integrated between the mobile platform and the robot manipulator. During the "inclining/tilting" stabilization strategy, the robot manipulator is tilted in the opposite direction of displacement of the mobile platform before the braking process starts, in order to shift back the robot manipulator’s center of gravity. In contrast, the "conservation of angular momentum" stabilization strategy impels the robot manipulator in the same direction of displacement of the mobile platform during the braking process, thus, generating beneficial angular moments.

The second mechanism attains a similar effect, but this time adopting a gyroscopic stabilizer as external actuator.
For the dimensioning and evaluation of the three stabilization strategies, mechatronic co-simulation's environments based on multibody system (MBS) models have been proposed. In this respect, results obtained from experimental modal analyses (EMA) of the real mobile platform and robot manipulator are employed to parameterize and validate their MBS models, so that they reproduce accurately their real dynamic behavior. The main challenge of modeling the mobile manipulator as MBS is to estimate the stiffness and damping coefficients of its joining elements, since they are essential to obtain the desired real modal parameters from the EMAs. An automated parameterization algorithm improved the manual time-consuming iterative adjustment of these coefficients.
The first approach was implemented and evaluated on a real mobile manipulator with closed-source operating system. This procedure provides one of the first investigations on the entire modeling of mechatronic co-simulations of robot manipulators and mobile platforms: from the identification of their modal parameters by means of EMAs, to their modeling and parameterization as MBS systems.
The mechatronic co-simulations demonstrated that the three stabilization strategies improved the stability of the mobile manipulator, even when large accelerations and decelerations affected the system. However, it should be denoted that the "Inclining/tilting" technique implies the prediction of the braking profile of the mobile platform, which makes it difficult to implement in a closed control loop. On the other hand, the method based on "conservation of angular momentum" can be integrated in a closed control loop, but its impact on the improvement of the system stability has been limited. Lastly, the gyroscopic stabilizer showed great potential by generating the largest compensation against instabilities.
Although the present study was implemented on particular robots, it provides a useful insight into the applied methodology for other similar mobile manipulators.

The second approach comprises the optimization of the mobile manipulator workspace and the repositioning of the robot manipulator joints.
The dynamic stability value for the tip-over detection was estimated using the Moment-Height Stability Measurement (MHS) method, which considers the effect of the internal forces of the robot manipulator and provides information about its degree of stability.
In order to restrict the maximum volume in which the robot manipulator can operate without risks of instability, its theoretical workspace was optimized based on the computed stability value. Additionally, an algorithm repositions the links of the robot manipulator when the stability value lies below a defined critical value. This algorithm determines the new configuration of the robot manipulator using a gradient method, which is also dependent on the stability value.
The approach was implemented in a second mobile manipulator with open-source operating system. All algorithms were processed in ROS environment and validated on the real system. Tests performed on the mobile manipulator (real and in simulations) indicated that the stabilization strategy was able to avoid tip-over under different scenarios.
Perhaps the most significant added value of this stabilization strategy with respect to previous work is its execution in real time and without manipulating the motion control of the robots, thus reducing the complexity for its implementation and increasing its flexibility as a universal solution.

SR. PRESIDENTE DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO EN ___ ENERGÍA Y CONTROL DE PROCESOS _____

Acknowledgements

First and foremost, I would like to give special thanks to my husband, Tobias Jäger, for his constant advice, unconditional support, patience and all the huge personal sacrifices he took in order to make this aspiration a reality.

In particular, I want to thank my supervisors, Professor Martin Kipfmüller and Professor Miguel Ángel José Prieto, for their invaluable continuous guidance, help and support. I wish to extend my thanks to Marcel Mohr, Volker Portje, Stefanie Seemann, Maximilian Bryg, Thomas Bertram, Tobias Bergmann, Franziska Kempf and each of the participants in this study, professors, students and colleagues of the Karlsruhe University of Applied Science, who helped me to arrive at this point.

I am also deeply grateful to Ekiñe Aristizabal Tolosa for her unfailing source of encouragement and her constructive contributions to this thesis.

To my family.
Ndi xti bixhoze bida, Don Joel.

Abstract

Mobile manipulators attain highly automated and flexible production facilities. They are designed for handling objects with different weights and sizes efficiently, without endangering production workers and machines. Their ideal requirements, such as having an optimal height for good reachability, light weight for good dynamics and small footprint for profitability, lead to one of their fundamental problems: mobile manipulators tend to tip over, most notably when their mobile platform suddenly brakes/accelerates. The compensation of forces affecting the system's dynamics is therefore needed to avoid unwanted tip-over.

Standard mobile manipulators are comprised by robot manipulators mounted on relatively bulky autonomous mobile platforms. They reach their driving stability through their own weight, by employing a comparatively large footprint to *passively* keep its stable driving behavior during their operation. However, mobile platforms with large footprint are associated with high costs, because they demand additional space of the generally limited layout of the production plant. Consequently, a cost-effective stable driving dynamic is best achieved with a mobile manipulator that, regardless of its high system center of gravity, has a small footprint. This idea led to the aim of the present research: assessing different stabilization strategies to reduce the risk of tip-over of small footprint mobile manipulators.

Due to the fact that most of the robot manufacturers supply their products with restricted access to their robot operating system, the first treated approach focused on upgrading those systems with an external stabilization mechanism. The utilization of a mechanism conformed by linear drives and another one based on the gyro effect were considered. The development and evaluation of the stabilization strategies were carried out using multibody-system simulations. During the development, the experimental determination of the system modal parameters played an important role, since they enabled the creation of close-to-reality simulation models.

The second approach focused on a tip-over detection and avoidance algorithm, designed for those mobile manipulators equipped with an open-access robot operating system. A tip-over detection algorithm based on the Moment Height Stability method was implemented to indicate how stable/unstable the system is during its operation/configuration and, thus, to effectively react against instability states. The theoretical workspace of the robot manipulator was also realigned by a calculated critical workspace boundary surface based on the analysis of the stability value. Then, in order to prevent a tip-over of the mobile manipulator during its navigation in the space, a tip-over avoidance algorithm was developed in a close-to-reality simulation environment. This algorithm repositioned the joints of the robot manipulator in real time, adopting a conveniently computed configuration to compensate the instabilities detected by the Moment Height Stability method. During the repositioning process, the initial orientation of the robot Tool Center Point was maintained to avoid a work piece mishandle.

Both stabilization approaches presented in this thesis can contribute to effectively accomplish a compact autonomous and stable mobile manipulator, capable of operating in real production environments.

Table of contents

List of symbols and abbreviations

Abbreviations

ADAMS	*Automatic Dynamic Analysis of Mechanical Systems*
AGV	*Automated Guided Vehicles*
AMR	*Autonomous Mobile Robots*
BDF	*Backward Differentiation Formula*
CAD	*Computer Aided Design*
Co-bot	*Collaborative Robot*
COG	*Center of gravity*
CSS	*Continuous System Simulation*
DAE	*Differential and Algebraic Equation*
DAKOTA	*Design Analysis Kit for Optimization and Terascale Applications*
DC	*Direct Current*
DOF	*Degree(s) of Freedom*
EMA	*Experimental Modal Analysis*
FA	*Force Angle Stability Measurement*
FE	*Finite Element*
FEM	*Finite-Element-Method*
FFT	*Fast Fourier Transformation*
FRF	*Frecuency Response Function*
HIL	*Hardware-in-the-loop*
HMI	*Human-Machine Interface*
HRI	*Human-Robot Interaction*
IMU	*Inertial Measurement Unit*
IT	*Information Technology*
ITOMSC	*Tip-Over Moment Stability Criterion*
MAC	*Modal Assurance Criterion*
MATLAB	*MATrix LABoratory*
MBD	*Model-Based Design*
MBS	Multibody-system Simulation
MHS	*Moment Height Stability method*
MIRA	*Middleware for Robotic Applications*
OMA	*Operational Modal Analysis*
PD-Controller	*Proportional–Derivative controller*
PI-Controller	*Proportional–Integral controller*
PID-Controller	*Proportional–Integral–Derivative controller*
RIA	*Robotic Industries Association*
RMC	*Robot Main Control*
RNEA	*Recursive Newton-Euler Algorithm*
ROS	*Robotic Operating System*
RViz	*ROS Visualization*
SRP/CS	*Safety-Related Parts of Controls Systems*
STL	*STereoLithography*
STRANDS	S*patiotemporal Representations and Activities for Cognitive Control in Long-Term Scenarios*
TCP	*Tool Center Point*
URDF	*Unified Robotic Description Format*

WOBSC	*Whole Body Operational Space*
XML	*Extensible Markup Language*
ZMP	*Zero Moment Point*

Latin symbols

A		Amplitude response
A_1, B_1		Constants (vibration engineering)
$\vec{a}_i$	rad/s^2	Acceleration vector, i=1,2,3
$\hat{a}_i$		Unit vector of acceleration in $\mathbb{R}^3$, i=1,2,3
$\vec{b}_i$	mm	Vector from robot origin to linear drive i, i=1,2,3
C_{obs}		Obstacle configuration space in $\mathbb{R}^3$
COG_x, COG_y, COG_z	m	Center of gravity in x,y,z
c, c_i		Damping coefficient
d_i	m	Lever arm with rotating point i, i=1,2,…,n
d_f	m	Distance between the system COG and one of the front wheels
e		Error value in close control loops
$\hat{e}_i$		Unit vector for connecting lines of tilting edges i, i=1,2,…,n
$\vec{e}_i$		Tilting shape, i=right, left, front
$\vec{F}$	N	Force (general)
$\vec{F}_0$	N	Force at equilibrium position
$\vec{F}_i$, $\vec{F}_j$	N	Force vector, i=1,2,…,n
$\vec{F}_r$	N	Reaction force
$\vec{F}_{ex}$	N	External force vector
$\vec{F}_{ee}$	N	Force acting on robot end effector
$\vec{F}_I$	N	Inertial force
$\vec{F}_g$	N	Gravitational force
$\vec{F}_{FA}$	N	Force Angle Stability Measurement
$F(\omega)$		Frequency response input magnitude
r_{max}, r_{theo}, r_{opt},	m	Maximum, theoretical and optimized radius
$\vec{F}_i^B$	N	Reaction force vector, i=1,2,…,n
$\vec{F}_i^*$	N	Resulting total force regarding the tilting edge i, i=1,2,…,n
$G_C(s)$, $G_M(s)$		Transfer function in frequency domain
G_C, G_M		Transfer function in time domain
$\vec{g}_x, \vec{g}_y, \vec{g}_z$	m/s^2	Gravitacional vector in x,y,z
H_1, H_2		Estimators
H_{mi}		Frequency response at measurement point i, i=1,2,…,n
H_{ie}		Frequency response at excitation point i, i=1,2,…,n
$\mathbf{H(\omega)}$		Frequency response
h_i	m	Height, i=1,2,…,n
I_i	kg·m^2	6x6 Inertia matrix of joint i=1,2,…,n
$\vec{I}_{vi}$	N·m	Moment of inertial regarding the vertex i of the tilting polygon, i=1,2,…,n
$Im[\ldots]$		Imaginary part of …
K_M		Motor gain constant

K_C		Controller gain constant
k, k_i	N·m, N·mm	Stiffness coefficient
K_I		Integral gain
K_P		Proportional gain
K_{PI}		Proportional-Integral gain
K_{PD}		Proportional-Derivative gain
K_s		Amplification gain
$\vec{L}$	kg·m^2/s	Angular momentum
$\vec{l}_i$	m	Distance vector, i=1,2,…,n
$\vec{l}_i^{\,B}$	mm	Vector that describes the trapezoidal threaded i, i=1,2,3
$\vec{M}_i$	N·m	Moment vector, i=1,2,…,n
$\vec{M}_A$	N·m	Moment vector regarding the reference point A
$\vec{M}_S$	N·m	Tilting moment
$\vec{M}_{ee}$	N·m	Moment vector acting on robot manipulator end effector
$\vec{M}_{ex}$	N·m	External moment vector
$\vec{M}_r$	N·m	Reaction moment
$\vec{M}_I$	N·m	Inertial moments
$\vec{M}_{vi}$	N·m	Moment regarding the vertex i of the tilting shape, i=1,2,…,n
$\vec{M}_{eq}$	N·m	Moment of equilibrium
$\vec{M}_p$	N·m	Precession moment
$\vec{M}_c$	N·m	Compensation moment from precession effect
$\vec{M}_d$	N·m	Disturbance moment from precession effect
$\vec{M}_{dyn}$	N·m	Moment obtained from dynamic analysis
$\vec{M}_{Stat}$	N·m	Moment obtained from static analysis
$\vec{M}_{i,T}$	N·m	Total moment acting on each tilting shape, i=1,2,3
$\vec{M}_{i,Basis}$	N·m	Total moment generated by the mobile platform and acting on each tilting shape, i=1,2,3
$\vec{M}_4$	N·m	Generated moment around the 4th joint of the manipulator
m, m_i	kg	Mass, i=1,2,…,n
n_{max}	min^{-1}	Maximal motor speed
n_{motor}	min^{-1}	Motor speed
n		Number of degrees of freedom
$\vec{o}_i$	mm	Vector to linear drive i with origin at the platform, i=1,2,3
P		Point of division of subsystems
P_{Origin}, P_{Target}		Origin and target pose in $\mathbb{R}^3$
$\vec{p}_i$	m	Position vector in $\mathbb{R}^3$, i=1,2,…,n
p_i, p_c		Point i or regarding the COG in $\mathbb{R}^3$, i=1,2,…,n
O_{Origin}, P_{Target}		Origin and target orientation in $\mathbb{R}^3$
q_i	rad	Robot manipulator joint variable, i=1,2,…,7
Re[…]		Real part of …
$\vec{r}_I = [x_i, y_i, z_i]$	m	Position vector of body i COG
S		Reference point for the calculation of tilting moment
S_i	rad	Matrix for motion range of robot manipulator link i, i=1,2,…,n
$\vec{S}_{ex}$	m	Point of application of external forces

$\overrightarrow{TCP_{mp}}$	m	Vector from the world coordinate system to the mobile platform
T_1	s	1st Time constant
T_2	s	2nd Time constant
T_Σ	s	Sum of time constants
T_V	s	Controller derivative time
T_p	s	Period duration of step function response
T_N	s	Controller reset time
u_i	m	Position of linear drive *i, i*=1,2,3
$\vec{v}_i$	m/s	Velocity vector, *i*=1,2,…,n
x	m	Displacement
x_0	m	Initial displacement (at t=0)
x_n		Filter input variable regarding discrete time (n)
$\mathbf{X}$		Eigenvector matrix
x,x_0,y,y_0,z,z_0		Coordinates in $\mathbb{R}^3$ space
$\vec{x}_{mp}$	m	Direction vector of the mobile platform
$X(\omega)$		Frequency response output magnitude
${}^iX_{\lambda(i)}$, ${}^iX_j^*$,		Matrixes for coordinate transformations
y_n, y_{n-1}		Filter output variable regarding discrete time (n)

Greek symbols

α		Stability value
$\alpha_{i\text{-}cm}$		Stability value of entire system COG regarding the tilting axis *i, i*=1,2,3
α_{max}		Current maximal stability value
$\alpha_{cm\text{-}critical}$		Critical stability value
β	rad	Angle of 2nd joint of the manipulator
γ		Coefficient (-1,…,+1)
γ^2		Coherence
Δ		Variation of a variable or function
θ	rad, grad	Angle
η		Frequency ratio
η_F		Filter coefficient
Θ	$kg \cdot m^2$	Moment of inertia
λ		Eigenvalue
$\lambda(i)$		Parent of link current link *I, i*=0,1,2,…7
$\mu(i)$		Children of link current link *I, i*=0,1,2,…7
Ω	1/s	Excitation frequency
ς		Damping ratio
σ		Coefficient (-1,…,+1)
Γ	rad	Angle for the position of the TCP relative to the mobile platform.
Υ	rad	Angle of 3rd joint of the manipulator
$\vec{\Upsilon}_i$	N·m	Torque acting on link *i, i*=0,1,2,…,7
φ	rad, grad	Phase response
φ_ω	rad, grad	Phase of frequency response
$\{\vec{\varphi}_A\},\{\vec{\varphi}_X\}$		Set of two vectors
ω_t	rad/s	Angular velocity

ω_4	rad/s	Angular velocity for 4th joint of the manipulator
ω_g	rad/s	Angular velocity of rotary mass of gyroscope
ω_p	rad/s	Angular velocity for precession of gyroscope
ω, ω_i	1/s, Hz	Eigenfrequency, i=1,2,…,n
ω_0	1/s	Undamped eigenfrequency for freely oscillating system
ω_d	1/s	Damped eigenfrequency
$\dot{\omega}$	rad/s^2	Angular acceleration
ω_p, ω_g	1/s	Tilting and rotary motor speed

1 Introduction

Today, there are great drivers of worldwide technological progress and innovation which, as transformation instruments, long-term revolutionize the world slowly but rather radically. Since their impact on the global society covers several decades, they are considered the key aspects for the economy and social vision for the future and their corresponding plan of action [1].

Some well-known global scale emerging trends are personal health, renewed environmentalism, mobility, globalization, connectivity, individualization, etc. Out of 12 current megatrends, the freedom of choice, ranked among the top 5 as particularly significant for the companies in the near future, because they have mostly characterized their business concern in the 2020s [2].

The individualization as a driver for the transformation refers to value systems, consumption patterns and every day culture [1]. One of their indicators is the mass customization: customers demand offers that fit to their personal preferences and individual needs. In order to remain competitive in the future market, from a business point of view, it is not enough just to identify the opportunities and potentials brought through those drivers, but also their main challenges for the coming years. For the specific case of the mass customization, the main challenge lies in a clear restructuring of the production processes in order to achieve the objectives of the product individualization in a profitable way. This leads to the "batch size 1" in the manufacturing: custom-made and single-item production to the cost of a mass production. Figure 1 shows how many companies already consider mass customization ("batch size 1") as a strategical topic for their transformation.

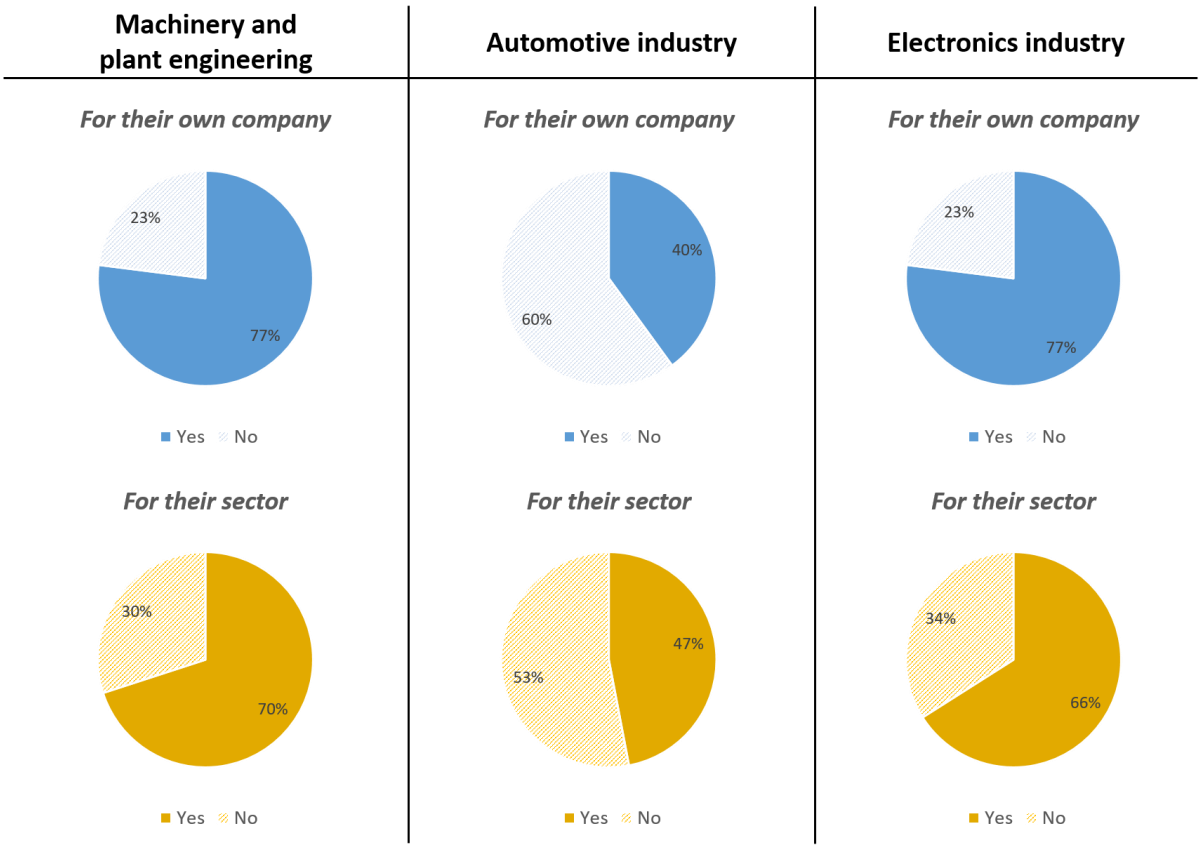


Figure 1 *Is batch size 1 already an important topic for your company and your sector? (based on* [3]*).*

For the purpose of fabricating lucrative and affordable multi-variant up to batch size 1 products, extensive digitalization as well as automation of processes and production chains are crucial. The most relevant automation technologies that enable faster manufacturing while fulfilling the highly-individualized customer needs are the additive manufacturing and the Fourth Industrial Revolution (or Industry 4.0) [4]. Despite the fact that the term "Industry 4.0" derived from the national strategic initiative of the German government in 2011 for the high-tech transformation of the industry, the vision behind has been adopted worldwide in the last years: flexible and full automated industrial manufacturing [5]. Although many authors, such as Mertens in 1995 [6], defined philosophically full automation as "concrete utopia, which is considered as a simple guide without a full-on achievement of the goal", some companies such as Fanuc Ltd. have already proved the opposite: a full automated fabric that seemed totally utopian became a reality thanks to the unattended continuous manufacturing system. The "light out" fully automated plant in Oshino was able to operate in summer 2002 720 hrs. without any interruptions [7]. The production of one thousand robots per month by means of full automated optimized processes saved permanently time, material and human resources [8] and enabled the company to achieve more cost effective production processes than its competitors. This full automation´s success story shows that nothing is impossible, while the necessary resources and strategies are forthcoming.

To ensure not only a fully automated production, but also individualized products, manufacturing processes must be able to adapt themselves to new circumstances, e.g. technical adjustments or near-term modifications regarding the batch size (number of units) determined by customer requirements. Compared with the continuous flow production, which requires big effort and investment for technical retrofits because it focuses on a certain product to achieve cost-efficient high volumes at low type diversity [9], the flexibility provided by the Industry 4.0 enables a quick response to different specifications. Several relevant factors for this flexibility are directly dependent on the properties of the machines deployed in the operations, setup time, automation of technical processes, type diversity, etc.

A flexible production in the "factory of the future" demands for modular machine concepts. Particularly, driverless transportation systems and robots based on AI technologies offer the highest level of flexibility and increase the speed and efficiency for any desired and variable adaptations or sequences due to changing requirements in the production lines, accompanied by the simplification of a production expansion [10]. Therefore, the "factory of the future" not only needs to be flexible but rather interconnected, integrated, adaptable and automated [11] to be able to manage in a smart and optimized way all the processes behind. Thanks to the well-advanced digitalization and 5G [12], the so-called "smart factory" can adopt autonomous robots, driverless transportation systems and drones as well as smart sensors, cameras, IT-systems and machine learning algorithms, which demand a large amount of data exchange and, thus, real time information in order to let machines and devices interact with each other and organize themselves [10]. In the "smart factory", all machines and systems are mobile and modular, the production volume can be increased at any time while running at full capacity in order to achieve a cost-efficient flexibility. Goods are transported to assembly and production equipment by Automated Guided Vehicles (AGVs), Autonomous Mobile Robots (AMRs), mobile manipulators or drones. This is the most cost-effective way, the original planned production line can be transformed for the manufacturing of new/short-term products with the same equipment and the extension or cut back of the overall production.

An example of a such "smart factory" is the intelligent and interconnected factory ARENA2036 conceived by researches of Fraunhofer IPA, university of Stuttgart and representatives from high-tech companies as Daimler and Bosch [13–15]. This project aims at

developing a decoupled, fully flexible and well-integrated production system by adopting loosely linked production modules. Mobile robotics represents an important technology its automation, because it makes the production versatile by providing flexibility and optimization of process chains [16].

As a part of the area mobile vehicles, automated guided vehicle systems, well-known as AGVs, are one of the most implemented technologies a few decades ago, especially to optimize the material flow and to integrate diverse sub-processes in manufacturing operations. Although, it is true that the nowadays widely used AGVs in logistic applications are individual and entirely intelligent, they are unfortunately not autonomous [17]. On the other hand, the autonomous mobile robots (AMRs) are able to navigate in a dynamic and constantly changing environment with a higher level of understanding via sensors, blueprints, artificial intelligence, 3D or 2D vision and more. AMRs are, compared to the traditional AGVs, which employ wires or magnets to guide them along a narrowly predefined route, fully automated. With the proper controller software, all AMRs can be bridged to a central traffic control interface, which manages the navigation of the AMRs based on their current location, thus working together, interacting with or around each other [11].

If the capabilities of an autonomous mobile platform and of a robot manipulator are combined into an integrated system called mobile manipulator, an extremely flexible mobile production module with extra benefits such as starting/stopping, loading/unloading machines as well as the transport of work pieces all over the plant can be created (see Figure 2). This kind of integrated systems helps the manufacturers to reach processes that go beyond the fixed automation, where an industrial robot manipulator fixed on the floor can only reach the work piece as far as its defined workspace limits allow for.

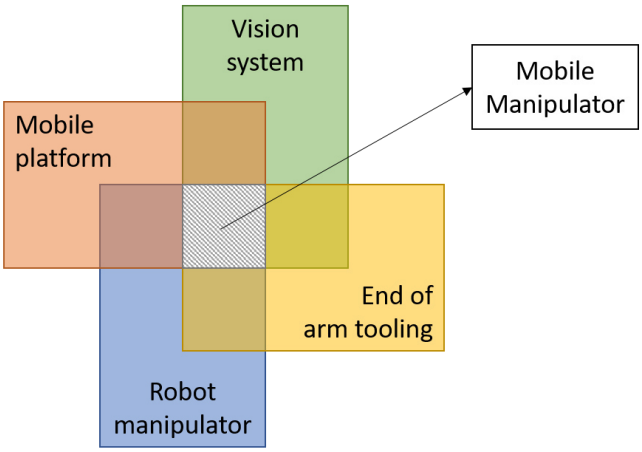


Figure 2 *Integration of mobile manipulators (based on* [18]*).*

Nevertheless, if the robot manipulator consists of a collaborative robot (the so-called co-bot), all the advantages of the system outlined above are complemented with the qualification to work alongside humans and, thus, physical supporting operators in the factory during the performance of manual tasks.

In general, the acceptance of mobile manipulators is high. Manufacturers know that a high level of automation, flexibility, smart manufacturing, smart maintenance and reconfigurable systems result in a time and cost-efficient production system [19]. “The companies need technology able to change along with the work environment”, reported Josh Cloer, Sales Director at Mobile Industrial Robots, Inc., (MiR).

Some notable projects related to the integration of manipulators mounted on mobile platforms that have been implemented during the last few years are shown in the Table 1:

Table 1 *Examples of robot manipulators mounted on mobile platforms.*

Mobile Manipulator	Weight of the mobile platform	Weight of the robot manipulator + Payload
KMR iiwa by KUKA[1] 	390 kg	LBR iiwa 7 R800 =22.3 kg + 7 kg =29.3 kg
		IBR iiwa 14 R820 =29.5 kg + 14 kg =43.5 kg
"Little helper" by Aalborg University[2] 	200 kg	Adept Viper s650 =30 kg + 5 kg =35 kg
		KUKA LBR iiwa 14 R820 =29.5 kg + 14 kg =43.5 kg
RB-KAIROS+ MOBILE ROBOT by Robotnik[3] 	115 kg	Universal Robot UR16 =33 kg + 16 kg =49 kg

[1] Picture source: [201].
[2] Picture source: [18].
[3] Picture source: [202].

Table 1 (cont.) *Examples of robot manipulators mounted on mobile platforms.*

Mobile Manipulator	Weight of the mobile platform	Weight of the robot manipulator + Payload
MM-400 & PRBT by Neobotix&Pliz[4] 	70 kg	PRBT by Pilz + Payload =19 kg + 6 kg =25 kg
MuR205 by the Insitute of Assembly Technology, Leibniz University Hanover[5] 	70 kg	Universal robot UR5e =20 kg + 5 kg =25 kg

The advantages of those systems for a smart factory are already clearly identified:

- Their functional scopes are not attributable to a fixed location, resulting in an unlimited workspace for the robot manipulator and, thus, in a wide range of applications.
- Free from unnecessary use of supporting systems such as loading and unloading stations, conveyor technologies, etc. and therefore, in theory, more affordable than classic industrial robot manipulators.
- Thanks to the human-robot-interaction by employing mobile manipulators, it is now possible to increase the level of automation or to support the operator during complex arduous physical activities.
- The robotic system is significantly more flexible for inaccuracies and deviations on the environment. E.g., if an object is not directly accessible to the robot manipulator, the mobile platform can navigate to a new favorable position and orientation in space.

However, despite their benefits, such systems still present some handicaps, which restrict their unlimited use in production environments:

[4] Picture source: [203].
[5] Picture source: [204].

- Challenging estimation of their flexibility and their limitations in order to quickly introduce new and completely different tasks. Technically speaking, the adaptation of the mobile manipulator and its tools to the working environment, which is constantly in change, is still ambitious for some cases. For example, the suitable lighting for its vision system should change depending on the current position of the mobile manipulator.
- Increasing standards/requirements for safety regarding human interaction and, additionally, the already implemented technically feasible solutions are considerably limited[6].
- Over-proportional higher requirements with regards to the autonomous error handling in comparison to the number of its operational tasks.

Furthermore, the fact that the robot manipulator is mounted on the mobile platform sometimes hinders each other:

- The manipulator requires for a large and stable mobile platform in order to operate safely. Nevertheless, the mobile platform should have a small size and be light weight to navigate in an easy, dynamic and efficient manner.
- The mobile platform must drive fast to reach shorter cycle times. An important aspect to be considered is to ensure not only accuracy for reaching the target position but also the stability during the navigation in the space. Being unstable, the current position of the robot manipulator and the mobile platform is less accurate and the path-planning algorithm requires additional correction mechanisms, resulting in longer cycle times.
- To offer more possibilities for action and, thus, be more efficient, the mobile manipulator comprises a large number of peripheral devices (e.g. vision systems, sonars, scanners, all-in-one grippers, etc.). However, the integration of more embedded components leads to a higher energy consumption and, consequently, to a lower battery duration. At best, the total weight of the system (manipulator and mobile platform together) must be kept to a minimum to increase the battery runtime of the integrated system.

Particularly these last three challenges lead to the design/development paradigm for mobile manipulators: if the mobile platform is large, the whole system is stable but not dynamic and, thus, not supplying optimum cycle times. Furthermore, especially turning maneuvers for large mobile platforms can be complicated in plants with narrow lane width, resulting in worsening cycle time. On the other hand, if the mobile platform has a small footprint, that means, it is compact, the whole system is more dynamic but, of course, unstable. Then, if the mobile platform is unstable, it represents a danger especially for humans, but also for machines and equipment. This paradigm raises the question about how dynamic or rather how compact the mobile platform can be to operate effectively, efficiently and stably.

The ideal solution would be to keep the mobile platform as compact as possible: the mobile manipulator would operate and navigate dynamically; additionally, its radius of motion would not be unnecessarily constrained, allowing short travelling times, and hence gaining a greater energy efficiency for the whole system. The main problem by employing compact mobile

[6] The Robotic Industries Association publishes continuously the R15.08 American National Standard for Industrial Mobile Robots and Robot Systems – Safety Requirements published by (RIA) [205] as common guidelines which comprises requirements for sensor systems, stability, physical and data interfaces, safety-related parts of the control system (SRP/CS), safety behaviors including safety-related stops, and other aspects related to the safety of people around the AMR [11].

platforms for the integration is their instability. This is due to the fact that the whole system's center of gravity lies quite high, as a result of the light weight of the mobile platform in comparison with the robot manipulator's weight mounted on it.

This problematic does not arise in existing mobile manipulators, as those shown in Table 1, because they involve a purely mechanical solution: the mobile platform's weight, compared with the robot manipulator's weight, is much heavier, which drastically shifts the whole system's center of gravity towards the ground.

In particular, during an interaction between a mobile manipulator and a human, machines or other mobile devices, the mobile manipulator must be able to suddenly accelerate and abruptly stop as well as to keep stable while these disturbances are happening, otherwise, the system could cause physical injury or material damage of expensive and unique devices while trying to interact with each other. If a so-called co-bot is mounted on a compact mobile platform, human injury must be avoided. Unintended tilting of the mobile manipulator during Human-Robot-Interaction (HRI) cannot be tolerated. This engineering gap, which more specifically, attempts to face the instability issue by compact mobile manipulators and, thus, to guarantee security and safety operation by a HMI/HRI, can be solved by adopting a mechatronic approach, instead of a purely mechanical one. The mechatronic concept may compensate all instability moments affecting the whole mobile manipulator.

Compact and autonomous mobile platforms, with light weight and low volume, such as those seen on Table 2, are already found in the market:

Table 2 *Compact autonomous mobile platforms.*

Mobile platform model	Weight
OEM LD-60 & LD-90 by Omron[7] 	62 kg
Freight100 Base by Fetch robotics[8] 	68 kg

[7] Picture source: [206].
[8] Picture source: [207].

Table 2 (cont.) *Compact autonomous mobile platforms.*

Mobile platform model	Weight
Scitos G5 by MetraLabs[9] 	60 kg

This implies that the idea proposed above about the development of compact mobile manipulators is indeed possible, if the incurred instability concern could be solved by a mechatronic approach, and not purely mechanical through a heavy weight.

It should be pointed out that for the development of a mechatronic concept to ensure the stability of mobile manipulators during handling tasks, the size and weight of the robot manipulator mounted on the mobile platform are limiting factors: definitely, the larger and heavier the robot manipulator is, the more difficult its integration with a compact mobile platform will be to become a compact mobile manipulator. Nevertheless, it can be argued that the so-called compact industrial robot manipulators and co-bots currently available in the market are the perfect candidates to allow for system integration, because they have a relatively small weight (light and middle weight) related to the payload that their end effector could handle. They seem to offer great potential for the development of compact mobile manipulators. A summary of the weight-to-payload ratio of some of the compact industrial robot manipulators and cobots within the common market is featured in Figure 3:

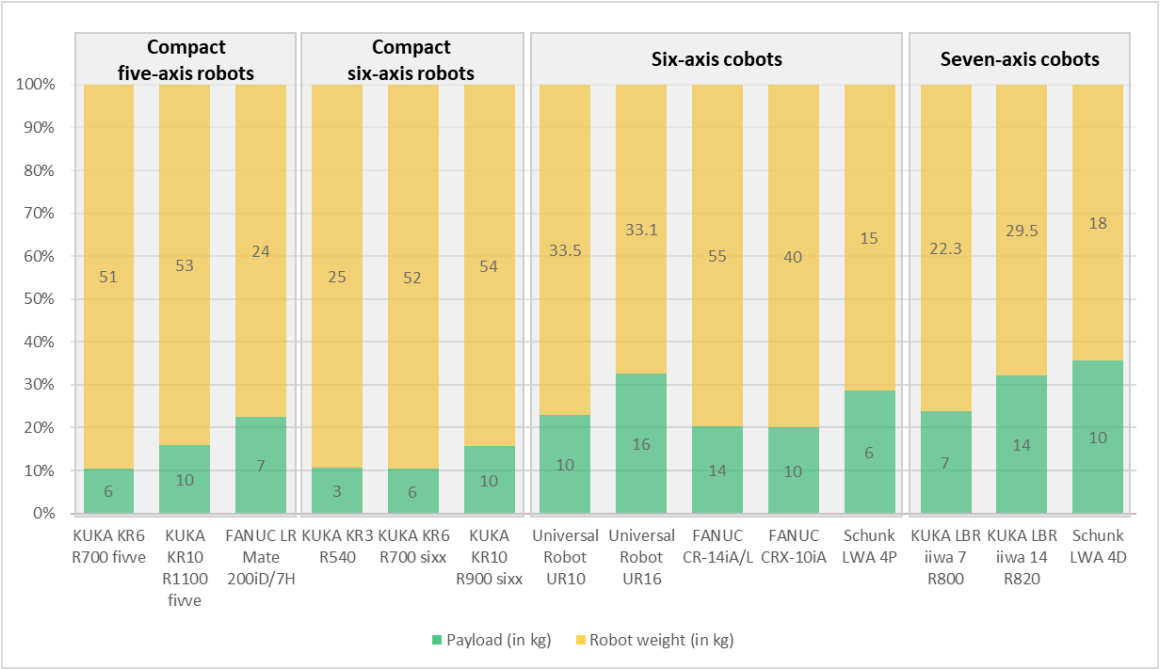


Figure 3 *Weight-to-payload ratio of some compact industrial robot manipulators and co-bots.*

[9] Picture source: [208].

The following configurations on Table 3 illustrate the integration of some already shown compact robot manipulators or co-bots that could be mounted on a compact mobile platform:

Table 3 *Weight ratio of some configurations for compact mobile manipulators.*

Autonomous mobile platform Compact robot manipulator/Co-bot*		OEM LD-60 & LD-90 by Omron (62 kg*)	Freight100 Base by Fetch robotics (68 kg*)	Scitos G5 by MetraLabs (60 kg*)
KUKA KR6 R700 fivve	57.0 kg	1.08	1.19	1.05
KUKA KR10 R1100 fivve	63.0 kg	0.98	1.07	0.95
FANUC LR Mate 200iD/7H	24.0 kg	2.58	2.83	2.50
KUKA KR3 R540	28.0 kg	2.21	2.42	2.14
KUKA KR6 R700 sixx	58.0 kg	1.06	1.17	1.03
KUKA KR10 R900 sixx	64.0 kg	0.96	1.06	0.93
Universal Robot UR10	43.5 kg	1.42	1.56	1.37
Universal Robot UR16	49.1 kg	1.26	1.38	1.22
FANUC CR-14iA/L	69.0 kg	0.89	0.98	0.86
FANUC CRX-10iA	50.0 kg	1.24	1.36	1.20
Schunk LWA 4P	21.0 kg	2.95	3.23	2.85
KUKA LBR iiwa 7 R800	29.3 kg	2.11	2.32	2.04
KUKA LBR iiwa 14 R820	43.5 kg	1.42	1.56	1.37
Schunk LWA 4D	28.0 kg	2.21	2.42	2.14

* Manipulator weight + its maximum payload.

In Table 2, the ratio between the manipulator's weight and the mobile platform's weight is lower than the ratio shown in Table 3.**Table 1**

Mobile manipulators demand reasonable moving speed to ensure effective task performance. Nowadays, "An AMR can approach a person without the person being aware of it," says Carole Franklin, RIA Director of Standards Development. "We need to consider how to reduce the risk of a person being injured by an AMR or its payload [11]".

Problem statement and aim of the presented study.

The constantly changing environment in the plant imposes big challenges on the dynamics of compact mobile manipulators: they tend to tip over when the robot manipulator works under heavy loads, when its position comprises specific critical arm configurations or as a consequence of fast or abrupt movements induced by the mobile platform [20–22].

An important aftereffect of a tip-over, especially for differential wheeled mobile manipulators, is the noncompliance between the calculated TCP (Tool Center Point) and its real current value because it has been altered by the loss of contact between the drive wheels and the ground during the tip-over. Even for mobile manipulators equipped with hydraulically sprung undercarriage, a dramatic compression/expansion of its suspension during an abrupt braking process can be counter-productive. Those circumstances cause inaccuracy for handling or for avoiding collision with other objects that, according to the trajectory computations, should be clear of the path.

Built on the premise that a system starts to tip over when applied forces generate a torque large enough to initiate a rotational motion around a tilting edge, stability disturbance appears when the sum of the tilting moments around one or more of the tilting edges is greater than the sum of its stability moments around the same edge(s) [23]:

$$Stability = \frac{\sum \vec{M}_{stability}}{\sum \vec{M}_{tilting}} \qquad (1)$$

with,
Stability ≥1, the system is stable and balanced.
Stability =1, limit value to define a system as stable or not (the body is not yet tilted over).
Stability <1, system is unstable and tends to tip over.

The simplest approach to avoid tilting risks is to enlarge the ground area of the wheeled system and/or to increase the weight of the system at its basis, shifting its center of mass further to the ground. The drawbacks of those countermeasures are the requirement for bigger aisles, which would directly increase the fixed costs of the plant. Moreover, they fail to address the loss of mobility that is caused by the increase in weight and dimension.

For the specific case of mobile manipulators, both countermeasures are unsuitable due to the fact that the mobile platform has a restricted payload and that the heavier the payload gets, the less agile the system is (important for the estimation of cycle-time) and the shorter the battery supply lasts. Therefore, the aim of this work is to introduce stabilization strategies for small footprint mobile manipulators that compensate instability moments occurring during navigation without restricting its mobility and dexterity. It addresses two approaches for solving the lack of stability problem, depending on the type of mobile manipulator to deal with.

The first approach discusses the development of the stabilization strategies for such mobile manipulators whose control system is closed-source. In this case, only the integration of external actuators allows for the compensation of emerging tilting moments. The stabilization strategies are developed within multibody-system (MBS) simulation environments. The building of close-to-reality simulation models for the employed testing systems implies the experimental determination of the real system dynamic behavior. The system modal parameters serve as reference for the parametrization of the MBS models, which is usually carried out through the iterative adjustment of their stiffness and damping values until the dynamic behavior of the real system matches the simulation model. Due to the fact that this repetitive

procedure is time-consuming, a parametrization algorithm is developed for fitting not only the system natural frequencies, but also their mode shapes by means of statistical indicators. Then, based on this dynamic simulation, the suitable actuators and its control can be developed by means of co-simulations.

The second approach focuses on the development of a stabilization strategy that can be adapted for all kind of compact-lightweight mobile manipulators, regardless of the model, size, manufacturer, etc. The only restriction imposed for its use is that both, mobile platform and robot manipulator, are operated with, e.g., the robotics middleware suite called Robot Operating System (ROS) or a similar open source platform. An algorithm provides information about the system stability state. Based on the computed stability value, the theoretical workspace of the robot manipulator is optimized to a working space in which the mobile manipulator can operate without causing instability. An additional algorithm compensates for the tilting moments emerged during navigation by means of repositioning the arm of the robot manipulator. Both algorithms are implemented and validated on the real testing system.

The following terms will be used throughout this document to refer to the different parts of the systems under study: mobile platform as the mobile robot that navigates autonomously around the environment; robot manipulator as the articulated robotic arm that manages the work piece; and mobile manipulator as the system consisting of the robot manipulator mounted on the mobile platform.

2 Fundamentals

This current chapter contextualizes key theoretical concepts to provide background information on the technical fields touched upon later in this work. The main themes covered in this section are regarding the dynamic and mechatronic modeling of robotic systems, including the theory corresponding to the Experimental Modal Analysis, Multibody-system Simulations and coupled simulations (co-simulations). Additionally, a brief overview of the most well-established robot operating system and a simple definition of the inverse dynamic approach for robot manipulators are given, both employed for the development of the stabilization algorithms.

2.1 Experimental Modal Analysis (EMA)

The implementation of control systems in robotics implies the good understanding of the dynamic behavior of the system of interest, represented by its transfer function. In case the mathematical function of the plant, i.e. its transfer function, is not familiar, a modal analysis helps to identify the modal parameters that characterize the dynamics of the system, such as its modal frequency, modal damping and modal shape. These modal parameters serve as the basis for building the mathematical functions that are employed to model the transfer function of a system in frequency range. Since this transfer function considers the system's dynamical properties, it enables the representation of a close-to-reality model of, e.g., a robotic system to simulate the dynamical characteristics of its structure, or to predict the system's response against external factors.

The method behind the identification of the system's modal parameters by means of modal analysis is based on the principles of the mechanical oscillation theory, whose mathematical statements are summarized employing the following simple example.

Every system that is able to oscillate can be described in terms of masses, dampers and springs. One of the most simplified models to represent an oscillating element is the well-known spring-damper-mass-system, the mass oscillator with a single degree of freedom (DOF) shown in Figure 4.

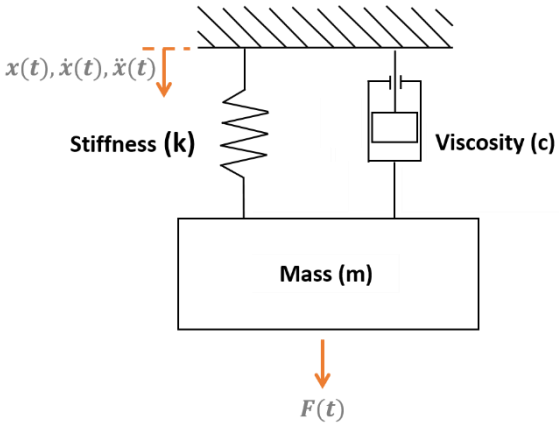


Figure 4 *Spring-damper-mass-system.*

The method of superposition allows to describe complex systems as spring-damper-mass-systems regardless of how many DOF the system has. A system with n-DOF can be described by the superposition of n spring-damper-mass-systems with a certain arrangement in the space.

Thus, a complex system such as mobile manipulator can be modeled by means of multiple masses, dampers and springs.

The equilibrium of forces of the system shown in Figure 4 is characterized by its equation of motion, in time domain resulting in

$$m \cdot \ddot{x}(t) + c \cdot \dot{x}(t) + k \cdot x(t) = F(t). \quad (2)$$

The modal parameters of a system describe the inner dynamic conditions of its structure when it is not being affected by any forces. To obtain the system's oscillation behavior at its characteristic natural frequency, the system has to be considered as freely oscillating, i.e. the system oscillates subject to its own internal forces after the initial excitation was applied. If it is assumed that the applied force $F(t)$ is zero and the system is undamped, meaning that c is also zero, Eq. *(2)* turns into the following second-order homogenous differential equation:

$$m \cdot \ddot{x}(t) + k \cdot x(t) = 0. \quad (3)$$

To solve this ordinary differential equation, the trial function in Eq. *(4)* replaces the terms $x(t)$ with

$$x(t) = A_1 \cdot \cos(\omega \cdot t) + B_1 \cdot \sin(\omega \cdot t). \quad (4)$$

Using Eq. *(4)* into Eq. *(3)* , the following Eq. *(5)* is obtained

$$(-m \cdot \omega^2 + k) \cdot [A_1 \cdot \cos(\omega \cdot t) + B_1 \cdot \sin(\omega \cdot t)] = 0. \quad (5)$$

Since the displacement $[A_1 \cdot \cos(\omega \cdot t) + B_1 \cdot \sin(\omega \cdot t)]$ in a free oscillation does not tend to be zero, in order to solve Eq. *(5)* at any point in time, the part $(-m \cdot \omega^2 + k)$ should be assumed to be zero, thus obtaining the angular frequency of an undamped free oscillating system (ω_0) as

$$\omega_0 = \sqrt{\frac{k}{m}}. \quad (6)$$

Coefficients A_1 and B_1 of Eq. *(4)* can be solved using some specific initial conditions. One of them defines the displacement of the mass at t = 0 as

$$x_0 = x(t = 0). \quad (7)$$

Using this initial condition into Eq. *(4)*, coefficient A_1 takes the value of x_0. Then, applying the first derivate to Eq. *(4)*

$$\dot{x}(t) = -A_1 \cdot \omega \cdot \sin(\omega \cdot t) + B_1 \cdot \omega \cdot \cos(\omega \cdot t) \quad (8)$$

and considering other initial condition, which implies the velocity of the mass at t = 0 as

$$\dot{x}_0 = \dot{x}(t = 0) \quad (9)$$

we can replace it into Eq. *(8)* and solve it for coefficient B_1:

$$B_1 = \frac{\dot{x}_0}{\omega_0}. \quad (10)$$

Consequently, using the values for A_1 and B_1, the motion of the undamped freely oscillating system in function of time can be described as

$$x(t) = x_0 \cdot \cos(\omega_0 \cdot t) + \frac{\dot{x}_0}{\omega_0} \cdot \sin(\omega_0 \cdot t). \quad (11)$$

The amplitude and the phase of the oscillation can be also obtained using A_1 and $B_1$with

$$A = \sqrt{A_1^2 + B_1^2} = \sqrt{x_0^2 + \left(\frac{\dot{x}_0}{\omega_0}\right)^2} \tag{12}$$

$$\varphi = \tan^{-1}\left(\frac{B_1}{A_1}\right) = \tan^{-1}\left(\frac{\dot{x}_0}{x_0 \cdot \omega_0}\right). \tag{13}$$

Thus, Eq. *(11)* can be written as

$$x(t) = A \cdot \cos(\omega_0 \cdot t - \varphi) \tag{14}$$

Eq. *(14)* represents a free and undamped oscillation that does not come to rest after receiving an excitation as input. In the real world, all systems possess a certain damping coefficient. They tend to stop to oscillate after a certain time, depending on its damping coefficient ζ.

From Eq. *(2)* and built on a free ($F(t)$=0) but speed-proportional viscous damping ($c \neq 0$) oscillating system

$$m \cdot \ddot{x}(t) + c \cdot \dot{x}(t) + k \cdot x(t) = 0 \tag{15}$$

the following numerical solution for differential equations[10] can be employed

$$x(t) = A \cdot e^{\lambda \cdot t} \tag{16}$$

where λ represents the eigenvalue. Replacing Eq. *(16)(15)* into Eq. *(15)*

$$m \cdot \lambda^2 + c \cdot \lambda + k = 0. \tag{17}$$

The solution of Eq. *(17)* provides the system eigenvalues as follows

$$\lambda_{1,2} = -\frac{c}{2 \cdot m} \pm \sqrt{\left(\frac{c}{2 \cdot m}\right)^2 - \frac{k}{m}}. \tag{18}$$

If the damping ratio is defined as

$$\varsigma = \frac{c}{2 \cdot \omega_0 \cdot m} \tag{19}$$

and using Eq. *(6)* for the eigenfrequency (ω_0) in a free oscillating system, Eq. *(18)* can be simplified as

$$\lambda_{1,2} = -\zeta \cdot \omega_0 \pm i\omega_0 \cdot \sqrt{1 - \varsigma^2}. \tag{20}$$

The solution of Eq. *(20)* for damped freely oscillating system depends on the value of ζ. Thanks to the particular solutions for differential equations, a system can be classified into the three different oscillation behaviors shown in Table 4:

Table 4 *Classification of oscillation behaviors depending on ζ.*

ζ	**System oscillation behavior**
=0	The system oscillates undamped with ω_0 as eigenfrequency
<1	The system has two complex conjugated eigenvalues
=1	The system has two real eigenvalues (aperiodic borderline case)
>1	The system does not present oscillations

[10] For a detailed description about the oscillation differential equations and its solutions, see [209].

In the real world, there is no system that possesses a damping ratio equal to zero. Most of the existing systems possess a damping ratio $0 < \varsigma < 1$, whose effect is observable e.g. via friction.

On the other hand, if the system has a forced oscillation, i.e. a deterministic harmonic force $F(t) = F_0 \cdot \cos(\Omega \cdot t)$ excites the system and continues to sustain the excited oscillation, the vibrations are said to be forced, resulting in a system displacement $x(t)$. Replacing this into the general Eq. *(2)*, a homogenous and a partial component ($x = x_h + x_p$) are derived. The homogenous component x_h can be estimated based on Eq. *(20)*, using the logarithmic decrement $\delta = \varsigma\omega_0$ and the following damped natural frequency

$$\omega_d = \omega_0 \cdot (1 - \varsigma^2)^{1/2}. \tag{21}$$

The corresponding eigenvalues is determined as

$$\lambda_{1,2} = -\delta \pm i \cdot \omega_d. \tag{22}$$

Employing this value into the initial function Eq. *(4)*, its real components can be obtained as

$$x(t) = e^{-\delta \cdot t} \cdot [A_1 \cdot \cos(\omega_d \cdot t) + B_1 \cdot \sin(\omega_d \cdot t)]. \tag{23}$$

Similarly, by applying the above-mentioned initial conditions, coefficients A_1 and B_1 can be calculated for the special solution of the differential equation as

$$x(t) = e^{-\delta \cdot t} \cdot \left[x_0 \cdot \cos(\omega_d \cdot t) + \frac{\dot{x}_0 + \delta \cdot x_0}{\omega_d} \sin(\omega_d \cdot t)\right]. \tag{24}$$

Further, to determinate the partial component x_p, the following can be applied

$$x_p = X \cdot e^{i \cdot \Omega \cdot t} \tag{25}$$

$$\dot{x}_p = i \cdot \Omega \cdot X \cdot e^{i \cdot \Omega \cdot t} \tag{26}$$

$$\ddot{x}_p = -\Omega^2 \cdot X \cdot e^{i \cdot \Omega \cdot t} \tag{27}$$

where Ω represents the excitation frequency and X its amplitude. Applying them into Eq. *(2)* results in

$$(-\Omega^2 \cdot m + i \cdot \Omega \cdot c + k) \cdot X \cdot e^{i \cdot \Omega \cdot t} = F_0 \cdot \cos(\Omega \cdot t). \tag{28}$$

Then, knowing that

$$F_0 \cdot \cos(\Omega \cdot t) = Re\left[F_0 \cdot e^{i \cdot \Omega \cdot t}\right] \tag{29}$$

Eq. *(28)* can be simplified as

$$(-\Omega^2 \cdot m + i \cdot \Omega \cdot c + k) \cdot X = F_0. \tag{30}$$

As previously stated, a very useful approach to analyze and design control systems is to represent the steady-state of the structure in frequency range by means of its frequency response, also known as transfer function. Considering the representation of the system's dynamics in frequency range, the output spectrum $X(\omega)$ has to be related to the input spectrum $F(\omega)$ via the Frecuency Response Function (FRF) in order to constitute the system's transfer function. From a physical point of view, if a system, e.g., a robot manipulator, is excited with

a sinusoidal wave of frequency ω, it is expected that the system (the robot manipulator) oscillates at the same frequency, as a response to the excitation. The FRF, $H(\omega)$, provides information about the internal dynamic properties of the system as a function of the angular frequency. $H(\omega)$ can be described as:

$$H(\omega) = \frac{X(\omega)}{F(\omega)}. \tag{31}$$

As a result, $|H_\omega|$ represents its amplitude. Thereby, the response amplitude is the input amplitude $|F_\omega|$ multiplied by $|H_\omega|$.

The FRF signal, between the excitation and the response, is shifted by the phase angle φ_ω [24], estimated as follows

$$\varphi_\omega = \tan^{-1}\left(\frac{Im[H(\omega)]}{Re[H(\omega)]}\right). \tag{32}$$

The relation of the output $X(\omega)$ regarding the input $F(\omega)$ is characterized in the so-called modal analysis. A modal analysis can be classified into two different types according to the nature of its execution:

- The Operational Modal Analysis, known as OMA, is based, as its name suggests, on purely mathematical calculations using, among others, Finite Element Methods and MBS (see next section).
- The Experimental Modal Analysis, known as EMA, is an experimental method to obtain the oscillatory behavior of a body or system by measuring its response to a defined excitation.

In this respect, unknown dynamical behaviors of real systems can be easily described by means of its modal parameters (natural frequencies, mode shapes and damping coefficients) through EMA. In this context, EMAs can be used to quantify the dynamic behavior of mobile manipulators, needed for the building and validation of their simulation models used for the development of the stabilization strategies examined within this work.

During an EMA, the magnitude and direction of the force applied to the structure to be examined is measured at one or more points. Using these measurements together with the information about the applied force, the structure transfer function, $H(\omega)$, can be calculated. If more than one system response is measured at the same time, the evaluated frequency responses are arranged into a matrix, following the principle illustrated in Eq. *(33)*.

$$[\boldsymbol{H}(\omega)] = \begin{bmatrix} H_{11} & \cdots & H_{1e} \\ \vdots & \ddots & \vdots \\ H_{m1} & \cdots & H_{me} \end{bmatrix} \tag{33}$$

where *m* corresponds to the measurement point and *e*, to the excitation.

If the system is excited with a frequency close to one of its natural frequencies, the system amplifies its oscillation and, as a result, its response grows into a maximum because the excitation frequency and its natural frequency are superimposed. Using a real-time analyzer and mathematical software tools[11], the transformation of the magnitudes from the time domain into frequency domain can be performed by the Fast Fourier Transformation [25], also well known as FFT. Then, the frequencies with a phase shift of 90° can be identified as a system natural frequency. Finally, the best-fitted modal parameters can be estimated with the help of

[11] For more information about the detailed signal processing, see [31,159].

curve fitting algorithms, which use mathematical models to match fitted parameters with the measured system modal parameters. The system's damping ratios, on the other hand, may contain great uncertainties from the EMA, because they cannot be determined so easily by the curve fitting algorithm. One possibility to determine them is employing the 3-dB bandwidth technique[12].

Besides the natural frequencies, an important system characteristics are the real eigenmodes, which are part of the complex eigenvalues of frequency responses. Those eigenmodes can also be used to assess the quality of simulation models when compared to measurements.

When comparing the eigenvectors from two different data sets, the Modal Assurance Criterion (MAC) checks the orthogonality properties of the eigenvectors by means of the normalized scalar product of the set of two vectors, $\{\varphi_A\}_r$ as test modal vector for mode r, and $\{\varphi_X\}_q$ as compatible analytical modal vector for mode q, as in Eq. *(34)* and Eq. *(35)*, following [26]

$$MAC(r,q) = \frac{\left|\{\varphi_A\}_r^T \cdot \{\varphi_X\}_q^T\right|^2}{\left(\{\varphi_A\}_r^T \cdot \{\varphi_A\}_r\right)\left(\{\varphi_X\}_q^T \cdot \{\varphi_X\}_q\right)} \tag{34}$$

or for complex eigenmodes $\{\psi_A\}_r$ and $\{\psi_X\}_q$, employing their respective complex conjugates $\{\psi_A\}_r^*$ and $\{\psi_X\}_q^*$ as

$$MAC(r,q) = \frac{\left|\{\psi_A\}_r^T \cdot \{\psi_X\}_q^*\right|^2}{\left(\{\psi_A\}_r^T \cdot \{\psi_A\}_r^*\right)\left(\{\psi_X\}_q^T \cdot \{\psi_X\}_q^*\right)}. \tag{35}$$

The correspondence between the two eigenvectors can be represented in a normalized matrix or visualized graphically, as in Figure 5, whereby a correlation 1 means a perfect match between the two modes.

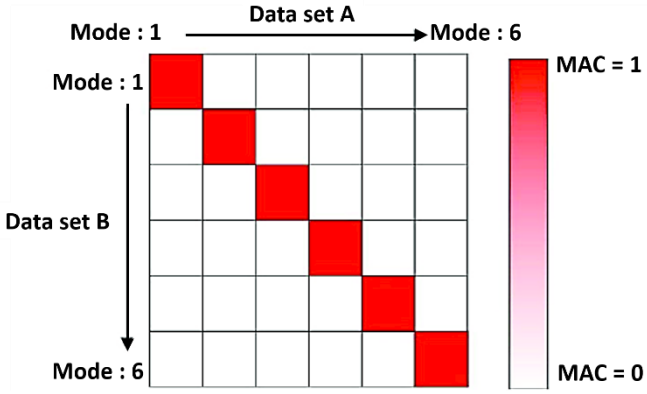


Figure 5 *Example of plot for a MAC with ideal correlations* [26].

The closer the value lays to 0, the lower the match and the bigger the deviations between, e.g., an eigenvector provided by simulations and another one gained during experimental sets. A correlation bigger than 0.9 is considered as good. Values that do not exceed 0.5 have hardly any correlation between the two vectors [27].

[12] For more information about the 3 dB method, see [210].

In fact, the correct selection of the excitation and measurement form is decisive to perform an appropriate EMA. The excitation and measurement points are to be chosen so that the experimental set is able to get all possible modal parameters of the system.

The excitation stimulus can be achieved by employing pressurized loudspeakers test benches, flow loudspeakers test benches, impulse hammers, modal-shakers, shaker test benches, etc.[13] The most common excitation form is the impulse hammer because of its reliability and simplicity, as well as its quick use and economical method [28]. Furthermore, the impulse hammer does not affect the dynamical behavior of the system, since it is not solidly coupled to the structure. At the moment the impulse hammer excites the structure, its internal force sensors (quartz crystal [29]) are able to transform the applied force into an analog signal. This analog signal contains all the information about the amplitude and phase from which the excitation function is derived. Disturbances in the input signal, e.g., noise, can be filtered using an H_2 estimator [30,31], which assumes no noise on the output signal and, consequently, the output measurements are accurate. Figure 6 presents the simple representation of the H_2 estimator transfer function. Only the input signal (X) features noise (M).

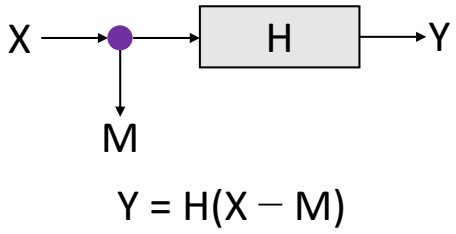


Figure 6 *H_2 estimator function* [32].

There is a risk of inaccuracy by using the impulse hammer, if the impact force applied by the operator is irregular and inconsistent in magnitude, direction and spot location. In order to reduce this risk, it is necessary to apply multiple hammer strokes to one set of measurement.

The impulse hammer tip is responsible for the frequency band for which the system will be excited. Depending on the requirements and/or the system to be investigated, a broad spectrum of the available frequency can be reached by adapting the hardness of the impulse hammer tip [25]. Generally speaking, the higher the degree of hardness of the tip, the higher the frequency range at which the structure excites. A proper tip can ensure that, e.g., a mobile manipulator is excited in low frequencies, the needed/desired frequency band for the further investigations.

The execution of EMA using an impulse hammer is suitable if the system to be examined is expected to present linearity. If nonlinear behavior is expected in the structure under investigation, a linear approximation of the system, or part of it, should be carried out [31]. Therefore, when expecting nonlinear system's behavior, a sinusoidal excitation with large amplitude is recommended, e.g., using electrodynamic shakers.

On the other hand, the corresponding response signal can be measured by sensors such as accelerometers, velocity and position transducers, microphones, laser vibrometers, etc. [29]. The advantage of adopting accelerometers as measurement unit for mobile manipulators is that they do not affect the dynamical behavior of the structure since their mass, in comparison with the system to be investigated, is extremely small and they are not fixed connected to the

[13] For detailed information about the characteristic of each test benches, see [211].

structure. Thus, when selecting an accelerometer, its mass, its stiffness and its damping ratio should be carefully examined in order to discard any influence on the system's behavior: the accelerometer mass is of the utmost importance because it could lead to a shift of those natural frequencies that are located in the lower frequency band. A large ratio between the accelerometer´s and the structure´s mass should be therefore avoided.

Accelerometers are often attached to the structure with a thin layer of wax. Wax as a fastening medium also exhibits high internal damping, which might reduce the usability of the accelerometers at high-frequencies. Moreover, the accelerometers should be mounted on surfaces as rigid as possible, avoiding vibrating parts such as metal sheets or thin plastic parts if the oscillation behavior of the complete structure, and not just a specific system part, is to be recorded.

Ideally, the measurement sets should be carried out in a vibration-isolated foundation and environment, otherwise, the measurement results can also be strongly affected. The possible presence of noise in the input can be excluded using an H_1 estimator [30,31]. In comparison with an H_2 estimator, the H_1 estimator assumes that there is no noise N in the input, i.e. all the noise is assumed to be in the output (see Figure 7).

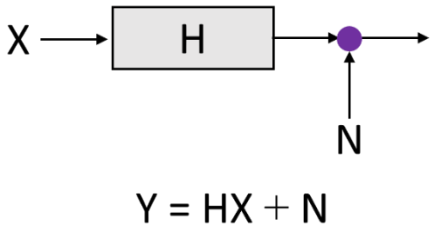


***Figure 7** H_1 estimator function* [32].

An H_1 estimator has the same value as an H_2 estimator only when both their assumptions are fitted: there is no noise at the input when computing H_1, and there is no noise at the output when computing H_2. If these premises do not match, H_1 and H_2 are not equal.

The H_1 and H_2 estimators can also help to estimate the quality of the excitations and the measurements using γ^2, which is defined as

$$\gamma^2 = \frac{H_1}{H_2} \ldots 0 \leq \gamma^2 \leq 1 \qquad (36)$$

whereby a coherence γ^2 equal to zero indicates that the input and/or output signals consist entirely of noise, meaning a lack in quality for the experimental set and, thus, an unacceptable transfer function of the real system. On the other hand, an experimental set with coherence γ^2 equal to one would represent a result without disturbances. The literature (e.g., [24]) suggests a coherence γ^2 equal or higher to 0.9 for a proper analysis.

Given the above considerations with regard to the MAC and coherence coefficients, it can be guaranteed that the modal parameters of a real system obtained by an EMA are accurate enough to derive a system model that provides a very realistic representation of the real system. As a result, this model can be employed in the design and implementation of control strategies: if the control algorithms work properly in the close-to-reality model, they will also behave properly in the real system.

2.2 Simulation Methods, Multibody–system Simulation (MBS)

Due to the high prototyping and tests cost for complex structures, simulation models have a vital role in the development of new systems. The modeling process can be implemented based on the approach developed by [33] for machine tools with parallel kinematics: first, the mechanical components of the system are represented in one model and its electrical drives and controls are then integrated in further modeling steps.

A mechanical system can be described as a set of bodies with mass affected by forces acting on them. The study of the dynamic behavior of mobile manipulators can be carried out with modern widely used multibody-system modeling methods, comprised by the Finite-Element-Method (FEM), Continuous System Simulations (CSS) or Multibody-system Simulations (MBS). They main distinction lies in its target nature: FEM provides a detailed investigation of the system structural behavior, mainly employing elastic body's principles; CSS is the most precise method to describe the mass and elasticity distribution of elastic bodies with simple geometry; and MBS is employed for existing or planned systems with a large number of rigid parts that interact with each other in a complex manner.

FEM is mainly used for the strength calculation of components and vibrational modeling, since it considers all system bodies as deformable. For the system modeling using FEM, a finite amount of single elements with defined physical characteristics is generated for each body, consisting of thousands of DOFs. The joining elements for the singles components are built by nodes with an own coordinate system each. Thus, by defining the DOF of each individual node, the system´s oscillation can be defined. All together, the finite single elements and the nodes, comprise the so-called mesh, which contains the body material properties, such as the Young's modulus and density. Then, a variety of algorithms can be used to compute the interaction between the individual finite elements and, thus, build a close-to-reality system behavior. As a consequence, FEM is also able to determine the system´s frequency response. However, since FEM focuses on the structural behavior rather than the dynamic behavior (in time domain) of a system, its main disadvantage is the very long computational time due to the large amount of DOF the models contain, even by relative simple models. In this respect, FEM is mainly employed for modeling parts in which stress-strain response or the deformation analysis takes place [34]. An example of application is the estimation of the stiffness coefficients (deformation against a predefined force) of a gearbox of a joint robot manipulator.

On the other hand, CSS studies a system behavior by means of differential-algebraic equation models in which only one of all its attributes is studied over time. Due to its implied difficulty, they are well suited for elementary models (beam shaped cantilever) in which stochastic is not expected. In CSS, a fraction of the continuous matter in a compartment is transferred during a small time-step. They mostly contain continuous state variables that change continuously (and not discrete) over time [35].

Finally, MBS is suitable for the development of mechatronic systems [36] for aviation, spacecraft, rail vehicles, machines and robotics. A multibody-system is a set of rigid or elastic bodies connected by kinematic or physical linking elements that help to reproduce the dynamic behavior of more complex mechanical structures, such as robot manipulators. A kinematic connecting element defines geometrical constraints between two bodies by specifying how they can move in relation to each other. They correspond to the classic joint types such as revolute joint, planar joint, cylindrical joint, spherical joint or universal joint [36]. In addition to these purely geometric constraints, physical and basically massless linking elements complete the multibody-system modeling process by assigning spring stiffness's and damping coefficients

to the joining elements, being the responsible for the force transmission between the two bodies. In this way, the motion DOF of rigid massed-bodies can be defined geometrically or via the forces and torques affecting the bodies. In order to obtain a close-to-reality MBS model, the parameters that might have the major influence on the dynamic behavior of the real system should be determined. In the case of a robot manipulator, these are represented by masses, center of gravity vectors and the six elements of the inertia tensors [37] as well as the stiffness and damping parameters of the individual joints.

Table 5 presents a good comparison of these three modeling techniques.

Table 5 *Comparison of modeling methods (Source: Kreuzer et al. (*[38]*).*

Modeling	**Multibody-system Simulations**	**Continuous System Simulations**	**Finite Elements Method**
Body	Rigid	Elastic	Elastic
Geometry	Complex	Easy	Complex
Good level of detail for deformation	Restricted	Available	Restricted
Forces/Torques	Discrete	Steady increments	Discrete
Suitability for control design	Good	After reduction of complexity	After reduction of complexity

Since the aim of the present study is to design stabilization strategies (and not to inspect the system mechanical structure), MBS is suggested as the most suitable method [39].

The accuracy of a MBS depends on the model complexity, the reliability of the system parameters and the quality of the numerical solution [40]. The selection of the degree of detail depends on the nature of the system and the problem as well as on the capacity of the computational analysis.

Since real systems can be represented by rigid or elastic multibody-system models, there are several possible levels of abstraction when modeling a MBS. In principle, modeling a joining element using geometrical constrains (without deformation) means low computing performance and time. In contrast, system joining elements modeled as deformable or non-deformable element each has its pros and cons: the use of deformable joining elements reproduces a close-to-reality behavior but causes considerable difficulties for the numerical solution; however, the use of non-deformable joining elements between rigid bodies only limits the motion of the rigid body(ies) [40]. The forces and torques acting/ reacting on the rigid bodies can be then transmitted through additional massless springs, damping or contact elements.

The theoretical approach for the dynamics of rigid bodies is based on equations of motion[14], e.g. the Newton-Euler approach, the Lagrange approach, etc. [41]. Nevertheless, modeling a system employing rigid bodies (without deformation) implies also a simplification

[14] For a detailed description about the physical principles of the multibody dynamic refer to [36].

of the reality, which could lead to discrepancy regarding the dynamics of the investigated system behavior or, in a worst case, to the model not representing any characteristics of the real system.

If a system is modeled employing elastics bodies, it replicates large body deformations, but implies great computing power. The simplest method to model an elastic structural element is the uniform local distribution of mass and elasticity on point-masses interlinked by massless springs, the so-called Lumped Mass Systems [38], generated in a FE-Mesh with thousands of DOFs. One of the biggest problems and limitations of the MBS modeling with elastic components is the determination of their corresponding spring constants, essential to represent the body properties. This is not a simple procedure and implies a good understanding of the expected system's motion behavior [38]. Additionally, the higher the system DOF, the more complex its mathematical description and, thus, the MBS dynamic calculation.

A reasonable compromise to create a close-to-reality system and to relatively easily model mobile manipulators without needing a high computer performance is the well-known hybrid MBS model. It combines rigid bodies with deformable bodies in the same model: the uncritical components (e.g. robot manipulator links) can be modeled as rigid bodies, while the critical deformable components (e.g. gearboxes) can be described by elastic bodies.

With regard to MBS modeling and simulation tools, the MSC.Software ADAMS/View (Automatic Dynamic Analysis of Mechanical Systems) is the most commonly used to perform kinematic, kinetic and dynamic analyses. MSC.ADAMS/View allows users to model complex mechanical systems, to simulate the dynamics of moving parts and to evaluate the results of the interaction of all defined system components, including motion, structures, actuation and controls (emerging forces, positions, velocities and accelerations) [42]. For motion analysis, all components can be connected via joints and can be loaded with forces and torques. In addition, MSC.ADAMS/View offers the possibility to measure translational as well as rotational movements, velocities, accelerations and forces in all spatial directions, important during the implementation of control algorithms by means of simulations. Furthermore, bearing stiffness and damping coefficients can be defined as physical joints to simulate forces and torques on rotational or translational bearings.

Some of the benefits of modeling mobile manipulators as multibody-systems in MSC.ADAMS/View[15] are:

- The use of robust numerical methods to solve dynamic and nonlinear problems. Clearances and frictions between parts as well as simple or complex contact problems that emerged during the navigation of, e.g., mobile platforms, can be computed by its fundamental numerical methods, including static, kinematic, dynamic, linearization and frequency analysis, being the latter the interesting in this work.
- A variety of libraries are accessible to replicate the same environment and constraints the real system experiences, as well as to set up the simulation requirements. Thereby, the electrical drivers of a robot manipulator together with their rotational velocity or linear displacement and their action/reaction forces can be reproduced using the available control library.
- CAD models can be imported directly into its user interface, helping the preparation of the mechanical system of the robots.

[15] For more information about ADAMS and its diverse libraries, tools as well as examples, please refer to [212–214].

- Its postprocessor provides many visual capabilities for the parametrization of the model, the evaluation of simulation results or their validation against test data through animations and diagrams.
- The plug-in solution ADAMS/Control helps to export the MBS models into black-boxes, in order to couple them with simulation environments for control algorithms, e.g., Matlab/Simulink. This supports the development and verification of control systems in a close-to-reality and cost-effective simulation environment [42].

Generally speaking, MSC.ADAMS/View supports the quick creation and test of mechatronic systems such as mobile manipulators. Thanks to its simulation environments, new developments require less time and cost that would be required to build and test a real prototype [43].

2.3 Co-Simulation (MSC.ADAMS/View & Matlab/Simulink)

The development of the stabilization strategies presented in this work involves the mechatronic representation of the mobile manipulator, comprised by models of the dynamic behavior of its structure, its control and actuation. As previously mentioned, MBS tools offer a suitable environment for modeling the mechanical part. However, the development of the required control loops are subject to many limitations. A long-established mechatronic simulation tool to solve specific problems on the field of control engineering and mechatronics is the software package Matlab/Simulink.

MATrix LABoratory (Matlab) serves as a high-level language and interactive environment for numerical computation, visualization and programming. Matlab supports the acquisition, analysis and visualization of data, the development of algorithms as well as the creation of models and applications [44]. It provides mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations as well as numerical computation methods for analyzing data, developing algorithms and creating models [45]. By combining their packages (libraries) resources, it is possible to build complex programs and applications.

One of the most important add-on programs of Matlab is the toolbox extension Simulink. It provides a graphical development environment for simulation and Model-Based Design (MBD) of linear and nonlinear systems [46], relevant in control design. Its Control System Toolbox is very useful, mainly for the analysis, design and optimization of linear control systems, particularly for the optimization of controllers and sensors [47]. Simulink also supports the rapid prototyping using Hardware-in-the-loop (HIL) and Software-in-the-loop (SIL) techniques [48], being the perfect choice for modeling the stabilization strategy controls.

Splitting a system into subsystems in accordance with the suitable software environment for each discipline facilitates the process of design of the stabilization strategies. The use of coupled simulations, the so-called co-simulations, allow the integration of these loosely and independently software environments, being concerned about the discrete synchronization and interaction of the sub-simulations. Table 6 displays some applications of co-simulations in the robotic field.

Table 6 *Examples of applications that employ co-simulations (based on* [41]*).*

Example of control design	Employing MBS (MSC.ADAMS/View)	↔	Employing MATLAB/Simulink
Control development of an industrial robot manipulator	Dynamics of the robot manipulator	**Co-Simulation**	Feedback for control loop containing joint angles, motor speed and current
	Joint angles, motor speed/acc.		Motor drive torque
Control development of a robot manipulator with elastic arms	Dynamics of the robot manipulator arm and its elastic deformations		Feedback for control loop containing joint angles, motor speed and current
	Joint angles, motor speed/acc.		Motor drive torque
Control development of a DC-Servomotor	Torque calculation, rotor angle/speed/acceleration		Speed and current control loops

Considering the particular case of mobile manipulators, its mechanical structure can be modeled as a MBS model and the necessary control loops can be implemented in Matlab/Simulink environment. Therefore, a co-simulation can be performed in order to make them interact, coordinating two separate software programs (e.g., ADAMS/View and Matlab/Simulink). Due to the fact that one software program requires the information delivered by the other and so on, all relevant information has to be exchanged reciprocally in real-time.

2.4 Robot Operating System (ROS)

In the second part of this work, a stabilization method that directly makes use of the path planning of the mobile manipulators is investigated. An easy accessible robot framework to assist the development and implementation of algorithms to control the robot motion over time is essential.

One of the most well-known frameworks for robot applications is the Robotic Operating System (so-called ROS). Much of ROS´s popularity is due to its modular building design principle to create complex and robust behaviors of robotic systems [49].

ROS is characterized by peer-to-peer networking architecture that enables a secure and high-performance communication between all subscribers (e.g., drivers) of the same network.

All ROS functionalities are distributed in a large number of small tools that solve simple tasks, i.e., for the execution of one complex ROS function such as navigation, several ROS tools are involved (odometers, sensor streams, pose and velocity commands, safety using cameras or scanners, etc.). The developed algorithm is implemented in ROS standalone libraries with no dependencies on hardware, offering the reuse of the developed code for other components or for solving different problem statements [50]. The source code in ROS interacts via the console.

ROS is fully supported by Linux operating system. The algorithms, their corresponding interpreted/compiled code and relevant data necessary for the execution of the programs [51] are contained in packages and located within a workspace that comprises the folder structure of a ROS project [52]. Packages also include the executable files, scripts and launch-files needed for the implementation of the complex functions.

A single ROS executable program performing a task is represented by a node, i.e. the whole project possesses one node for each required task. Usually, a robot control system is comprised by many active nodes, which exchange information with each other in order to accomplish a

function. As an example for a mobile platform, one node performs the localization in space; using the information delivered by this node, another node performs the path planning to reach the target point on the map; then, a third node controls the wheel drive motors for the displacement with the velocity and acceleration the path planning node calculates. Thereby, the entire ROS program consists of a large number of several nodes, which interact/exchange information with each other.

Nodes can publish or subscribe to other nodes via topics [52], the data bus used for the information exchange being the channel for the message flow between two nodes. Thus, nodes can receive messages from other nodes if they subscribe to the respective topic [52]. Similarly, messages from nodes are available to other nodes via published topics.

The ROS master establishes the communication between all nodes that participate in a network. It helps nodes to find each other and to ensure the exchange of information and the supply of services. Nevertheless, once two or more nodes are connected to each other, they can communicate in a direct way without any intervention of the master. In other words, the master only manages the nodes to publish or subscribe on a defined topic.

The node graph assists the visual representation of all nodes and their corresponding messages and topics.

The following additional plugins facilitated the handling of the robot control developed in this work and its simulation environment:

- The robot models can be built within realistic scenarios and environments using the tool Gazebo. In this way, the developed algorithms can be virtually verified in advance before they are implemented on the real robot. For the implementation on Gazebo, the robot geometry is read via the special file in Unified Robotic Description Format (URDF). Furthermore, sensors like accelerometers, scanners or cameras can be also implemented into the Gazebo environment [53]. In order to achieve a close-to-reality model, not only static but also dynamic conditions as gravity force, magnetic field, wind, external disturbances, unknown objects, etc. can be considered.
- RViz as powerful widely used tool for close-to-reality and real-time cognition of robotic systems, in which the perception of the robot such as scans, camera images, point clouds or directions can be read out and displayed graphically.
- MoveIt! is a motion planning framework to program robots in a fast and agile way [49], including a wide range of algorithms for motion and path planning. In addition, MoveIt! helps to analyze and calculate vision perception, forward/inverse kinematics [54], navigation as well as control operations. Specific controllers can be assigned to the individual joints and groups and self-collision matrixes can be defined to avoid collisions during the robot trajectory. MoveIt! also visualizes graphically the robot movement and its effects on the simulation model allowing a risk free implementation of the same planned movement on the real robot.

Altogether, RViz and MoveIt! as control tool and Gazebo as virtual world, assist the development of the stabilization strategies presented in the second part of this work (see Chapter 5) in a close-to-reality virtual environment. The big advantage of their use is the risk-free verification of the developed functionalities in real-time, thus saving time and money. After a successful verification phase in the ROS virtual environment, the control algorithms can be implemented directly on the real robot without requiring major further adaptations.

2.5 Recursive Newton-Euler Algorithm (RNEA)

If the position and orientation of each coordinate system attached to all bodies of the kinematic chain of a robot manipulator are clearly identified, it is possible to determine not only the location of all bodies in the space, but also the location and orientation of the Tool Center Point (TCP) regarding, e.g., the world coordinate system. On the other hand, if the location and orientation of the TCP in a defined space is provided, the location and orientation of all parts of the kinematic chain can be estimated in order to get the configuration they need to reach the given TCP, by means of the so-called inverse kinematics.

Moreover, the Recursive Newton-Euler Algorithm (so-called RNEA) method by Featherstone [55] estimates the inverse dynamics of a rigid-body system required for the evaluation of the stability state of the mobile manipulator.

All 6D spatial vectors and matrixes employed in the RNEA, such as position ($\boldsymbol{S}_i$), joint velocity ($\vec{v}_i$) and acceleration ($\vec{a}_i$), as well as inertia ($\boldsymbol{I}_i$), forces ($\vec{F}$) and reaction forces ($\vec{F}_i^B$) caused by each joint of the robot manipulator, can be described regarding its body fixed coordinate system. The velocity of a certain link i is calculated by the sum of the velocity of its parent link $\lambda(i)$ and the velocity of the joint i that connects both links as follows

$$\vec{v}_i = {}^i\boldsymbol{X}_{\lambda(i)} \cdot \vec{v}_{\lambda(i)} + \boldsymbol{S}_i \cdot \dot{\vec{q}}_i \tag{37}$$

with

$\vec{v}_{\lambda(i)}$ Parent link 6D velocity vector,

$\boldsymbol{S}_i$ 6 x n-DOF matrix for the movement space of joint i,

$\dot{\vec{q}}_i$ n-DOF vector for joint i velocity, with the same dimension as the joint i DOF,

${}^i\boldsymbol{X}_{\lambda(i)}$ Matrixes for the coordinate transformation of the parent link $\lambda(i)$ regarding the child link i coordinate system. For detailed information about coordinate transformation matrixes see [56].

Considering the basis of a mobile manipulator as fixed, its velocity can be denoted as $v_0 = 0$. Based on Eq. *(37)*, the acceleration of the same link i can also be estimated by applying its derivate with respect to time, where the derivate of $\boldsymbol{S}_i$ is performed related to the body coordinate system

$$\vec{a}_i = {}^i\boldsymbol{X}_{\lambda(i)} \cdot \vec{a}_{\lambda(i)} + \boldsymbol{S}_i \cdot \ddot{\vec{q}}_i + \dot{\boldsymbol{S}}_i \cdot \dot{\vec{q}}_i + \vec{v}_i \times \boldsymbol{S}_i \cdot \dot{\vec{q}}_i \tag{38}$$

with

$\vec{a}_{\lambda(i)}$ Parent link 6D acceleration vector,

$\ddot{\vec{q}}_i$ 6D vector for joint i acceleration.

Considering the basis of a mobile manipulator as fixed, its acceleration can be denoted as

$$\vec{a}_0 = -\vec{a}_{gravity}. \tag{39}$$

Similar to Eq. *(37)* and *(38)*, the forces and torques acting on a link $\vec{F}_i^B$ as a result of accelerations, can be estimated by combining the equations for linear and rotational motion of rigid bodies given by Newton and Euler, as follows:

$$\vec{F}_i^B = \boldsymbol{I}_i \cdot \vec{a}_i + \vec{v}_i \times {}^*\boldsymbol{I}_i \cdot \vec{v}_i \tag{40}$$

with

$\boldsymbol{I}_i$ 6 x 6 inertial matrix (mass and moment of inertia of the link i),
$\times^*$ represents a special form for the Cartesian cross product [56].

Based on the last equation, the external forces and torques, $\vec{F}_i$, affecting the joint can be calculated by

$$\vec{F}_i = \vec{F}_i^B - {}^i\boldsymbol{X}_0^* \cdot \vec{F}_i^x + \sum_{j \in \mu(i)} {}^i\boldsymbol{X}_j^* \cdot \vec{F}_j \tag{41}$$

with

$\vec{F}_i^x$ known external forces and torques affecting the joint i, and described regarding the world coordinate system. For this reason, the coordinate transformations matrices ${}^i\boldsymbol{X}_0^*$ are needed. In case, the external forces and torques are defined regarding the O_i, no coordinate transformations are needed and this part of the equation can be omitted.
$\mu(i)$ all children of link i,
$\vec{F}_j$ forces and torques generated by the children $\mu(i)$ affecting the parent link i,
${}^i\boldsymbol{X}_j^*$ matrices for the coordinate transformation of the force vectors.

Similarly, the vector of torques for the joint i can be estimated as $\vec{\tau}_i = \boldsymbol{S}_i^T \vec{F}_i$.

Employing the RNEA for the calculation of the robot inverse dynamics, all forces and torques acting on each joint of the robot manipulator can be computed if the system geometry and the parameters regarding the kinematics and motions are known over time. This enables the estimation of the forces and moments generated by the robot manipulator that affect the mobile platform across their physical connection point(s), and which could induce a tip-over of the entire system.

3 State-of-the-art

The current chapter provides an overview of the most common methods that have been developed by different researchers to assess the stability problem of mobile systems. The first part presents those approaches that compensate for disturbing/destabilizing forces and moments by employing external mechanisms. The subsequent sections introduce a number of studies that comprise other kind of stabilization strategies, namely those that do not employ any external stabilization mechanism, but rather the manipulator itself, to guarantee the system stability.

3.1 Stabilization approaches using external mechanisms

Several institutes [57] and companies [58] are currently working on lightweight mobile platforms. Although this form of robotic solution avoids the potential risk of losing their stability while the mobile platform moves through the plant, they are limited by their restricted payload in contrast with previous large systems introduced in Chapter 1. The demand for handling higher payloads implies the use of big robot manipulators (with higher center of gravity) mounted on large-footprint and/or heavier mobile platforms to prevent the robot manipulators from tipping over [59], as the examples shown in Figure 8.

Figure 8 *Two representative models of mobile manipulators (left: KUKA KMR iiwa* [60]*, right: OMRON TM-manipulators with LD-mobile platforms* [61]*).*

Likewise, large-footprint platforms are directly associated with more workspace required, and thus higher costs for the plant surface area. There is, therefore, a definite need to develop a mobile manipulator comprised by a high payload robot manipulator on a small-footprint autonomous mobile platform.

Numerous studies have attempted to develop mechanisms and algorithms to guarantee the stability of mobile platforms and mobile manipulators.

The first systematic studies about mobile platforms equipped with an external stabilization mechanism was reported by Graf and Dillmann [62–64] in 1997 and 1999. They attempted to compensate accelerations and decelerations of mobile transport systems using the concept of a six degree-of-freedom (DOF) Stewart-platform mounted on a wheeled vehicle. An example of application of the principle introduced by Graf and Dillmann [62–64] is shown in Figure 9.

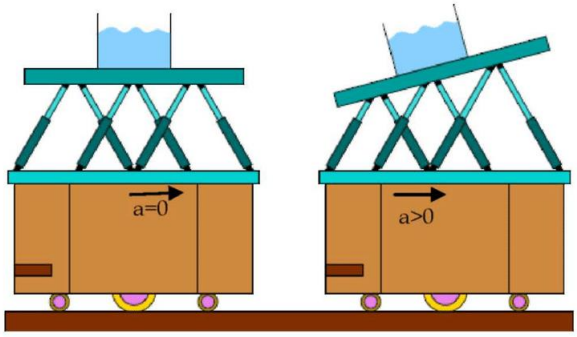


Figure 9 *Compensation of linear accelerations by means of a Stewart-platform (*[65] *based on* [63]*).*

This motion is implemented by the superposition of accelerations generated by a Washout-Filter[16] and the so-called g-tilt effect [64]. The Washout filter calculates valid positions and orientations for the upper plate of the Stewart-platform by means of the double integration of the angular and linear acceleration of the mobile platform. Moreover, the g-tilt effect utilizes the gravity force to generate continuing accelerations in horizontal directions. The washout filter acts as limiter for the g-tilt and executes a tilt so slowly that this movement produces only an insignificant rotatory acceleration of the upper plate of the Stewart-platform. Both accelerations are outlined by the curves in Figure 10, with acceleration by movement for the Washout-filter effect and acceleration by angle for the g-tilt effect.

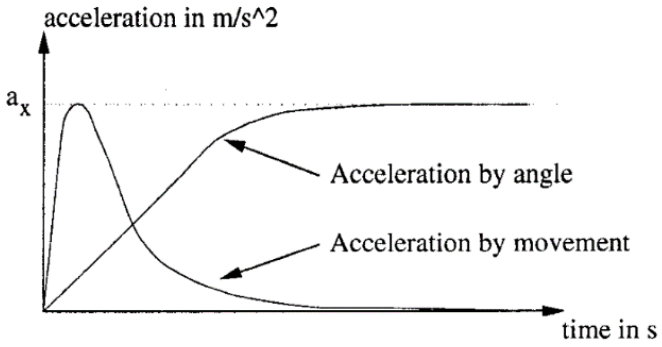


Figure 10 *Compound motion generation* [63].

Using inverse kinematics, the position controller determines the stroke length of the 6 linear actuators of the Stewart-platform that have to be controlled to compensate disturbing accelerations on the mobile system. The complete control algorithm is represented graphically in Figure 11.

[16] Detailed information about the filter is described in [63] and [215].

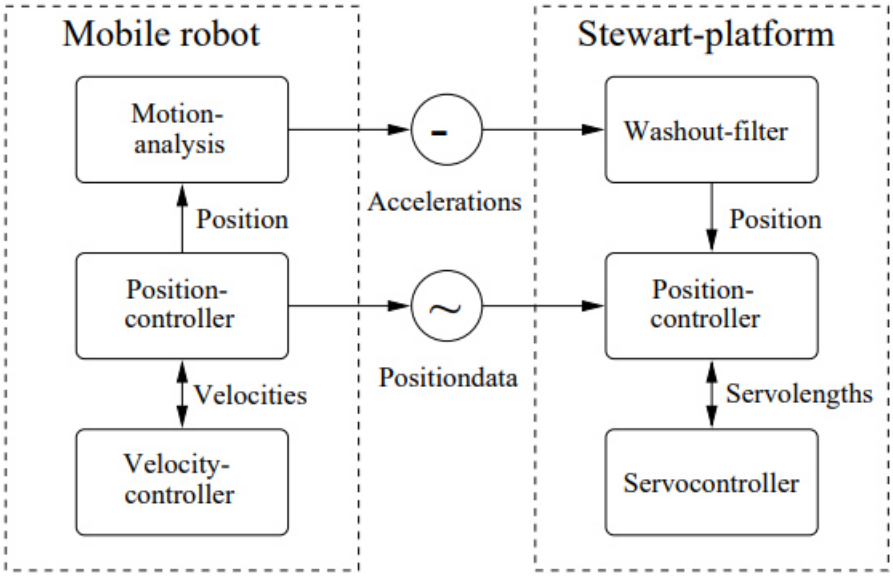


Figure 11 *Stewart platform controlling architecture* [66].

Besides the complex design, this concept was originally developed especially for the transportation of liquids in small containers, i.e. for smaller weights than for standard industrial robot manipulators. The main disadvantage of the concept presented by Graf and Dillmann [62–64] is that large sized components for the 6-DOF-platform are required to achieve only small compensations, which would seriously limit the use of small footprint wheeled systems. Adopting this approach for industrial robot manipulators mounted on mobile platforms would imply the use of very larger sized lineal positioning elements which displace the robot manipulator center of mass higher (decreasing stability) and demands a much bigger footprint area of the wheeled system suitable to mount the Stewart-platform. Further researches [67–69] include the application of this approach.

Another similar approach is the well-known two-wheel inverse pendulum principle mostly employed for the balanced locomotion of two-wheel robots, as shown in Figure 12.

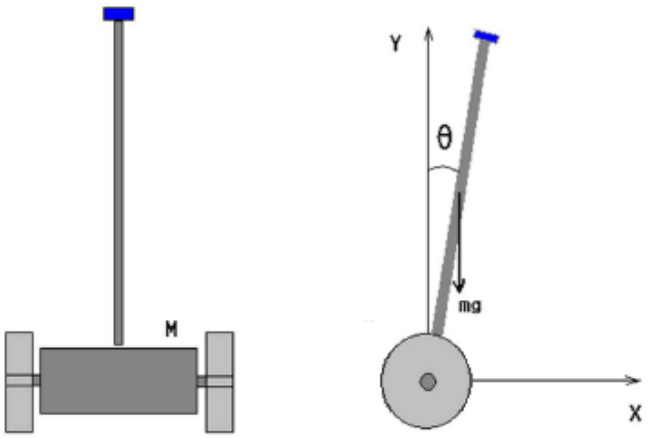


Figure 12 *Ideal two wheeled inverted pendulum system* [70].

With a reasonable design and robust control techniques, many concepts have been already successfully implemented for person transporter purposes [71–76]. Figure 13 gives an overview of the controllers that have been investigated for two-wheeled robots.

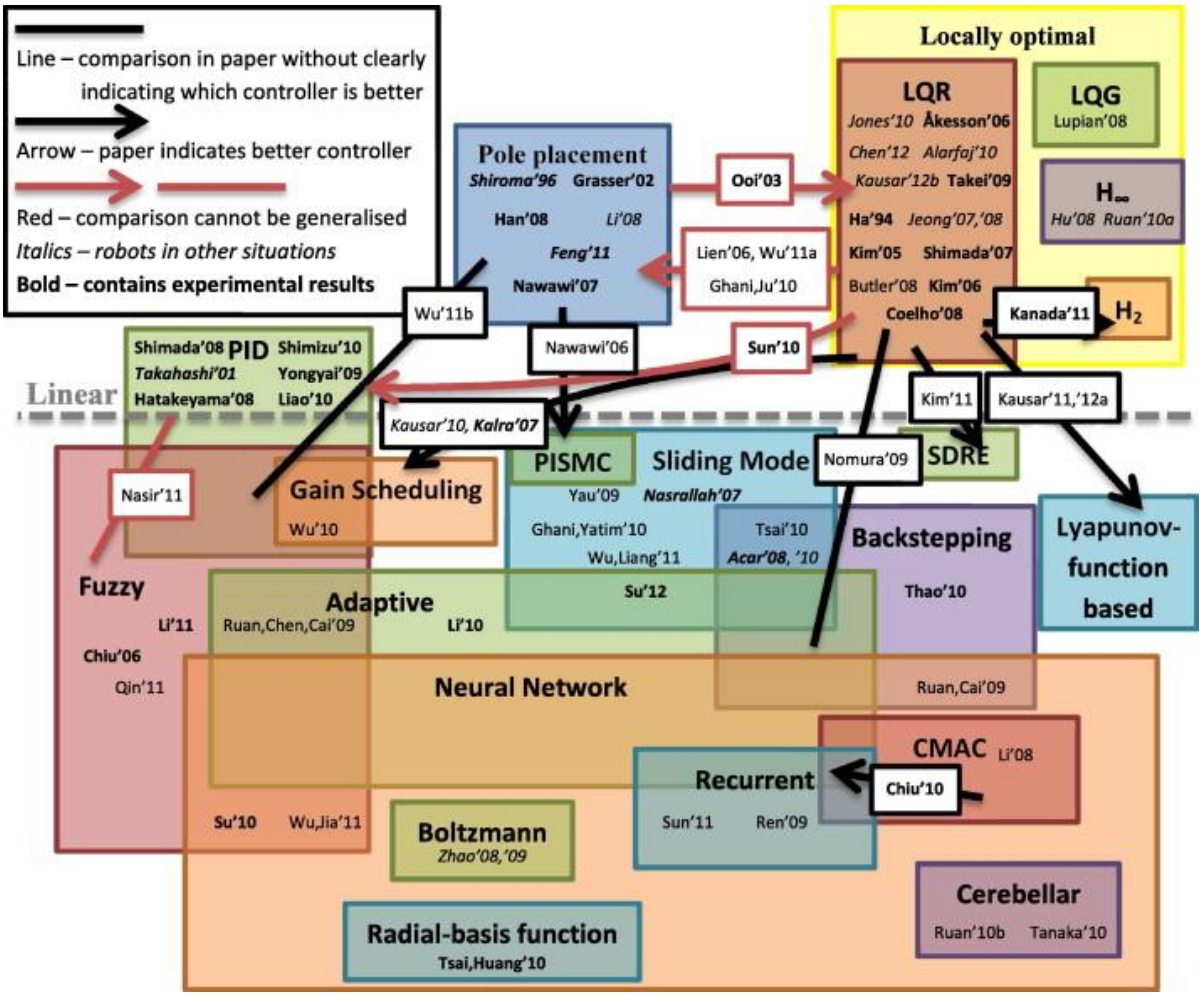


Figure 13 *Outline of most-used controllers for two-wheeled robots* [71].

An extension to the standard stabilization functionality of the two-wheeled inverse pendulum is the mechanism adopted to self-tilt-up without any driver's intervention, as described in [77]. To achieve a self-tilting balancing, a precession motion is achieved via a flywheel mounted inside the vehicle. The flywheel and the body of the vehicle are forced to move around a fixed point. This motion generates a synchronous moment that tilts the body into the upright position. A prototype of this mechanism is shown in Figure 14.

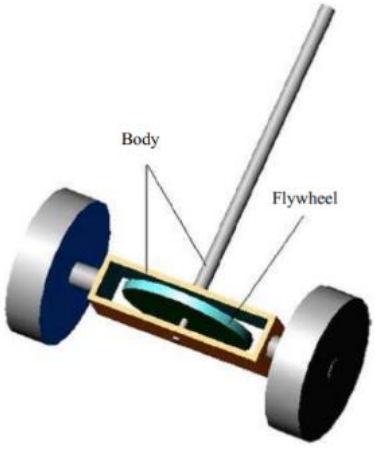


Figure 14 *Concept of flywheel as stabilization mechanism for two-wheeled inverse pendulum* [77].

The dynamics of those systems have been significantly enhanced by further studies [78–80]: In addition to the typical pitching, yawing and straight motions, the two-wheeled vehicle is integrated with an auxiliary mechanical tilting mechanism composed by a ball screw spindle that allows rolling and vertical motions (see Figure 15).

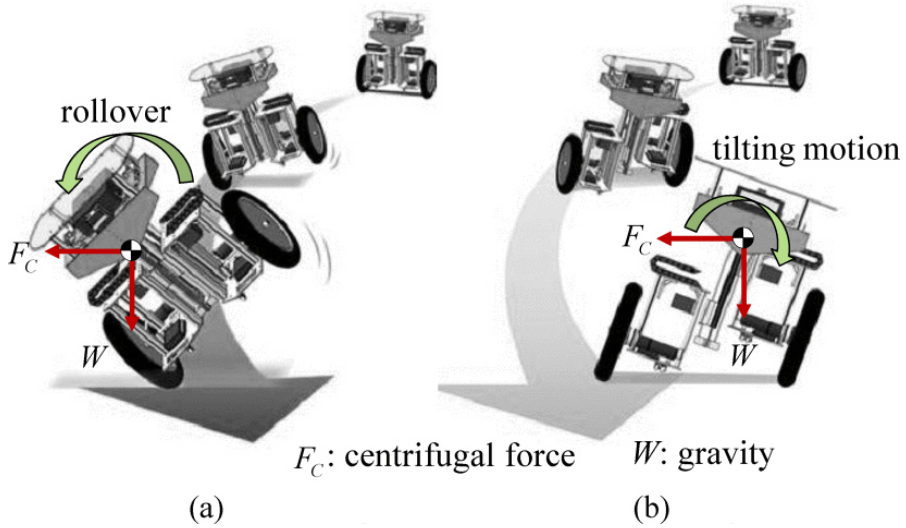


Figure 15 *Mechanism to achieve lateral stability for two-wheeled vehicles* [80].

The tilt control of the auxiliary balancing mechanism that compensates the centrifugal acceleration of the two-wheeled vehicle is carried out by the control concept illustrated in Figure 16. In order to determine the angle of inclination required to compensate the centrifugal forces occurring in the transverse direction, the force equilibrium and moment equilibrium conditions must be established as a function of the angle of inclination.

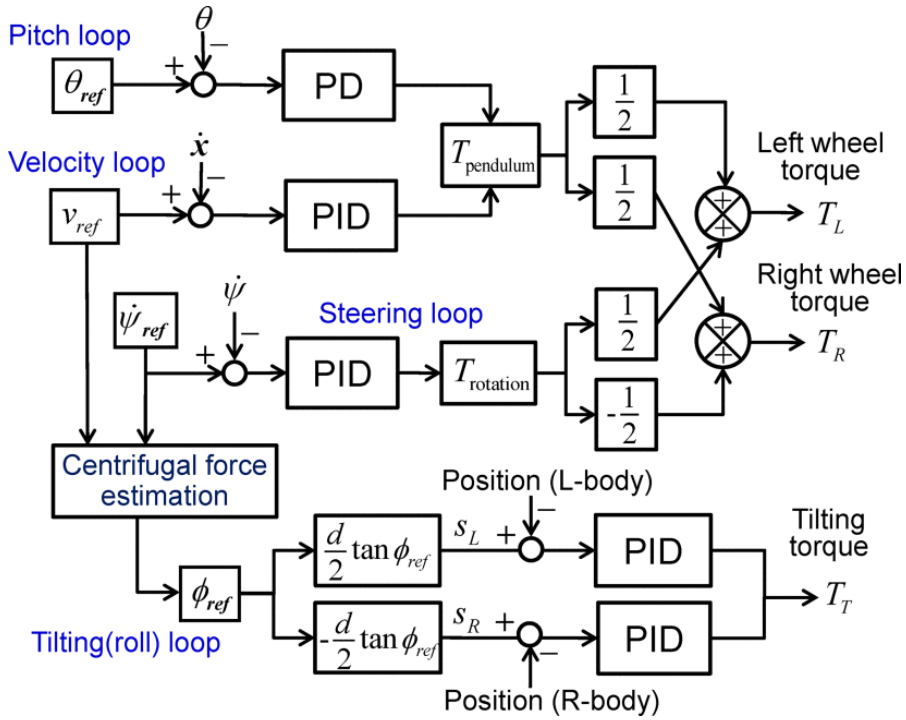


Figure 16 *Control concept for the auxiliary balancing mechanism of two-wheeled vehicles* [80].

Zhao et al. [81] as well as Acar and Murakami [82], have attempted to implement the system comprised by the three DOF manipulator mounted on the two-wheeled mobile platform

displayed in Figure 17. They applied the inverted pendulum principle to model the robot system's dynamics. In the model, the robot center of gravity (COG), an important parameter to keep the system stable, varies depending on the displacements, velocities and acceleration of the mobile platform, as well as on the weight and height of the robot manipulator.

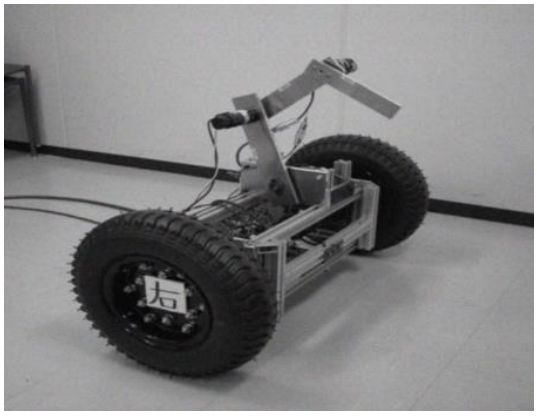


Figure 17 *Prototype of 3-DOF manipulator mounted on a two-wheeled vehicle* [82].

The design approaches developed at the Institute for Cognitive Systems at the Technical University of Munich [83] as well as at the Korea Institute of Science and Technology [84] and at the Fraunhofer Institute of Optronics in Karlsruhe [85] consist of two-arm humanoid upper bodies mounted on Segway devices to allow them be moved. In Figure 18, the robot manipulator's body of the MAHRU-M [84] is mounted on a "Compact Omni-directional Mobile Platform".

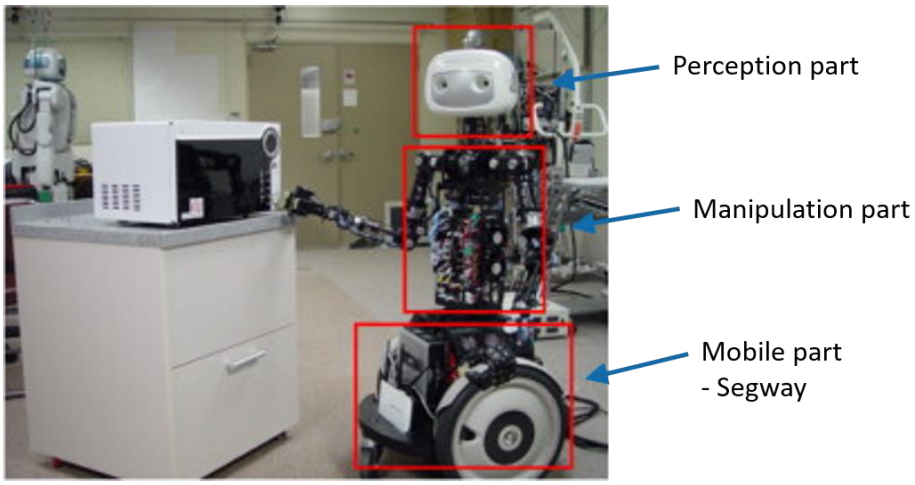


Figure 18 *Mobile humanoid robot MAHRU-M* [84].

Both systems, the three DOF mobile manipulator and the two-arm humanoid mounted on two-wheeled vehicles, are not suitable for application purposes in industrial environments due to their lack of DOFs and their high cost and complexity, respectively.

Another potential approach for external stabilization, apart from tilting, revealed the use of gyroscopes, similar to the technic presented for the two-wheeled inverse pendulum with integrated flywheel abovementioned. Early examples of research into gyroscope mechanisms and its successful implementation include the today´s predominant use to reduce roll on ships, to control the airplanes automatic pilot, as well as missiles, satellites (Figure 19 [86]) and high-degree stabilization capabilities of camera systems.

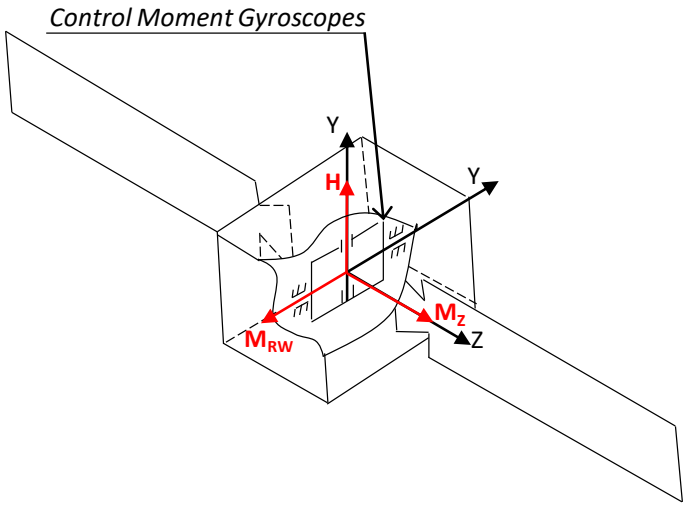


Figure 19 *Schematic representation of satellite orientation by a control moment gyroscope* [86].

Research into the gyroscopic effect and its benefits has a long history. Arnold and Mauder [87] demonstrated that the use of a gyroscope as stabilizer is particularly promising because of the immense torques it can exert under precession, and its rapid response capacity. In 1959 and 1960 respectively, Novoselov [88] and Matrosov [89] described in detail the forces and torques acting during the gyroscope effect. Most recent researches, as in [90], attempt to evaluate the impact of perturbations and how the gyroscope can manage them by implementing new torque control algorithms based on continuous high-order sliding mode [86].

3.2 Detection of instability states

In order to avoid a tip-over of the mobile manipulator, it is necessary to first detect all instability states. This far, several techniques have been developed to analyze the dynamical tilting stability of mobile manipulators. The study proposed by Ghasempoor and Sepehri [91] calculates the system energy level with respect to the tilting axes. On the other hand, Li [92] suggested the evaluation of the support forces between the wheels and the ground. A significant analysis and discussion on the subject was presented by Papadopoulos and Rey [20,93,94], the so-called Force Angle stability measure (FA), which inspects the vector between the system resulting total force (measured by an Inertial Measurement Unit) and the normal vector of each tilting axis. The reaction force vector $\vec{F}_r$ and the normal vectors of the tilting axes T_1 and T_2 build the angles θ_1 and θ_2 shown in Figure 20 (for manageability, the schema illustrates only the components i=1 in a 2-DOF system). The length of the vectors $\|\vec{d}_1\|$ and $\|\vec{d}_2\|$ describes the distances between the resulting force line of action and the contact point between the wheels and the ground.

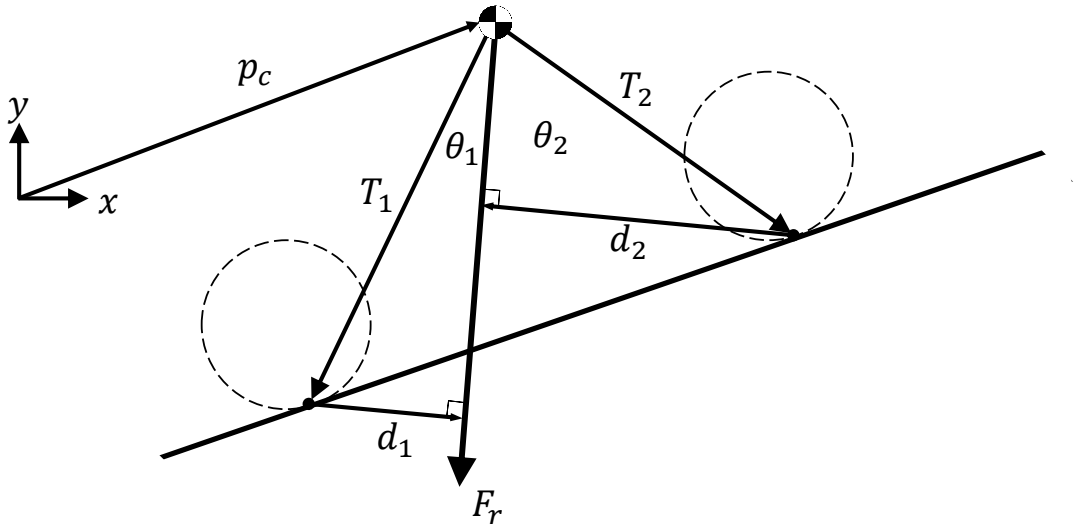


Figure 20 *Planar Force-Angle stability Measure* [95].

The Force-Angle stability measure is defined as

$$\alpha = \theta_i \cdot \left\|\vec{d}_i\right\| \cdot \left\|\vec{F}_r\right\|. \tag{42}$$

If α contains a negative value, if the total reaction force points outside θ_i or if the COG points outside the tilting shape (comprised by the wheels), the system starts to tilt over.

Defining the connecting line between the system tilting axes $\vec{e}_i$ as

$$\vec{e}_i = \vec{p}_{i+1} - \vec{p}_i \quad i=\{1,2,\dots,n-1\} \tag{43}$$

where n is the amount of contact points and $\vec{p}_i$ is the position vector between these contact points with the ground, using as reference frame the coordinate system located at the system COG. By employing the normalized vector $\hat{e} = \frac{\vec{e}}{\|\vec{e}\|}$, the normal vector of the tilting axes T_i that passes to the COG is characterized by

$$\vec{T}_i = (1 - \hat{e}_i \cdot \hat{e}_i^T)(\vec{p}_{i+1} - \vec{p}_c) \tag{44}$$

with p_c as the instantaneous position vector of the system COG and 1 as the $3x3$ Identity matrix. Then, the total reaction force of the system acting on the COG is calculated as follows

$$\vec{F}_r = \sum \vec{F}_g + \sum \vec{F}_{ee} + \sum \vec{F}_s + \sum \vec{F}_d - \sum \vec{F}_I \tag{45}$$

with $\vec{F}_g$ for gravitational force, $\vec{F}_{ee}$ for the forces transmitted by the end effector to the system (e.g. payload), $\vec{F}_s$ for the reaction forces of the vehicle support system, $\vec{F}_d$ for external forces (eg. disturbances) and $\vec{F}_I$ for the inertia forces. In addition to the forces, the resulting moment generated around the COG is determined as

$$\vec{M}_r = \sum \vec{M}_{ee} + \sum \vec{M}_d - \sum \vec{M}_I \tag{46}$$

which comprises both external and internal moments caused by $\vec{F}_{ee}$ and $\vec{F}_d$ as well as $\vec{F}_I$ and their corresponding cantilever. For a given tilting axis $\hat{e}_i$, only the components of $\vec{F}_r$ and $\vec{M}_r$ that act about $\hat{e}_i$ are substantial for the calculation:

$$\vec{F}_i = (1 - \hat{e}_i \cdot \hat{e}_i^T) \cdot \vec{F}_r \tag{47}$$

$$\vec{M}_i = (\hat{e}_i \cdot \hat{e}_i^T) \cdot \vec{M}_r. \tag{48}$$

In order to estimate the angle θ_i for the FA stability measure according to the vector $\vec{F}_i$ related to each tip-over axis normal $\hat{T}_i$, the moment $\vec{M}_i$ must be replaced by the equivalent force couple of $\vec{F}_i$. Thus, the force $\vec{F}_i$ intersects the tilting axis i, forming a cantilever to the normal vector of the tilting axis as follows

$$\vec{F}_i = \frac{\hat{T}_i \times \vec{M}_i}{\|\vec{T}_i\|} \tag{49}$$

whereby, the normalized vector for the normal vector of the tilting axis is

$$\hat{T} = \frac{\vec{T}}{\|\vec{T}\|}. \tag{50}$$

Thus, the resulting force $\vec{F}_{FA}$ regarding the i tilting axis of the system is

$$\vec{F}_{FA} = \vec{F}_i + \frac{\hat{T}_i \times \vec{M}_i}{\|\vec{T}_i\|} \tag{51}$$

In order to determine the tilting stability coefficient, the distances between the force and the ground contact point must be calculated

$$\vec{d}_i = -\vec{T}_i + (T_i^e \cdot \hat{\vec{F}}_{FA}) \cdot \hat{\vec{F}}_{FA} \tag{52}$$

as well as the angle between the force and the normal vector to the tilting axis as

$$\theta_i = \sigma_i \cdot \cos^{-1}(\hat{\vec{F}}_{FA} \cdot \hat{T}_i) \tag{53}$$

whereby, the normalized vector for the resulting force $\vec{F}_{FA}$ is

$$\hat{\vec{F}}_{FA} = \frac{\vec{F}_{FA}}{\|\vec{F}_{FA}\|} \tag{54}$$

and σ_i represents the proper sign for θ_i, which adopts a positive value if the projection of the cross-product of the force vector $\hat{\vec{F}}_{FA}$ and the normal vector of the tip-over axis $\hat{T}_i$ is positive, i.e. in this case the $\hat{\vec{F}}_{FA}$ points inside the defined tilting edge[17]:

$$\sigma_i = \begin{cases} +1 & \left(\hat{\vec{F}}_{FA} \times \hat{T}_i\right) > 0 \\ -1 & otherwise. \end{cases} \tag{55}$$

Finally, the tilting stability is obtained with

$$\alpha = \min(\theta_i \cdot \|\vec{d}_i\| \cdot \|\vec{F}_{FA}\|) \tag{56}$$

Therefore,

$$\alpha = \begin{cases} > 0 & system\ is\ stable \\ < 0 & system\ is\ unstable. \end{cases} \tag{57}$$

The result of FA is a binary value, distinguishing between stable and unstable, but lacking a statement about the degree of risk.

[17] Only applicable if the direction of the tip-over axis is the same as the stabilizing moment vector.

Another relevant approach is the estimation of the stability by the so-called Zero Moment Point (ZMP), which was first established by Vukobratovic et al. [96] and implemented by Sugano, Huang and Kato [97,98]. ZMP assumes a system is stable if the sum of all moments acting on the system is equal to zero. It calculates a stability value by means of one point (marked as ZMP in Figure 21), which is located on the mobile platform supporting surface and represents the location where the sum of all resultant moment should be zero.

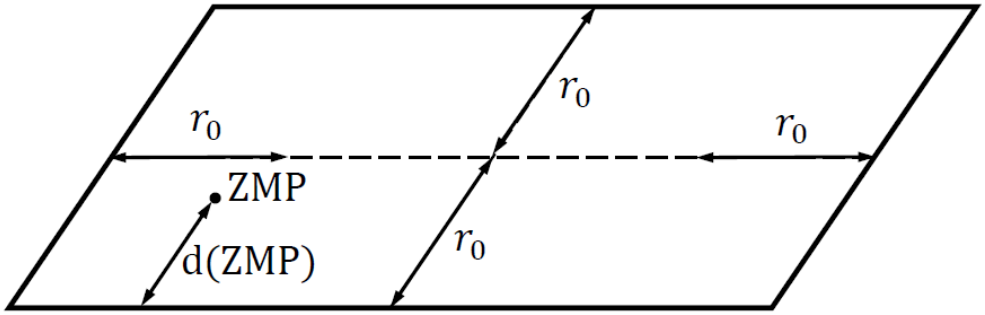


Figure 21 *Stable regions determined by the ZMP* [95].

All moments are derived from gravitational forces and linear accelerations as well as from external forces and moments that affect the system. Internal forces generated by the joints and their movements, and external forces produced by the acceleration of the mobile platform [99] are also considered. A predefined stable region helps to estimate if the system, depending on its ZMP, can be considered stable or not.

In the specific case of the mobile manipulator, this region corresponds to the inner surface of the wheels polygon. For the calculation, the bodies that comprise the system are considered as point masses with an own coordinate system. The acceleration of any point located in the multibody-system may be calculated and modified by applying d'Alembert's strategy, in order to obtain the x- and y-coordinate of the ZMP as follows:

$$x_{ZMP} = \frac{\sum m_i \cdot (\ddot{z}_i + g_z) \cdot x_i - \sum m_i \cdot (\ddot{x}_i + g_x) \cdot z_i + \sum M_{exy} + \sum (S_{ez} \cdot F_{exx} - S_{exx} \cdot F_{exz})}{\sum m_i \cdot (\ddot{z}_i + g_z) - \sum F_{exz}} \quad (58)$$

$$y_{ZMP} = \frac{\sum m_i \cdot (\ddot{z}_i + g_z) \cdot y_i - \sum m_i \cdot (\ddot{y}_i + g_y) \cdot z_i + \sum M_{exx} + \sum (S_{exy} \cdot F_{exz} - S_{exz} \cdot F_{exy})}{\sum m_i \cdot (\ddot{z}_i + g_z) - \sum F_{exz}} \quad (59)$$

with m_i for the mass of body i, $\vec{r}_i$ for the position vector of body COG i ($[x_i, y_i, z_i]$), $\vec{M}_{ex}$ ($[M_{exx}, M_{exx}, M_{exz}]$) and $\vec{F}_{ex}$ ($[F_{exx}, F_{exy}, F_{exz}]$) for external moments and forces affecting the system, and $\vec{S}_{ex}$ ($[S_{exx}, S_{exy}, S_{exz}]$) for the point of application of these external forces. The coordinate z is located on the surface of support; for this reason, its value does not need to be calculated.

A stable region has to be defined to estimate if a system can be considered stable or not. Similar to the FA method, if the positions x_{ZMP} and y_{ZMP} calculated by ZMP are situated inside the defined region, the mobile manipulator can be considered as stable; otherwise, if the resultant ZMP lies outside this region, the mobile manipulator is unstable. In general, the farthest away from an edge of the region to the ZMP resultant, the more stable against tip over the mobile manipulator is regarding this edge. According to this, it is possible to define

$$\alpha_i = \frac{d_i}{r_{max}} \tag{60}$$

with α_i being the stability criterion regarding the edge i of the stability region, d_i as the smallest distance and r_{max} for the longest distance between the ZMP resultant and the edge i of the stability region, as shown in Figure 21.

The ZMP strategy was originally developed for humanoid robots, whose COG does not change significantly. Moreover, this method considers only mass points, ignoring the bodies inertia moment. Both situations could lead to handicaps in its use for mobile manipulators.

The main disadvantage of the above methods is that they did not consider parameters produced by the system dynamics. A method that takes into account this aspect is the Moment Height Stability method (so-called MHS) introduced by Moosavian and Alipour [99–101] for the computation of the moments acting on each tilting axis of the system. The MHS is also a moment-based method which considers, in addition to the internal and external forces (as ZMP does), the body moments of inertia in the stability calculation. In a highly simplified way, the MHS method calculates the total torque acting on each of the individual tip-over edges of the system and, additionally, emphasizes them with inertia values and with the overall COG height. For the estimation of a stability value by means of MHS, the system should first be split into the two parts that are physically connected, as the example illustrated in Figure 22.

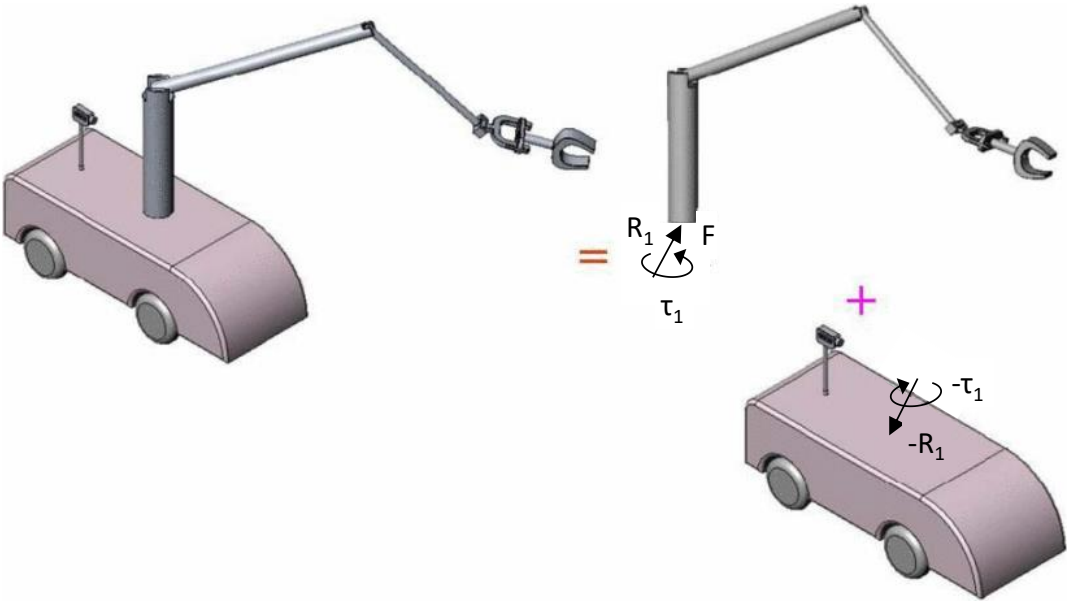


Figure 22 *Separation of whole system into two subsystems, since the MHS measure is computed on the part which produces mobility (the mobile platform)* [101].

This method serves to kinematically and dynamically analyse the two main bodies separately:

- For the manipulator, joint positions, joint angular velocities and joint angular accelerations over time need to be determined. All forces and moments acting on the mobile platform across the connexion point can be estimated by means of inverse dynamic method (RNEA).
- For the mobile platform, similar to the already described Force-Angle stability measure method, a unit vector $\widehat{e_i}$ is defined for each tilting edge. Given $\vec{p}_i$, $\vec{p}_{i+1}$, …, $\vec{p}_{n-1}$ as the vector for the contact points of the mobile platform with the ground (wheels), with respect to the coordinate frame formed by the point where the robot manipulator is attached to the mobile platform, the unit vectors can be calculated as follows

$$\hat{e}_i = \frac{\vec{p}_{i+1} - \vec{p}_i}{\|\vec{p}_{i+1} - \vec{p}_i\|} \text{ , respectively } \hat{e}_n = \frac{\vec{p}_1 - \vec{p}_n}{\|\vec{p}_1 - \vec{p}_n\|}. \quad (61)$$

In addition, the resultant moments around a vertex have to be calculated and projected to the unit vectors that conform the tilting polygon. The resultant moment is comprised of all forces and moments acting over the connection point (caused by the robot manipulator) as well as of all forces and moments acting on the COG of the mobile platform (inertial and gravitational forces, etc.),

$$\vec{M}_{vi} = -\vec{p}_i \times \vec{F}_r + \vec{M}_r \quad (62)$$

$$M_i = \vec{M}_{vi} \cdot \hat{e}_i. \quad (63)$$

The dynamic stability value α_i related to the i edge can be estimated by considering the mass moment of inertia of the mobile platform regarding the i edge, I_{vi}:

$$\alpha_i = (I_{vi})^{\sigma_i} \cdot M_i \quad (64)$$

Employing

$$\sigma_i = \begin{cases} +1 \; for \; M_i > 0 \\ -1 \; otherwise. \end{cases} \quad (65)$$

The MHS is defined as the critical value (smallest dynamic stability α) regarding an edge of polygon i, indicating thus the critical tilt edge

$$\alpha = \min(\alpha_i) \quad (66)$$

which must be interpreted as

$\alpha > 0$ System is stable
$\alpha = 0$ System is critically stable
$\alpha < 0$ System tends to tip over the edge i.

The MHS method also provided information about the system stability in relation to all possible tip-over edges. By continuously calculating and monitoring the stability value of each edge during the operation of the mobile manipulator, it is possible to determine the risk of tipping over and, if applicable, which edge is most likely to be affected.

The effectiveness of the last three mentioned methods were already evaluated and compared by other researchers. The authors in [101] confirmed that the ZMP method differs substantially from the other two methods because of the fact that the ZMP does not directly consider the inertial moment of the bodies and the height of the COG. Furthermore, the FA method requires much more computing power. In turn, the MHS demands much lower computing power and considers the system to be dynamic. Roan et al. in [102] carried out comparisons by using a real mobile manipulator whose COG location was fixed during the whole set, obtaining similar results to those in [101].

In summary, it has been shown from this review that the MHS technique appears to be the most suitable of these methods to detect instability conditions in mobile manipulators.

3.3 Tip-over prevention approaches employing the robot manipulator

Generally speaking, there are two basic approaches currently being adopted in research to solve the problem of tip-over avoidance for the mobile manipulators:

1. either the traveling speed of the mobile platform is reduced/suited or,
2. the robot manipulator takes another position/orientation.

The combination of both approaches is also possible [20,22,103].

Perhaps the best-known studies include those carried out by Rey et al. [20], who proposed a strategy based on the FA for the tilting detection in such a way that, in case an instability is detected by FA, the mobile manipulator assumes a predefined categorized safe position [20,104]; the major disadvantage of this method, however, is that the executed motion planning task must be completely aborted for the reposition, leading to big delays during the operation time. He [105] estimated the stabilization by the so-called contact force method, which employs the contact force between the mobile platform wheels and the ground to trigger a repositioning of the robot manipulator (see Figure 23). The new manipulator's joint arrangement depends on the system COG and its speed, adopting the position in which the contact force is the same at all contact points (wheels).

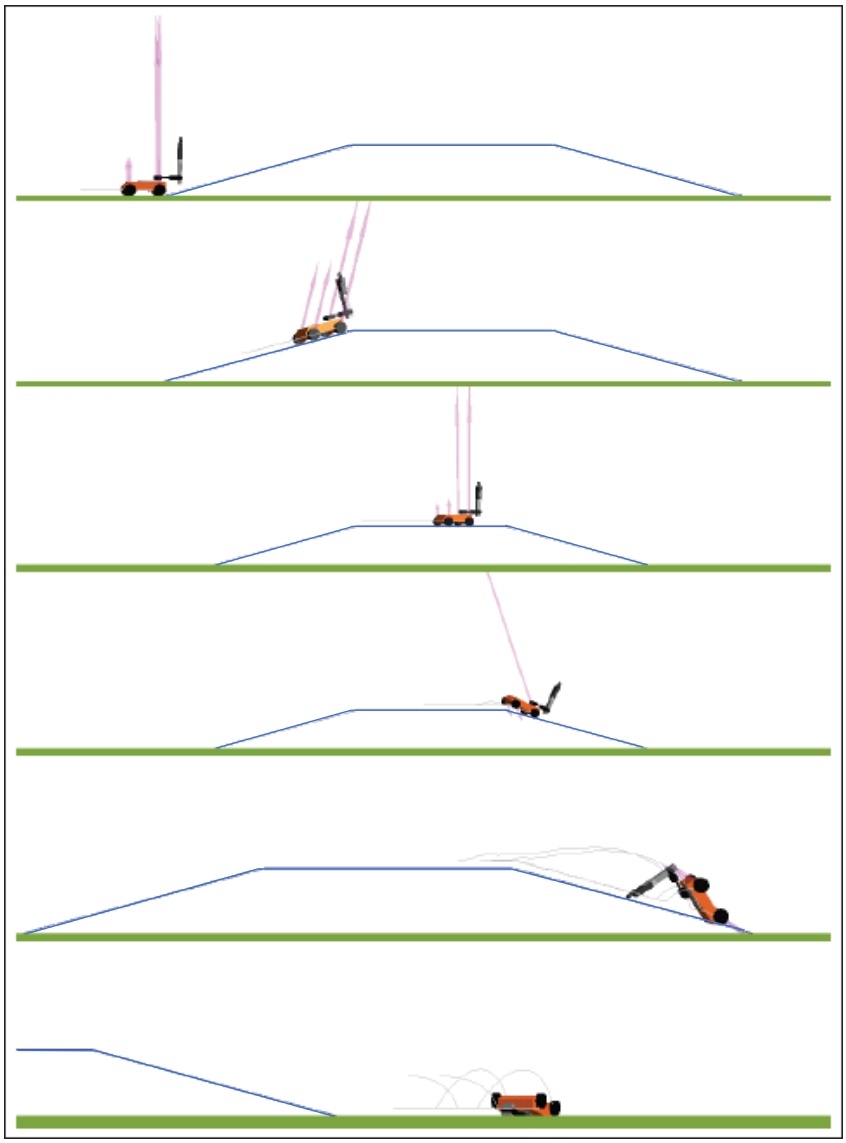


Figure 23 *Ramp crossover with tip-over avoidance algorithm of He* [105].

The limitation of this approach is the measurement of the contact forces between the wheels and the ground, which implies the use of additional sensors and its integration on the wheels. Hatano and Obara [106] employed a highly simplified mobile manipulator consisting of single link with mass. As soon as an instability is detected by means of ZMP, the manipulator is moved to a predefined position, generating a force acting against the tilting moment. This principle is displayed in Figure 24.

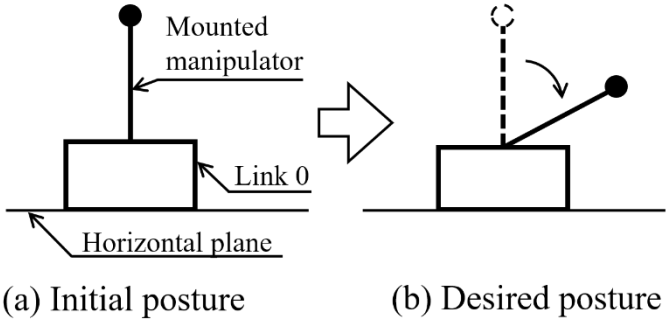


Figure 24 *Stabilization principle of single link mass by Hatano and Obara* [106].

Ding et al. [22] algorithm detected risk of tip-over by means of Improved Tip-Over Moment Stability Criterion (ITOMSC). It defined the position and orientation the robot manipulator's joints should adopt or the speed the mobile platform should assume to accomplish the calculated optimum tilting moment for the current state of the mobile manipulator. The simplified algorithm is represented in Figure 25.

Once again, the mobile manipulator aborts its original motion planning task, standing still during the stabilization and resuming it after the system is considered stable. The new position/orientation for the robot manipulator's joint does not have to be part of the original trajectory path.

Input: $\theta_m, m_m, m_r, d_{zr}, {}^{m}Z_{mc}, \mathbf{a}, TOM_1, TOM_2, q^i_{now}, \dot{q}^i_{now}, \ddot{q}^i_{now}(i = 1 \ldots 6), \dot{\xi}_{now}$
Output: $\dot{q}^i_{next}(i = 1 \ldots 6), \dot{\xi}_{next}$

While not The task complete
  if Tip-over detection cycle arrival
    if The ground can provide enough tractive force
      Calculating $F_{rZ}, M_{rX}, M_{rY}, \mathbf{a}$ and G_M
      Calculate the TOM and the TOM_{max}
    if $max\{TOM\} \leqq TOM_1$
      if $\{TOM\} \geqq TOM_2$
        Calculating the optimization function

$$\sigma = \frac{1}{2}\sum_{i=1}^{n} \| \Delta TOM_{ii+1} \|$$

        if The manipulator in the optimal configuration
          Changing the wheeled mobile platform's velocity by

$$\dot{\xi} = -k_m \frac{\partial \sigma}{\partial \xi}$$

        else Adjusting the onboard Manipulator according to

$$\dot{q}_i = -k_i \frac{\partial \sigma}{\partial q_i}$$

        end if
      else Continue the task
      end if
    else Reporting error, sending stop request and exit
    end if
  else Reporting error, sending stop request and exit
  end if
else Continue the task
end if
end while

Figure 25 *Tip-over prediction and avoidance algorithm by Ding et al.* [22]. *For nomenclature, refer to* [22].

The algorithm presented by Huang and Sugano [103] (shown in Figure 26) calculates an unadjusted optimal trajectory before the mobile platform starts moving. This algorithm inspects which points of the whole trajectory do not fulfill the ZMP criteria and recalculates these critical points as often as necessary until the criterion is fulfilled for each point. Only then, the mobile manipulator is ready to operate.

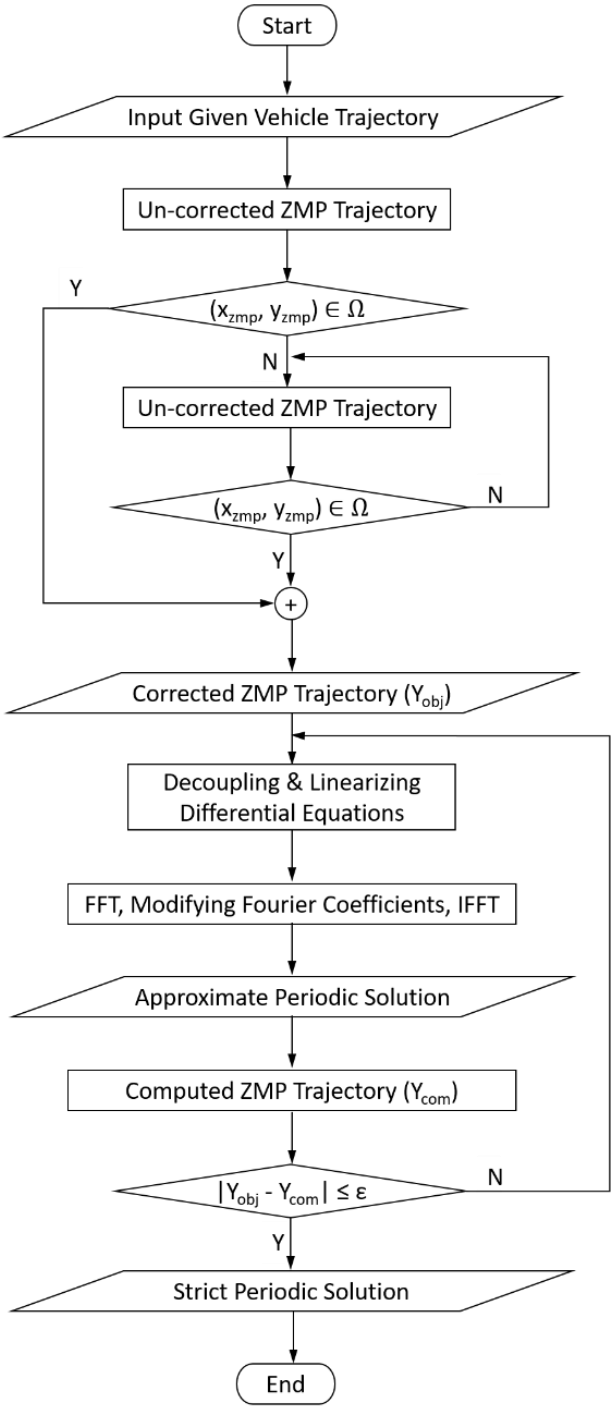


Figure 26 *Algorithm of motion planing for maintaining stability by Huang and Sugano* [103].

In theory, a tilting risk by means of ZMP will be never detected during a task execution, if the mobile manipulator follows the optimized pre-calculated path. This approach enormously manipulates the motion and path planning of the robot manipulator and, perhaps, the most serious disadvantage of this method is that it cannot react to unexpected behaviors (e.g., abrupt braking maneuvers).

Furuno, Yamamoto and Mohri [107] also implemented a similar method based on the ZMP for tilting detection and a predefined stable trajectory path, which adjusts beforehand the position of the robot manipulator and the movement of the mobile platform. The same limitations apply to this approach.

Kim et al. [108] suggested a real-time null-space motion approach, which is only applicable to robot manipulators with kinematic redundancy.

A different real-time method is proposed by Li and Liu [109,110] as well as by Alipour et al. [104], who implemented a fuzzy logic to get non-numerical statements, which depends on the joint positions, velocities and accelerations. Alipour et al. [104] triggered the fuzzy logic algorithm according to the information delivered by the MHS for the tip-over detection, as can be seen in Figure 27, whereas Li and Liu [109,110] rely upon the contact forces between the wheels and the ground.

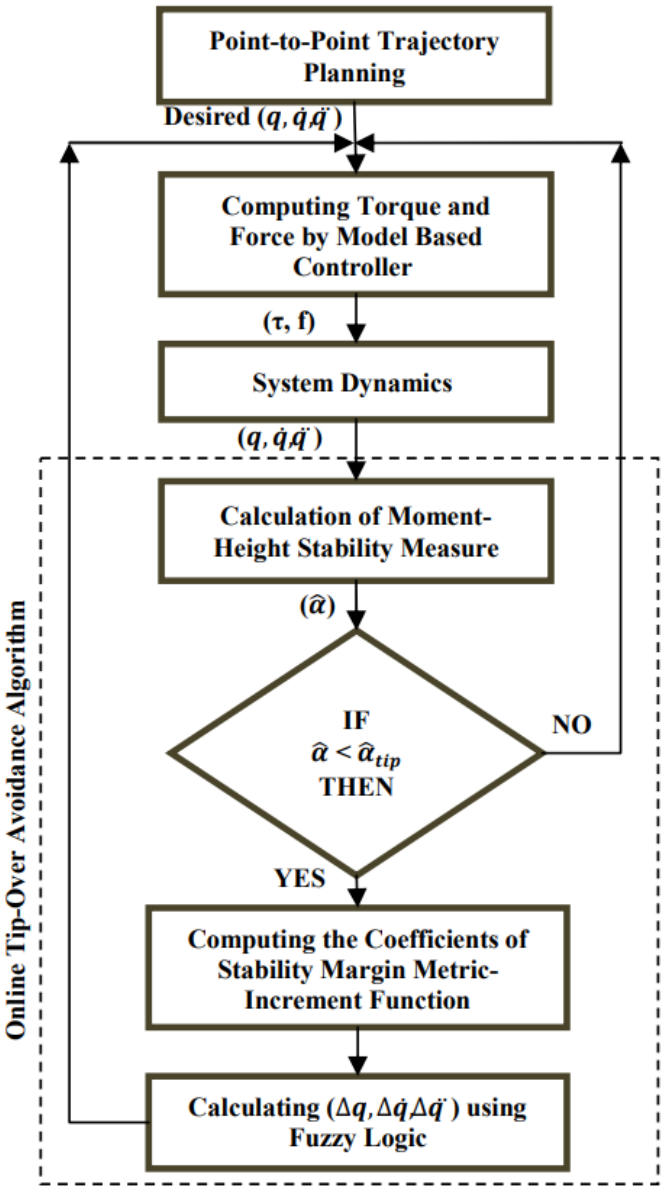


Figure 27 *Fuzzy logic tip-over avoidance planner proposed by Alipour et al.* [104]. *For nomenclature, refer to* [104].

So far, however, the mentioned approaches ignore either the real time aspect, in order to react against unexpected abrupt braking maneuvers, the numerical statements (and not just Boolean) in order to detect the degree of tip-over risk or the simplicity of its measurement and calculation methods. Therefore, there is a clear need for a stabilization strategy that covers all these aspects.

This work offers insights into two different stabilization approaches: Approach A, as an external stabilization method that integrates additional actuators onto the mobile manipulator for the compensation of instability torques; and an Approach B, as an incorporated stabilization method in which the mobile manipulator independently brings itself into a stable state.

4 Approach A: Stabilization strategies for mobile manipulators with limited access to the robot controller

This work describes two different approaches to deal with the instability problem affecting small-footprint mobile manipulators, differing from each other in the system to which they will be implemented: mobile manipulators with closed-source or open-source operating systems, respectively.

The first approach, A, focuses on the design and development of stabilization strategies based on external mechanisms. Their purpose is to ensure the tilting stability of mobile manipulators at standstill or during normal plant operation, regardless of the robot manipulator's and mobile platform's controllers. If an external actuator system is used for the compensation of instability moments, an affordable mobile manipulator can be built by simply integrating currently available robot manipulators at the plant onto small footprint mobile platforms. The particular importance of this approach lies in the possible use of low-cost robot manipulators or the re-use of already existing robots/equipment, since its implementation does not require for access or alteration of the robot controllers.

The main challenge the stabilization approach faces is to guarantee for the stability of the mobile manipulator during its operation, even when the mobile platform abruptly starts or stops. In other words, the stabilization mechanisms should be self-adaptive to react to any external influences and, consequently not to affect the human safety in the working areas [111]. Their adequate control strategy should ensure the equilibrium of dynamical forces affecting the system balance in every moment.

4.1 Stabilization strategies employing tilting effect

The first part addressed in approach A analyzes techniques to compensate accelerations affecting the mobile manipulator's stability based on the g-tilt method, using a principle similar to that of the Stewart-platform presented by Graf and Dillmann [62,64,66]. Following the findings reported during the development of the Stewart-platform [62,64,66], it is a fact that external forces affecting a mobile system are mainly caused by sudden and unpredictable accelerations and decelerations of the mobile platforms. The mechanism proposed by Graf and Dillmann [62,64,66] reaches the compensation of these accelerations by implementing a translational displacement of its upper plate, while the inclination of the payload increases (see Section 3.1). Considering that the translation displacement effect only procures short acceleration impulses, longer lasting accelerations cannot be implemented due to the limited the Stewart-platform's workspace [63]. Moreover, if only the gravity force effect (g-tilt) is taken into account, a compensation of braking processes could be achieved by shifting the system's center of gravity (COG).

Therefore, the stabilization strategies presented below propose the use of a distinct mechanism, in which only the g-tilt effect is carried out by independent linear actuators, while the robot manipulator's weight is supported by a universal joint placed on the upper plate of the mobile platform. It is worth emphasizing that the stabilization strategies, thereby, do not involve any active motion of the robot manipulator or the mobile platform itself.

4.1.1 Actuation mechanism

In order to compensate accelerations and decelerations affecting the mobile manipulator, the stabilization mechanism shown in Figure 28 consisting in trapezoidal screw drives mounted on the mobile platform in a delta configuration has been proposed.

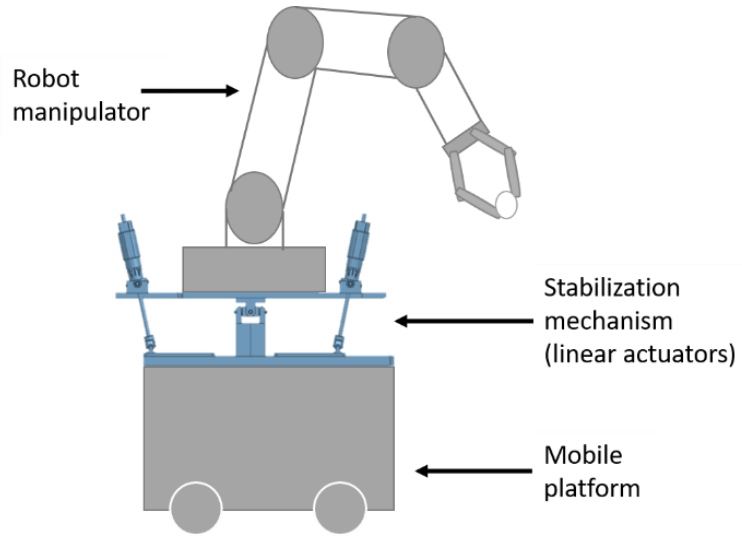


Figure 28 *Linear actuators as external stabilization mechanism for a robot manipulator mounted on a small footprint mobile platform.*

The central column mounted on the mobile platform is a universal joint that connects both subsystems, supports most of the robot manipulator's weight and allows motion in 2 degrees of freedom (DOFs). Each of the ball screw spindles mounted in delta configuration can execute a linear motion. Their motors are fixed to the upper plate by additional universal joints, as illustrated in Figure 29.

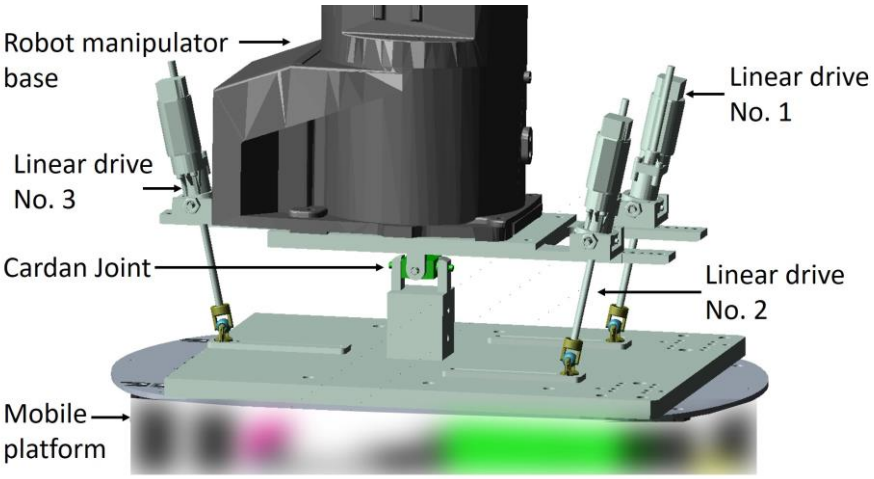


Figure 29 *Linear drives mechanism designed for the stabilization strategy via tilting effect.*

In order to quantify the influence of the accelerations acting on a mobile manipulator, the entire system can be simplified in a two-mass system and analyzed in a two-dimensional space. The acceleration at which the entire system starts to tip-over regarding the front wheels can be estimated by employing the moments generated during the equilibrium of forces ($\vec{M}_{eq}$=0).

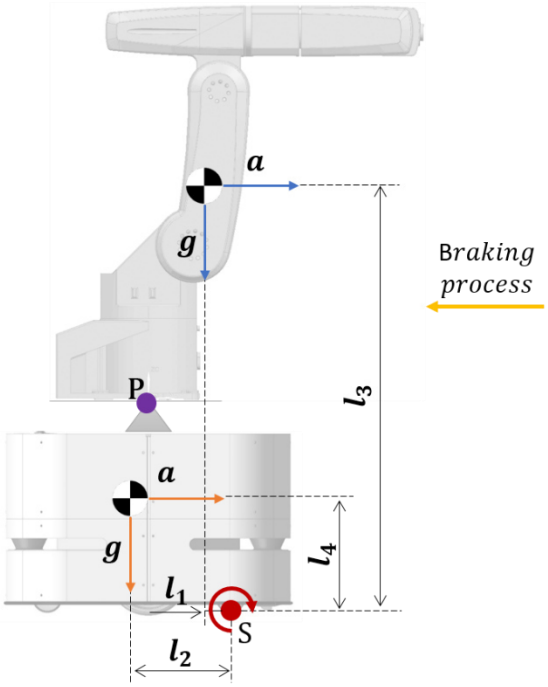


Figure 30 *Free body diagram of mobile manipulator at home position (following* [112]*).*

According to the schematic diagram in Figure 30, the moment of equilibrium $\vec{M}_{eq}$ can be calculated as follows

$$\vec{M}_{eq} = \vec{M}_S = -m_{MP} \cdot \vec{g} \cdot l_2 - m_R \cdot \vec{g} \cdot l_1 + m_{MP} \cdot \vec{a} \cdot l_4 + m_R \cdot \vec{a} \cdot l_3 = 0 \qquad (67)$$

whereby, m_{MP} and m_R are the mass of the subsystems conformed by the mobile platform (MP) and by the robot manipulator (R), respectively; l_1 and l_2 are the horizontal distance between the tilting point S and the COG_{Robot} and the $COG_{MobilePlatform}$, respectively; l_3 and l_4 are the vertical distances between the tilting point S and the COG_{Robot} and the $COG_{MobilePlatform}$, respectively; a is the linear acceleration emerged by the acceleration/braking process performed by the mobile platform, and g is the gravitational force.

Based on Eq. *(67)*, the acceleration at which the mobile manipulator tips over around point S is given by

$$a = \frac{g(m_{MP} \cdot l_2 + m_R \cdot l_1)}{m_{MP} \cdot l_4 + m_R \cdot l_3}. \qquad (68)$$

4.1.2 Stabilization via "Inclining/tilting"

By tilting the robot manipulator in a controlled manner backwards (opposite to the direction of travel of the mobile platform) before the braking process, the robot manipulator COG is shifted to the back-support wheels. During the braking process of the mobile platform, this stabilization strategy exploits the deceleration of the mobile platform itself to compensate the destabilizing forces affecting the entire mobile manipulator and, thus, preventing the system from tipping over. With the robot manipulator tilted backwards, the resulting torque caused by the deceleration of the mobile platform points forwards.

Figure 31 illustrates this idea, whereby ω_t is the tilting velocity of the robot manipulator's COG.

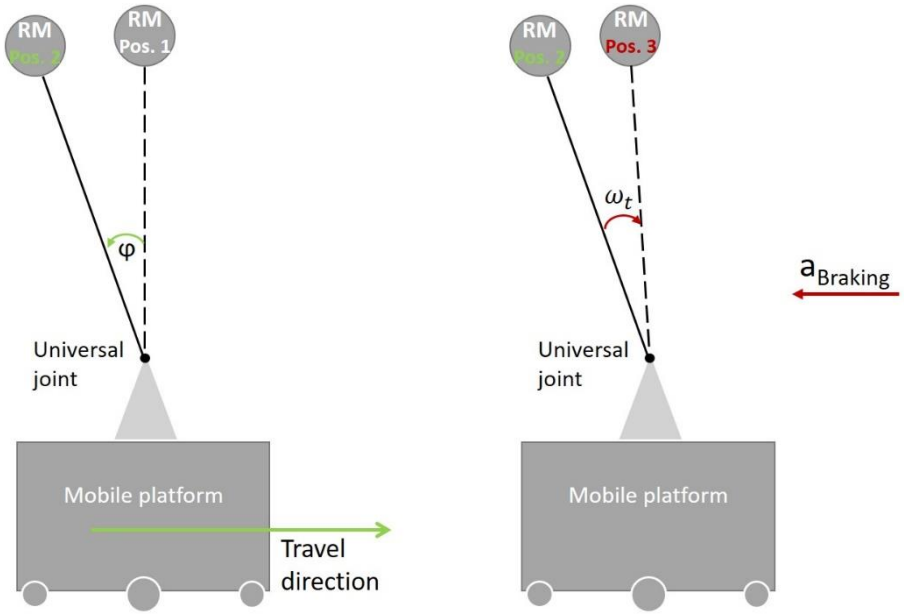


Figure 31 *Description of the inclining/tilting method* [113].

In order to reach a positive effect, the tilt (position 2 in Figure 31) must take place before the mobile platform starts to brake. For this, sensor signals, e.g., from widely used laser scanners on the mobile platforms, can be employed as activation flag for the tilting motion: they reveal that a braking process will be initiated by the mobile platform in the near future.

The acceleration of the mobile platform might serve as input for the control loop of the stabilization strategy, which should calculate the inclination angle of the upper plate of the actuation mechanism needed to compensate presented instability moments.

The outcomes of the implementation of this stabilization strategy employing a testing system are presented in Section 4.5.4.1.

4.1.3 Stabilization based on the "Conservation of angular momentum"

The previous stabilization strategy (through shifting the robot manipulator's COG) cannot be performed during the braking process as there is a counteracting effect, which, however, could also be used for the stabilization: the angular acceleration of the robot manipulator generated during the inclination also affects the degree of stability of the whole system. Due to the conservation of angular momentum, a deceleration of the mobile platform produces a resulting external torque on the overall system that causes a change in angular velocity (angular acceleration). This leads to the idea of generating an angular momentum forwards, opposite to the direction of the angular momentum caused by the deceleration, in order to generate a moment that counteracts the destabilizing moment produced during the braking process.

As a result, in contrast to the previous strategy, the robot manipulator is now impelled forwards (in the direction of travel) during a braking process, as described in Figure 32. For this purpose, the robot manipulator has to be carefully tilted backwards when the mobile platform is traveling straight ahead, before the braking process occurs so that when the mobile platform starts to brake, the robot manipulator can be impelled forwards.

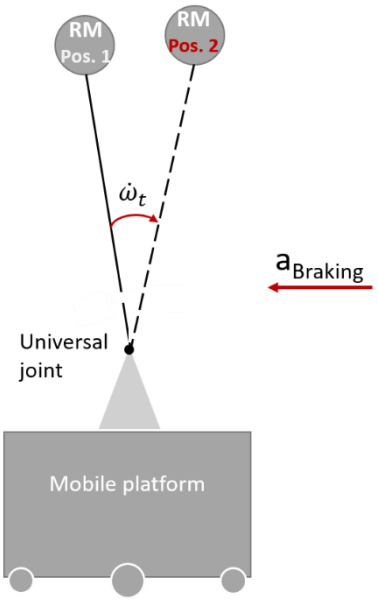


Figure 32 *Description of the conservartion of angular momentum method* [113].

The required angular acceleration $\vec{\dot{\omega}}_t$ can be estimated based on the diagram given in Figure 33,

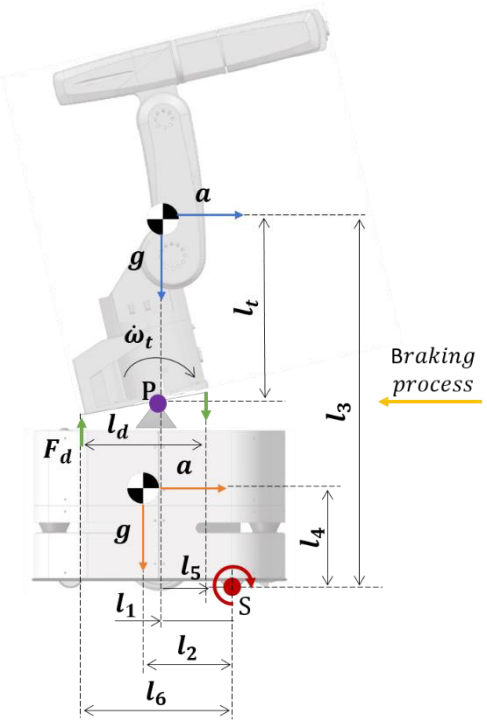


Figure 33 *Free body diagram of the testing system at equilibrium position (following* [112]*).*

The moment equilibrium ($\sum\vec{M}_S = \vec{M}_{eq} = 0$) of the Figure 33, results in

$$\vec{F}_d \cdot l_6 + m_{MP} \cdot \vec{a} \cdot l_4 + m_R \cdot \vec{a} \cdot l_3 - m_{MP} \cdot \vec{g} \cdot l_2 - m_R \cdot \vec{g} \cdot l_1 - \vec{F}_d \cdot l_5 = 0 \qquad (69)$$

whereby, additional to the already specified nomenclature from Figure 30, $\dot{\omega}_t$ corresponds to the angular acceleration of the robot manipulator, l_d is the lever arm between the coupled forces $\vec{F}_d$ that create the angular motion of the robot manipulator and l_t is the distance between the pivot P and COG_{Robot}. Force $\vec{F}_d$ can be estimated after rearranging Eq. *(72)* as follows

$$\vec{F}_d = \frac{-\vec{a} \cdot (m_{MP} \cdot l_4 + m_R \cdot l_3) + \vec{g} \cdot (m_{MP} \cdot l_2 + m_R \cdot l_1)}{(l_6 - l_5)} \quad (70)$$

that corresponds to the force applied by the linear drives. Then, the moment about pivot P (cardan joint) can be determined with

$$\vec{M}_P = \vec{F}_d \cdot l_d. \quad (71)$$

The angular momentum theorem for the robot manipulator can be expressed with

$$\vec{\mathrm{M}}_P = \Theta_P \cdot \vec{\dot{\omega}}_t \quad (72)$$

with Θ_P as the moment of inertia of the robot manipulator with respect to the point P. Thus, the angular acceleration required to compensate instabilities can be calculated with

$$\vec{\dot{\omega}}_t = \frac{\vec{M}_P}{\Theta_P}. \quad (73)$$

Similar to the "inclining/tilting" strategy, the acceleration of the mobile platform might serves as input for the control of the stabilization strategy. The outcomes of this stabilization strategy employing a testing system are presented in Section 4.5.4.2.

4.2 Stabilization strategy using the "Gyroscopic effect"

Data collected during the investigation of the effect of conservation of angular momentum suggested a gyroscope stabilizer as further actuation mechanism to achieve greater moments generated by its precession motion [71,114]. This effect is exploited in technical applications such as stabilization techniques, among other things [115]. Achieving stabilization using gyroscopic effects is a state-of-the-art technique, e.g., in the field of cannon tanks or ships to compensate the effect of the waves [116]. While the gyro stabilizer is considered as a long-established technology, no research has been found that surveyed its application in the stability problem of mobile manipulators.

By accelerating or decelerating the rotating mass of the gyroscopic stabilizer and/or changing the direction of the axis of its angular motion, a moment perpendicular to the axis of rotation is generated [78–80]. As a result, this moment can be used to stabilize the mobile manipulator by mounting the mechanism on the mobile platform. This concept is illustrated in Figure 34. Hence, if the axis of rotation of the mass turns, a torque is applied at the point on the mobile platform that is perpendicular to the axis of rotation.

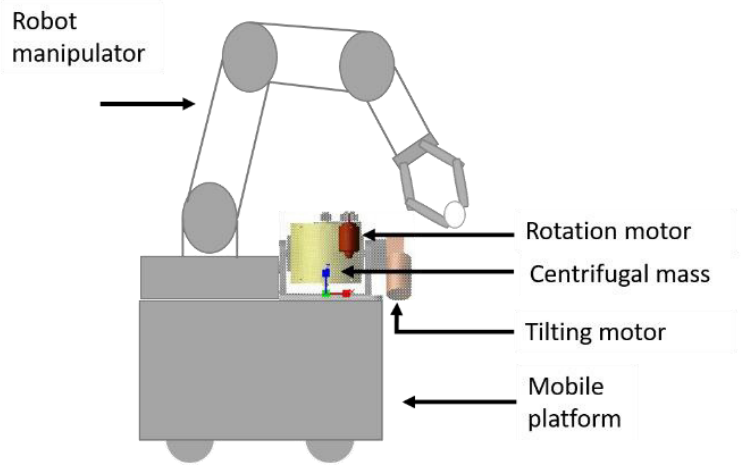


Figure 34 *Gyroscope mechanism on mobile manipulator.*

Some of the advantages of the gyro mechanism are the generation of high torques with small sized components and its easy control. The accurate design and sizing of the gyro stabilizer as well as its control system are crucial for its effectiveness.

The gyroscope given in Figure 35 rotates around its own axis, generating an angular momentum $\vec{L}$ parallel to its rotation axis. If force $\vec{F}$ acts on the rotating mass, a moment $\vec{M}_p$ is produced, implying also a change in angular momentum over time ($\vec{M}_p$=$d\vec{L}/dt$) and pointing in the same direction of the change of the angular momentum $d\vec{L}$. This direction of change is defined by the angle $d\varphi$.

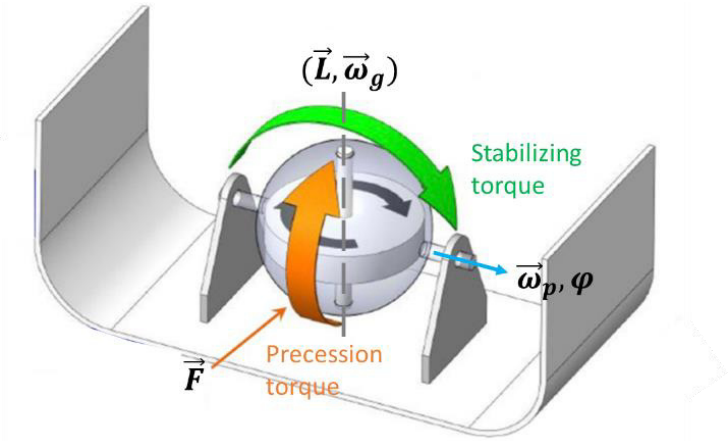


Figure 35 *Gyroscopic principle (following* [117]*).*

The angular velocity of the gyroscope rotational axis during this change in direction, $\frac{d\varphi}{dt}$, can be calculated using the formula

$$\omega_p = \frac{d\varphi}{dt} = \frac{M_p}{L} = \frac{M_p}{\Theta_{zz} \cdot \omega_g} \tag{74}$$

with Θ_{zz} being the moment of inertia of the gyroscope and $\vec{\omega}_g$ its mass angular velocity. The generated torque can be then estimated by the precession moment $\vec{M}_p$ as

$$\vec{M}_p = \vec{\omega}_p \times \vec{L} = \vec{\omega}_p \times (\Theta_{zz} \cdot \vec{\omega}_g). \tag{75}$$

Hence, the design parameters the gyroscope should have to achieve a desired precession moment ($\vec{M}_p$) is determined by its tilting speed ($\vec{\omega}_p$), the rotation speed of the gyro-mass ($\vec{\omega}_g$), and the mass moment of inertia (Θ_{zz}), as described by Eq. *(75)****Error! Reference source not found.***, and can be estimated using the following general rule: the higher the rotational speed of the gyroscope-mass (ω_g), the lower the inclination speed (ω_p) needed to achieve the desired torque. A DC motor can accelerate the mass to a constant angular velocity (ω_g). Then, if the gyroscope is additionally tilted (ω_p) by a second DC motor, the gyroscope generates a precession force ($\vec{F}_p$), resulting in a moment ($\vec{M}_p$) that can be used to stabilize the whole system against instabilities.

Perhaps the most serious disadvantage of the gyro effect is its undesirable inherently side effect of the disturbance torque generated by the precession motion. The components of the resulting precession torque are the compensation torque $\vec{M}_c$ and the disturbance torque $\vec{M}_d$.

$$M_p = \sqrt{\|\vec{M}_c\|^2 + \|\vec{M}_d\|^2}. \tag{76}$$

This disturbance moment ($\vec{M}_d$) makes the system rotate around its own horizontal axis, whose magnitude depends on the angle of deflection: given a precision moment $\vec{M}_p$, the compensation moment ($\vec{M}_c$) becomes smaller with increasing angle of deflection (>90°), while the disturbance moment ($\vec{M}_d$) becomes larger. Therefore, an important consideration for the mechatronic co-simulations to carry out is that only the horizontal portion of the precession moment $\vec{M}_p$, i.e. the compensation torque $\vec{M}_c$, acts on the mobile manipulator to compensate the destabilization.

4.3 Methodology: Modeling a close-to-reality system of the mobile manipulator

The use of real technical systems to conduct investigations in the field is, in most cases, very expensive and time-consuming [40]. Therefore, simulation models that reproduce the behavior of a real system accurately can be employed to carry out investigations that cannot be performed on the real unit for economic and time reasons [41].

Due to the fact that a prototype of a mobile manipulator, including the abovementioned external actuators, would imply large investment of capital and time, a computer-generated environment can be implemented for the development of the three proposed stabilization strategies by means of mechatronic co-simulations. They permit real-time actuation and measurement required for the closed-loop control of the stabilization strategies.

For that purpose, the dynamic behavior of the robotic systems can be described as multibody-system (MBS) models using software tools such as MSC.ADAMS/View. The modeling process of both, the robot manipulator and the mobile platform, as MBS can be derived from the procedure used in [118–120]: as a first step, the dynamic properties (mode shapes and natural frequencies) of each real subsystems are identified separately by means of experimental modal analyses (EMAs). Subsequently, CAD-Models of each robotic system are exported and adapted to build them as MBS models. The obtained experimental findings from the EMA are compared with the results produced by the MBS simulations. By adjusting the modal parameters of the simulation models, the dynamic behavior of the real system is replicated, resulting in a close-to-reality MBS models of the mobile manipulator.

The physical reproduction of the real structure of the mobile manipulator in a MBS model allows for the testing of stabilization strategies using diverse scenarios without the real system being available. The accuracy and reliability of such models depends on how well the simulation correlates with the real system behavior.

In order to accomplish the described approach A, the control loop of the external actuation mechanism (including controllers and drivers) needs to interact with the mechanical model of the mobile manipulator (MBS model) by means of mechatronic co-simulations. As illustrated in Figure 36, mechatronic simulations enable a group of system components, such as mechanical, electrical, software and control, interact together in order to build an integral system model [121]. The prefix “co” is added to the term mechatronic simulation when each of these stand-alone simulations (performed in different software tools) are joined into one single simulation environment.

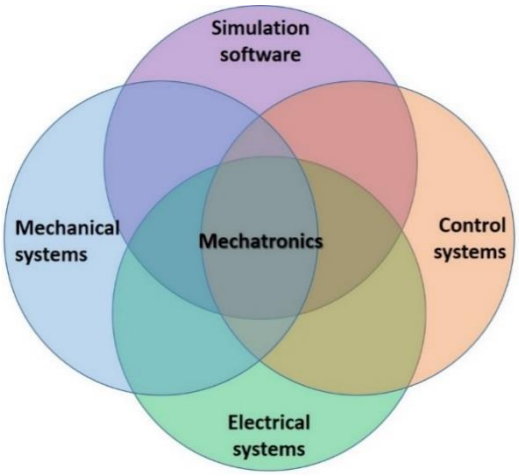


Figure 36 *Principle of co-simulation of mechatronic systems (following* [122]*).*

Despite the fact that the EMA and the MBS are well-known methods, especially in the automotive industry, there is a surprising lack of literature describing those methods applied in robotic systems [123]. For this purpose, the following sections describe how the EMA and the MBS methods have to be adapted for their used on modeling robot manipulators and mobile platforms. Furthermore, this study may be the first comprehensive assessment of performing an EMA carried out for a mobile platform.

4.3.1 Experimental Modal Analysis of mobile manipulators

The dynamic behavior of mechanical systems can be experimentally estimated by means of EMAs. They consist in externally exciting a real mechanical system and measuring the oscillations caused in response to the applied stimulus. The system response function describes the relation between its excitation and response (e.g., force/acceleration) and serves to validate the MBS models, required to perform accurate mechatronic co-simulations.

The necessary equipment to perform EMAs includes a real-time analyzer, piezoelectric 3D accelerometers, vibration mats and impulse hammers with their associated impact tips, as illustrated in Figure 37.

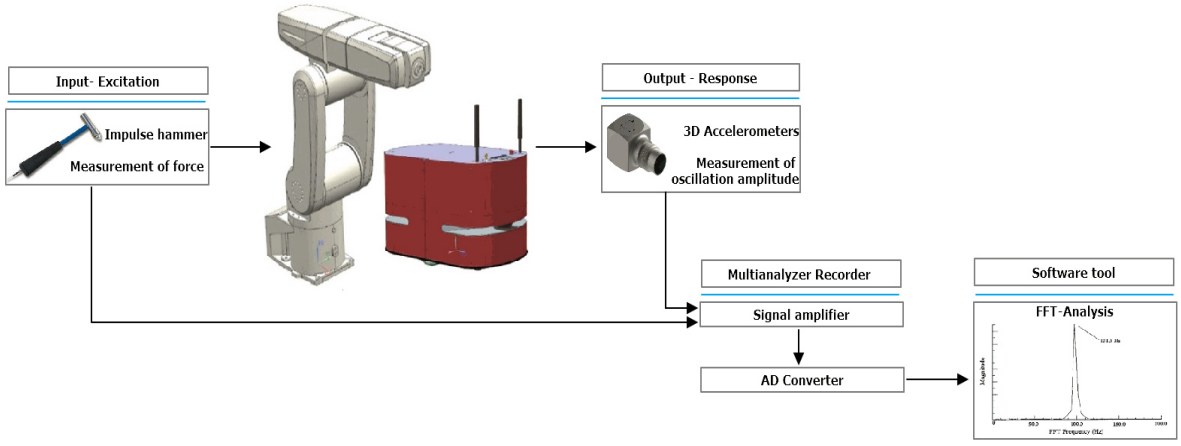


Figure 37 *Test arrangement for the experimental modal analyses.*

The election of suitable excitation and measurement spots are relevant aspects that have to be considered during the experimental set-up of an EMA. The challenge lies in choosing the

appropriate excitation and measurement spots on the structure, without having previous precise knowledge about the dynamic behavior of the real systems.

The risk of ignoring some natural frequencies and mode shapes during EMAs can be significantly reduced if the excitation points are placed non orthogonal to the chosen measurement points. The designation of the measurement points as a first step is therefore suggested. It is also recommended that all directions of the system's motions are excited to ensure that all desired eigenmodes are fully identified in the relevant frequency range. Since the connecting elements of the robot manipulator and the mobile platform are relatively elastic/flexible compared to their bodies, a frequency range up to 500 Hz can be defined for their EMAs [118].

Additionally, the EMAs should be determined in different configurations of the structure in order to corroborate their dynamic linear behavior.

4.3.2 Multibody-system model of mobile manipulators

On the basis of the system response function (transfer function) of robot manipulators and mobile platforms, close-to-reality mechanical models of their real structure can be built as MBS following the modeling hybrid approach previously mentioned in Section 2.2. The mobile manipulators can be modeled with a low level of detail to get good computing performance, but they must be complex enough to guarantee that the reproduction of the real dynamic behavior is not affected by excluding some elements in the model. In other words, the MBS models must contain just the necessary components to describe the dynamic behavior of the real mobile platform and the robot manipulator. Parameters such as masses and moments of inertia are crucial for modeling the MBS simulation models and have to be determined.

For the model simplification, bodies that are firmly connected to each other and elements with the same material can be merged into one or more main bodies. In case different material properties are presented in a merged group, their averaged density and their COG can be combined into a single body parameter. Furthermore, screws, screw nuts, washers and slot nuts can be removed and replaced by fix joints.

An additional measure to simplify the models implies that the structure of most of the robot manipulators and mobile platforms has considerably uncritical deformation in comparison with their joints. Elastic deformations of the bodies themselves are less critical for the structural components and, therefore, might not be considered in the simulations. As a result, their bodies can be modeled as rigid bodies.

To interlink the individual rigid bodies that form the robot's structures, kinematic joints or physical connections (spring-damper systems so-called bushings) have to be located at the articulation point between two rigid bodies. They are assumed to be mass-free with only the constraint forces acting on their respective links. Each bushing had to allow the real DOF of the connecting element i.e. that for the robot manipulator, a bushing would allow only one rotational DOF for emulating the robot manipulator axis. Figure 38 and Figure 39 show examples of arrangements of the joint elements for a mobile platform and a robot manipulator, respectively.

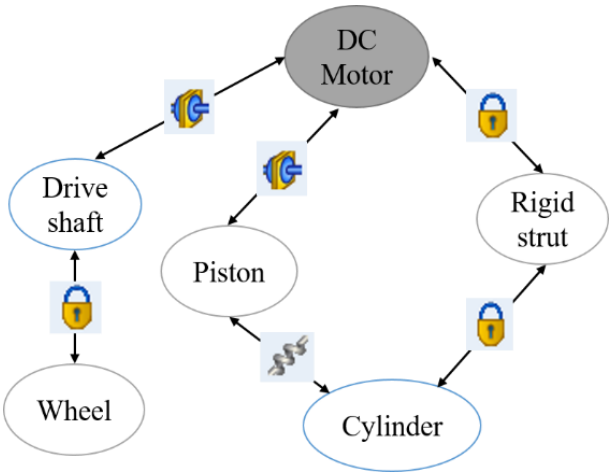


Figure 38 *Graphical topology of a basic wheel suspension mechanism of a mobile platform.*

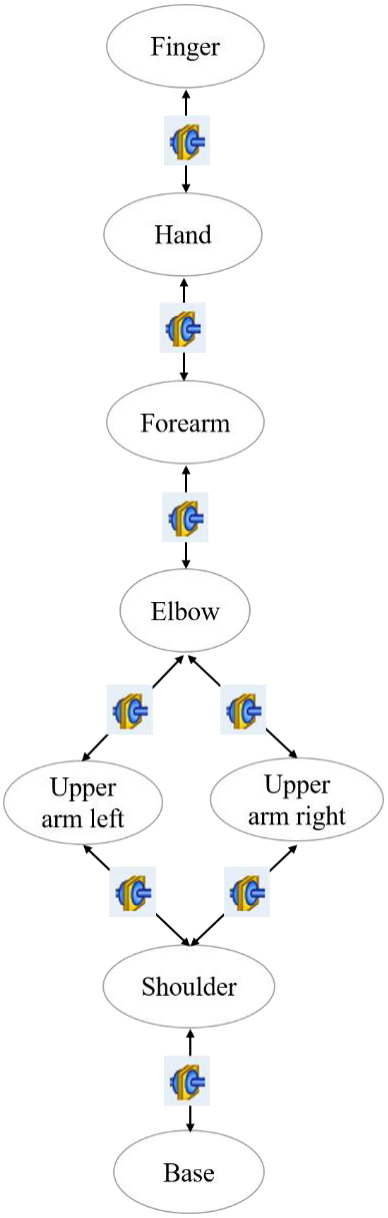


Figure 39 *Graphical topology of the elastic joint elements for a robot manipulator.*

In every bushing, both translational and rotational constraints and applied forces and torques have to be replicated by introducing stiffness and damping values for each available DOF. It is assumed that the stiffness of the bodies is significantly greater than the stiffness of its joints, being the responsible of reproducing the first natural frequencies of the robots (based on [124]). In consequence, the interaction between the rigid bodies and the entire set of bushings defines the system oscillation behavior: Force and moments that are applied to the bodies by the bushing or acted from the bodies to the bushings, are comprised by a six components constraint ($\vec{F}_x$, $\vec{F}_y$, $\vec{F}_z$, $\vec{M}_x$, $\vec{M}_y$, $\vec{M}_z$), whose stiffness and damping values had to be adjusted. A good best practice for the adjustment of the model's stiffness coefficients is to first estimate them by using finite element analysis (FEM) or, e.g., for the stiffness of bearings, by using manufacturer's catalogues.

The contact connections between the ground and the wheels of the mobile manipulator can be considered as Coulomb's friction contacts, since the wheels should slide as in the real mobile platform. Here, the mass of the wheels, which is firmly connected to the ground by a spring, slides in the earth's gravitational field over a plane (ground) with a coefficient of friction μ_f. The significant feature of a Coulomb's friction contact is that the frictional force always acts against the movement of the mass [125]. Thus, as soon as the spring force is smaller than the friction force, the movement of the mass stops.

The concrete values for the stiffness and damping parameters of the contacts between the ground and the individual wheels are usually unknown and difficult to estimate. Large values in the simulation give small positional inaccuracy, but on the other hand, lead to long simulation time [41]. Referring to the statement of Glöckler [41], a reasonable simulation time with a good accuracy can be guaranteed if these values are empirically set high enough, without letting the simulation time increase excessively.

The MBS models must describe the first natural frequencies and mode shapes of the real systems to ensure a suitable design of the control algorithms for the stabilization strategies. For this, the stiffness and damping values of the bushings in the MBS simulation models have to be parametrized. The natural frequencies and mode shapes from the EMAs served as a reference for the model parametrization.

For an objective comparison between the modal parameters collected from the EMA and those obtained from the MBS, the model can be provided with small massless bodies at the positions the accelerometers are attached on the real system during the EMA. During the modal simulations, each modal shape of the corresponding natural frequency has to exhibit the same oscillation as in the real system, i.e. the obtained natural frequencies and mode shapes can be compared with the experimental findings from the EMA [40].

In order for the simulation to reproduce similar dynamic behaviours with sufficient accuracy, the stiffness and damping parameters of the elastic joints have to be iteratively adjusted by increasing or decreasing their values one at a time. Their adjusted values are subject to the natural frequencies and mode shapes generated during the corresponding simulation. If the modal parameters of the MBS model correspond to the modal parameters obtained experimentally, the MBS model is ready to be employed for the mechatronic co-simulations [40]. In this respect, the model of the mobile manipulator might be close enough to reality to allow a subsequent application in the development of the stabilizing algorithms.

The described MBS modeling process is summarized in Figure 40.

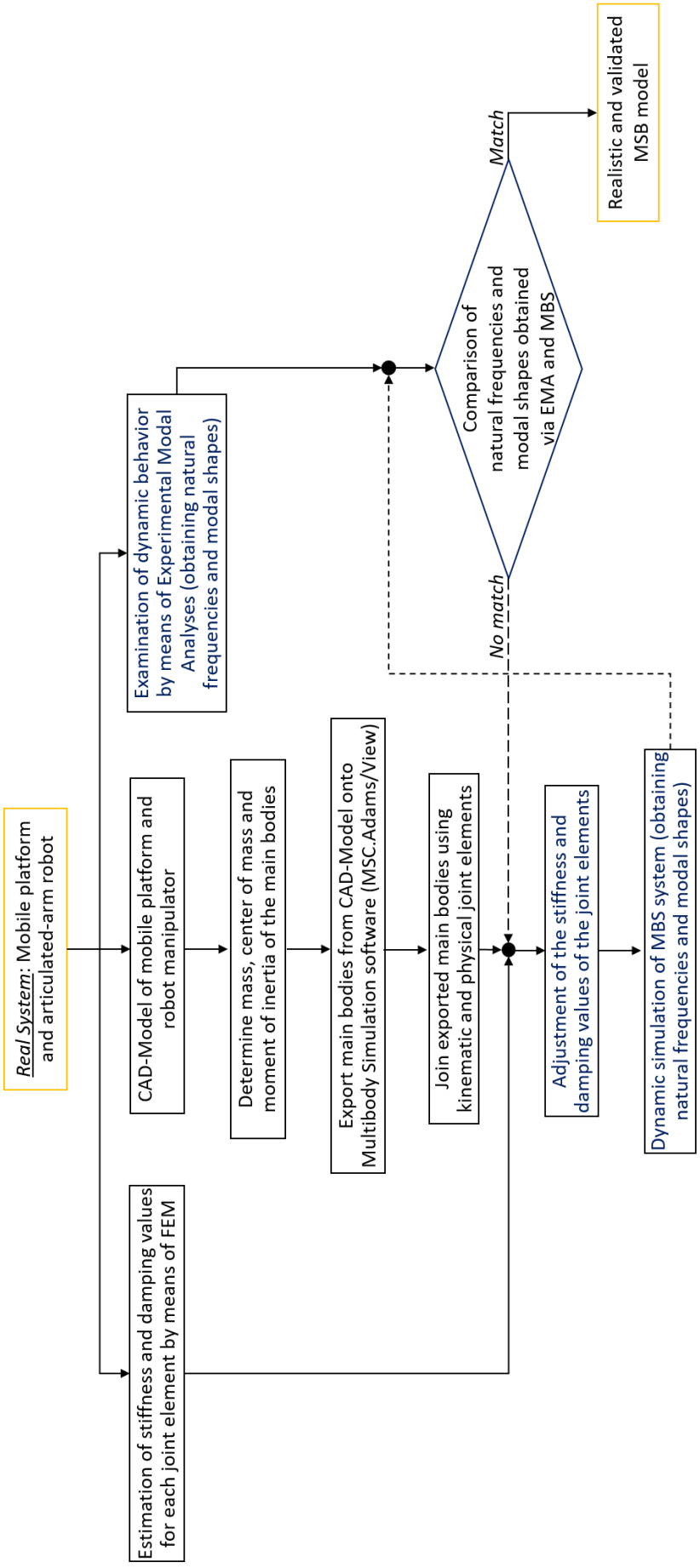


Figure 40 *Iteration procedure for the MBS-modeling of a real system.*

4.3.2.1 Optimization of multibody-system model using a full automated algorithm

The degree of correlation between a real system and its corresponding MBS model can be influenced accordingly to the following statements:

- The system dynamics is described by its modal parameters, comprised by the natural frequencies and mode shapes as well as their reciprocal damping ratio. They are unique for each system.
- The amount of natural frequencies resembles the amount of DOF of the system.
- Each identified natural frequency belongs to a certain modal shape.
- The mode shapes represent only amplitude ratios, not the absolute amounts of the deflections.

With increasing model complexity, the adjustment process of the modal parameters of a MBS model is difficult or even impossible to implement manually. In order to facilitate this adjustment process and, at the same time, increase the quality and reliability of the system's dynamic behavior in the simulation models, a computer-aided parametrization method is implemented to minimize the numerical deviation between the experimental and the simulative modal parameters. For the implementation of the computer-aided parametrization method, the open source software DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) is adopted. It provides a variety of algorithms for different applications, e.g., for the parameter study, statistical analysis, design of experiments, quantification of uncertainties, calibration and optimization of complex physical systems [126,127].

The adopted fundamental correlation is an entirely quantitative comparison of the determined modal parameters, described by the system natural frequencies and the Modal Assurance Criterion (MAC). The goal of the parameter identification is to ensure that the experimentally determined values, natural frequencies and mode shapes of the real system, match the simulation accurately. If it is assumed that the mass distribution as well as the mass moments of inertia of the models sufficiently represent the reality, only the stiffness and damping coefficients of the joint elements (bushings) need to be parametrized.

Hence, the developed algorithm in DAKOTA extracts the modal parameters calculated by the solver of the MBS software tools, in this specific case MCS.ADAMS/View, and verifies if the termination criterion (correlation with the modal parameters from EMA) is fulfilled during the last iteration. If so, the algorithm finishes the parametrization, otherwise, it starts a new simulation iteration with new adapted parameters.

The selection of the proper numerical method for the optimization algorithm depends on the nature of the problem under consideration. For the optimization of modal parameters in MBS models, the two following elemental approaches can be adopted:

- Gradient-based methods. They use gradients to minimize error with regard to a target function value, until the error cannot decrease further (local minimum) or until a defined termination criterion has been reached. A gradient-based method always converges at the same optimum employing the same initial values.
- Heuristic methods. They are based on plausibility or analogy considerations that can only be substantiated experimentally. The best known methods are the so-called evolutionary approaches, in particular, the genetic algorithms as a mathematical imitation of the biological mechanisms occurring during the evolution processes in the nature. The

hereditary information of each individual forms its distinctive parameters and is changed by random combinations and random alterations of the information (mutation). By changing the "genetic" information, an adaptation from generation to generation (from iteration to iteration) takes place. Like the natural selection process, only the fittest individuals (parameter sets) of each generation (iteration) survive. The iteration process remains running until a defined termination criterion is reached. The environmental conditions are decisive for the selection, given by search space boundaries and restrictions. Due to their stochastic character, evolutionary algorithms are able to find both, local (suboptimal combinations) and global optimum.

The nl2sol algorithm of DAKOTA [128] is a gradient-method algorithm that focus on finding local minimum within specified lower and upper bounds [129]. Its termination criterion is defined by tolerance ranges. Moreover, the coliny_ea algorithm (from the SCOLIB package [130]) is an evolutionary algorithm, particularly suitable for such systems whose initial parameters cannot be predicted or are situated within a wide parameter scope. Both algorithms are employed by the conceived automated parametrization method, introduced in the following sections.

4.3.2.1.1 Conceptualization of the parametrization algorithm

In the parametrization algorithm, an interaction between DAKOTA and the MBS model of the system to be characterized takes place, in which the MBS model operates as a black box system for DAKOTA. During an iteration step, DAKOTA specifies initial parameters, which are automatically read into the MBS model. Then, the MBS software tool performs a modal simulation. The simulation results with this set of parameters are then processed and transferred again to DAKOTA. After each iteration, the results are evaluated by DAKOTA and, if the termination criterion is not fulfilled, a new set parameters is output to the MBS model. This iteration loop runs until the defined termination criterion is met.

The following steps are compulsory to perform the parameterization of the MBS models employing the developed parametrization algorithm:

1. Conduct modal analyses (EMA) to determine the modal parameters of the real system.
2. Process the modal parameters by means of a mode transformation.
3. Prepare the MBS simulation model (with unknown/uncertain parameters) in MSC.Adams/View, considering the same positions and orientations the measuring spots had during the performed EMA.
4. Select the numerical method for the parametrization (gradient-based or evolutionary).
5. Setup interface in DAKOTA.
6. Specify the parameter set (stiffness and damping coefficients).
7. Arrange the data exchange between DAKOTA and MBS model (preprocessor and postprocessor), in which the measured modal parameters need to be provided as reference for parametrization algorithm.
8. Adapt the MBS model solver script (for MSC.Adams/View, the *.cmd file) with the parameter set.
9. Start the parametrization. DAKOTA triggers the simulation of the MBS model with the predefined parameter set. Then, DAKOTA reads out and processes the simulation results

from the set. It evaluates the correlation and, if necessary, specifies a new parameter set. The identification is aborted by DAKOTA when the termination criterion is reached.

All mechanisms customized for their use in the MSC.Adams/View environment that are involved in the parametrization procedure are described in Figure 41.

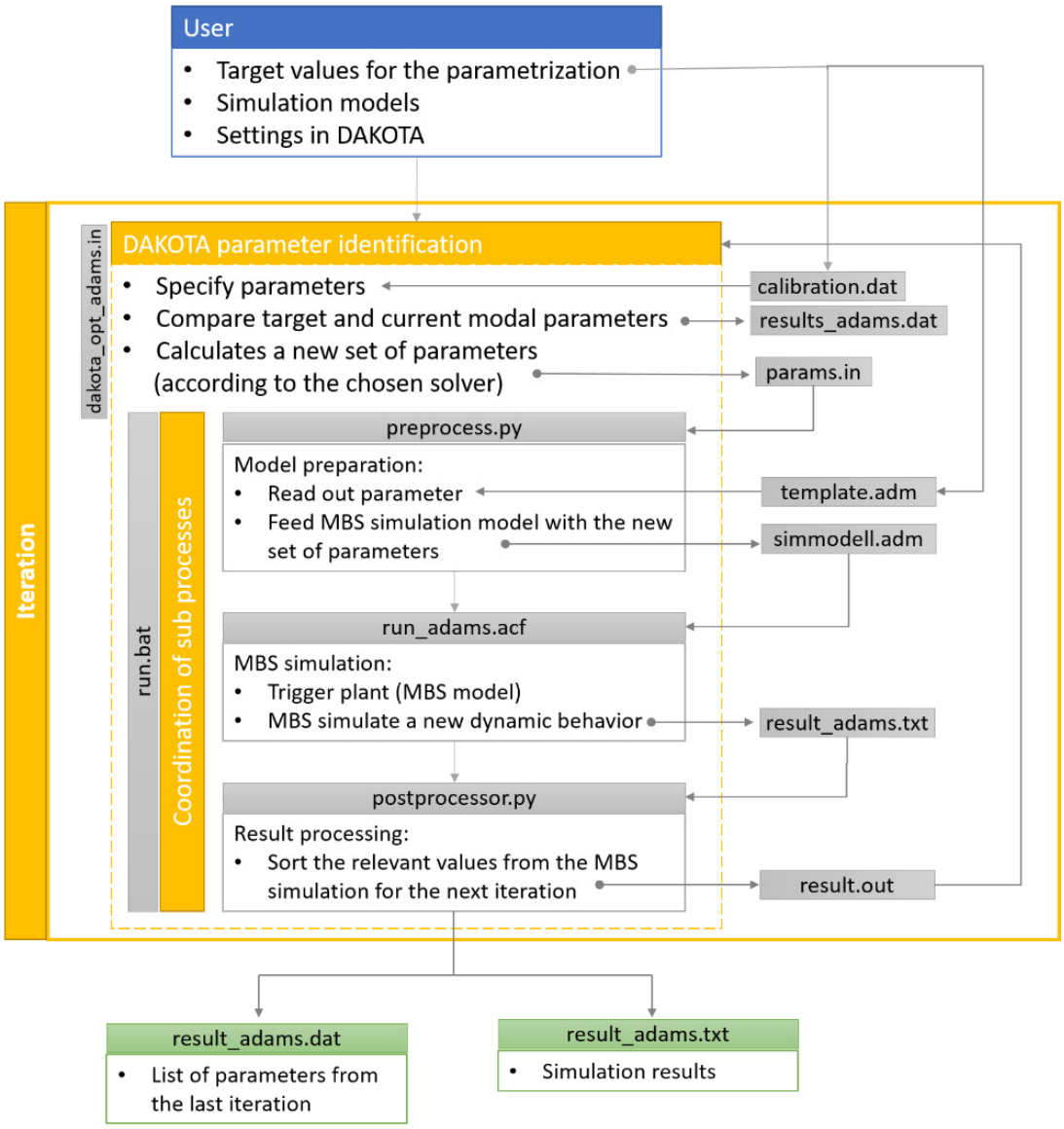


Figure 41 *Functions/file interactions of the algorithm implemented in DAKOTA (following* [131]*).*

4.3.2.1.2 Enhanced parametrization algorithm based on multiple-mass oscillators.

The further development of the elemental parameter identification mechanism provided by DAKOTA has its basis in the behavior of the two-mass oscillator shown in Figure 42.

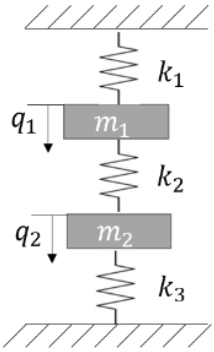
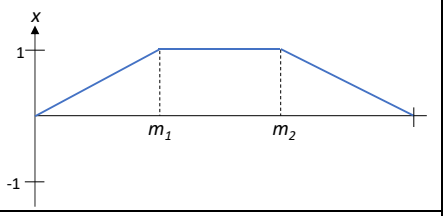
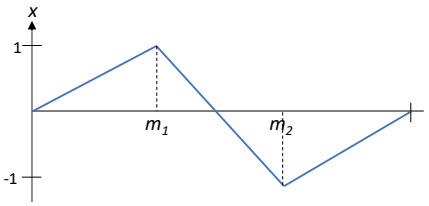


Figure 42 *Two-mass oscillator.*

Since the system has two DOF, two natural frequencies are expected. The masses of the adopted system were identical, being 0.358 kg each. The stiffness of the employed springs was analytically calculated based on experimentally measured elongations, being 0.49 N/mm for all three. Considering these masses and stiffness values, the system natural frequencies and mode shapes can be calculated according to the Lagrangian equations of the second kind [132], resulting in 5.9 Hz and 10.2 Hz for the specific two-mass oscillator employed, as summarized in Table 7.

Table 7 *Modal parameters of the two-mass oscillator anallytically obtained (following* [133]*). The amplitudes of the eigenvectors are normalized.*

Natural frequency	Graphical representation of mode shapes
5.9 Hz	
10.2 Hz	

A simple MBS model of the two-mass oscillator (including its physical quantities) has to be built, as the one shown in Figure 43.

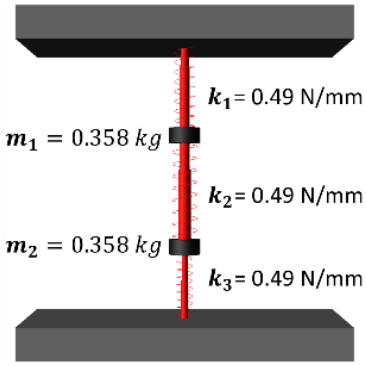


Figure 43 *MBS model of the employed two-mass oscillator (*[133]*).*

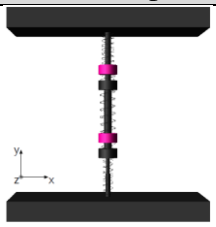
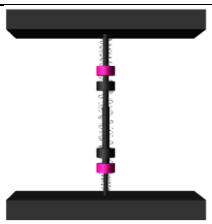
The acquired natural frequencies can be adopted as target reference values for the parameter identification. The first parametrization attempt of the adopted two-mass oscillator using the gradient-based algorithm required 26 iteration steps to ascertain the stiffness coefficients presented in Table 8.

Table 8 *Stiffness coefficients calculated by the first version of the parametrization algorithm in DAKOTA (following* [133]*).*

Stiffness (N/mm)	k_1	k_2	k_3
	0.433	0.467	0.433

Table 9 exemplifies the natural frequencies and mode shapes obtained from the modal simulations in MSC.Adams/View for the employed two-mass oscillator.

Table 9 *Modal parameters for the two-mass oscillator: The system at rest is shown in black; the magenta blocks illustrate the masses at their maximum deflection during the oscillation process* [133].

Natural frequency	Mode shapes
5.53 Hz	 Both masses oscillate in phase
9.83 Hz	 The oscillate phase is shifted 180°

Although in the MBS simulation no big deviations were presented regarding the natural frequencies and the mode shapes (f_1=5.53 Hz and f_2=9.83 Hz), the stiffness coefficients calculated by the algorithm differ from the analytically determined spring stiffness of 0.49 N/mm by up to 11.5%.

In order to improve the robustness of the parameter identification, the two-mass oscillator was extended to the system in Figure 44, comprised by 5 masses and 6 springs.

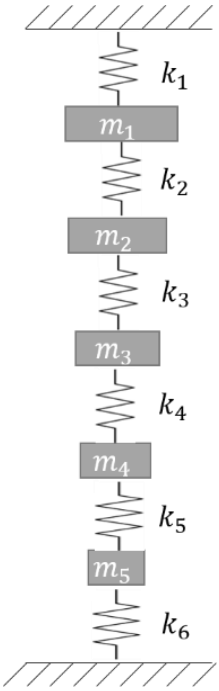


Figure 44 *Five-mass oscillator (m_1=5 kg, m_2=4 kg, m_3=3 kg, m_4=2 kg, m_5=1 kg, k_1=6 N/mm, k_2=5 N/mm, k_3=4 N/mm, k_4=3 N/mm, k_5=2 N/mm, k_1=1 N/mm).*

For the first trial, the lower and upper limits for the spring´s stiffness in the parametrization algorithm was set to 0.1 N·mm and 9 N·mm. Using these boundary limits, DAKOTA determined the natural frequencies and stiffness coefficients shown in Table 10:

Table 10 *Natural frequencies and sitffness coefficients for the five-mass oscillator obtained by the parametrization algorithm (following* [133]*).*

	Natural frequencies (in Hz)					Stiffness coefficients (in N/mm)					
	ω_1	ω_2	ω_3	ω_4	ω_5	k_1	k_2	k_3	k_4	k_5	k_6
Analytical value	3.8	8.1	11.3	13.9	16.1	11.0	9.0	7.0	5.0	3.0	1.0
Determined by DAKOTA	3.8	8.1	11.3	13.9	16.1	10.2	6.7	5.2	5.5	4.0	1.7

Despite the fact that all natural frequencies matched the experimental values (see Table 10), the stiffness coefficients still differ significantly from the analytical results. This suggests that, for complex structures with many bodies and joint elements, only considering the natural frequencies as reference for the termination criteria is not sufficient for parameter identification.

For this reason, the eigenvectors are integrated in the evaluation process (PostProcessor.py): the Modal Assurance Criterion (MAC) calculates the angles between the simulated (MBS) and experimentally (EMA) determined eigenvectors. Then, MAC checks the orthogonality properties of the two vectors [134], which indicates the degree of compliance and linear dependence of the two complex eigenvectors. Hence, the mode shapes can be not only qualitative but also quantitatively evaluated.

Merely equal modes from the EMA and from the MBS can be compared with each other. Therefore, only the diagonal values of the MAC matrix (see Section 2.1) need to be computed

and the calibration.dat file has to be extended with the amount of modes to be expected, each of them with the set point 1.0.

It is important to mention that the location of the measurement points in the MBS model has a significant influence in the parameter identification based on MAC; thus, the real positions of each individual accelerometer used during EMAs must match the measurement points attached in the simulation model.

The diagram in Figure 45 shows the MAC of the last parameter identification for the five-mass oscillator, without considering the eigenvectors as additional reference for the parametrization algorithm.

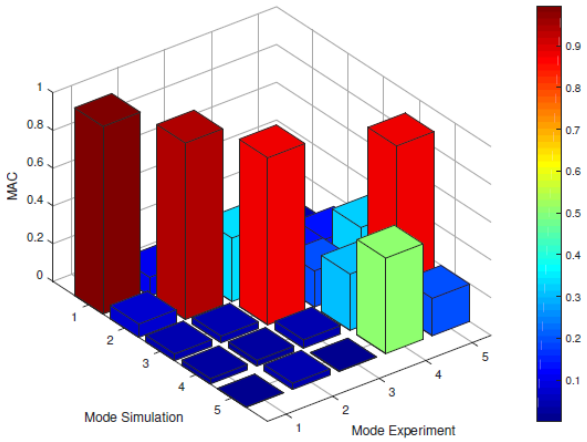


Figure 45 *MAC results without mode matching* [135].

A new parameter identification under the same conditions, but with the implemented MAC as termination criteria was carried out. Although the computed stiffness coefficients exhibited large deviations (see Table 11), the mode shapes presented a better match (see Figure 46).

Table 11 *Natural frequencies and stiffness coefficients obtained by the algorithm with implemented mode matching (following* [135]*).*

	Natural frequencies (in Hz)					Stiffness coefficient (in N/mm)					
	ω_1	ω_2	ω_3	ω_4	ω_5	k_1	k_2	k_3	k_4	k_5	k_6
Analytical value	3.79	8.12	11.34	13.97	16.10	11	9	7	5	3	1
Determined by DAKOTA	3.66	8.10	11.34	14.01	16.08	7.5	6.0	6.4	6.5	3.5	2.0

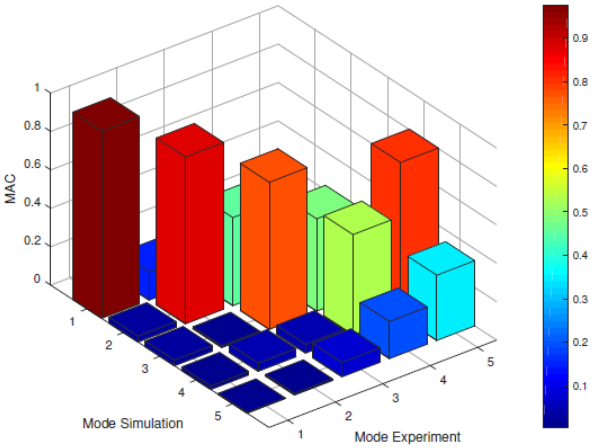


Figure 46 *MAC results with mode matching* [135].

It is presumed that gradient-based algorithm always converges to about the same local minimum optimum due to the restricted preselected boundary limits. Therefore, in order to scan a wider parameter range, the evolutionary optimization algorithm was implemented. As a result, the elements of the MAC matrix represent the good correlation exhibited in Figure 47, Table 12 and Table 13.

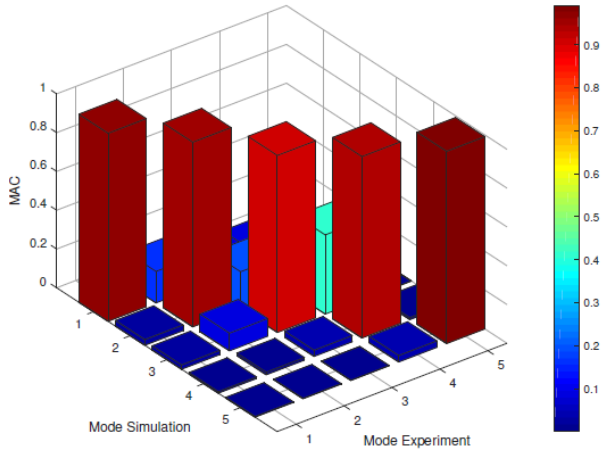
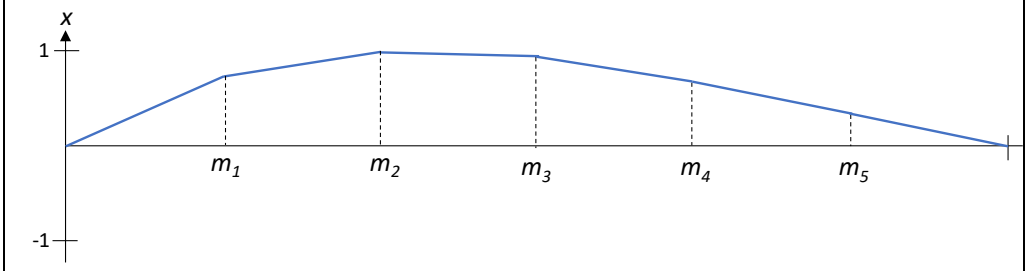
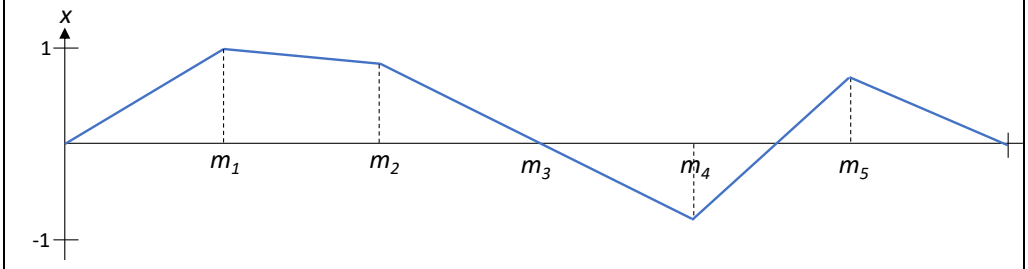
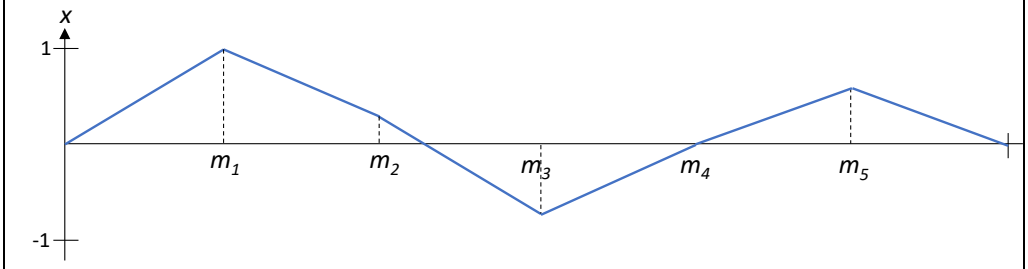
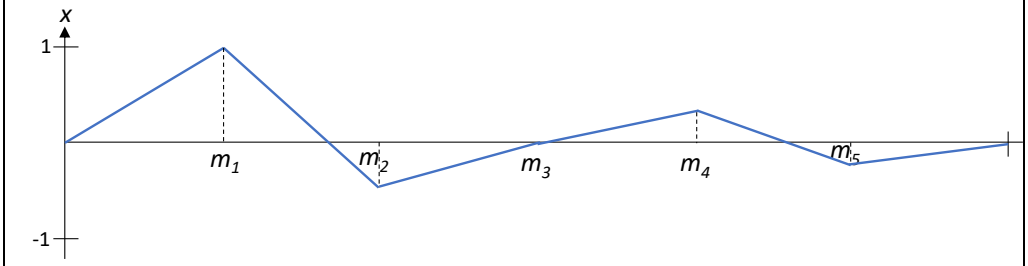
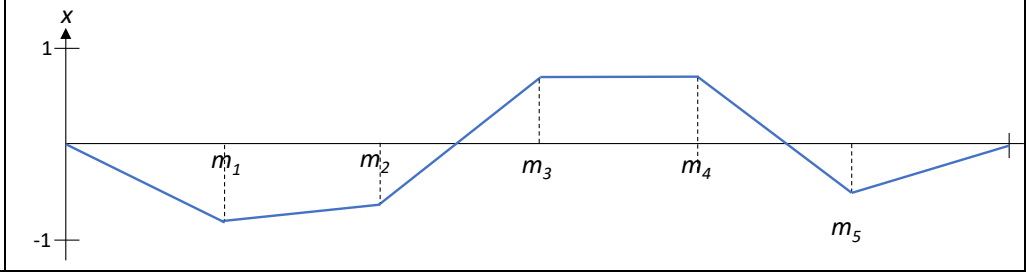


Figure 47 *MAC results with mode matching and employing evolutionary algorithm* [135].

Table 12 *Natural frequencies and stiffness coefficientes obtained by the algorithm with implemented mode matching and employing evolutionary algorithm (following* [135]*).*

	Natural frequencies (in Hz)					Stiffness coefficient (in N/mm)					
	ω_1	ω_2	ω_3	ω_4	ω_5	k_1	k_2	k_3	k_4	k_5	k_6
Analytical value	3.8	8.1	11.3	13.9	16.1	11.0	9.0	7.0	5.0	3.0	1.0
Determined by DAKOTA	4.0	8.0	11.3	13.9	16.0	9.2	8.8	7.5	4.0	2.7	2.2

Table 13 *Graphical representation of the eigenmodes of the five-mass oscillator obtained by the parametrization algorithm* [133]. *The blue lines represent the eigenmodes derivated from the parametrization algorithm and cover the (non visible in the graphical representation) lines for the eigenmodes obtained analitically, proving their congruence.*

Eigenmode	Graphical representation of modal shape
1st	
2nd	
3rd	
4th	
5th	

In order to verify the effectiveness of the developed algorithm, a parameter identification was performed on a simplified 3 DOF mechanism, comparable with a robot manipulator, in which oscillations in all spatial directions took place. The initial upper and lower bounds for the estimation of its stiffness coefficients were set as in Table 14:

Table 14 *Lower and upper bound for the stiffness estimation of the 3 DOF mechanism (following* [135]*).*

Stiffness	Lower bound	Upper bound
k_1 (in N/m)	0.1	100.0
k_2 (in N/m)	10.0	2500.0
k_3 (in N/m)	1.0	300.0

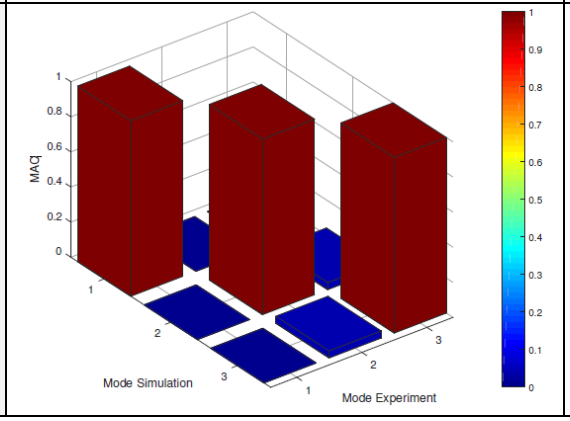
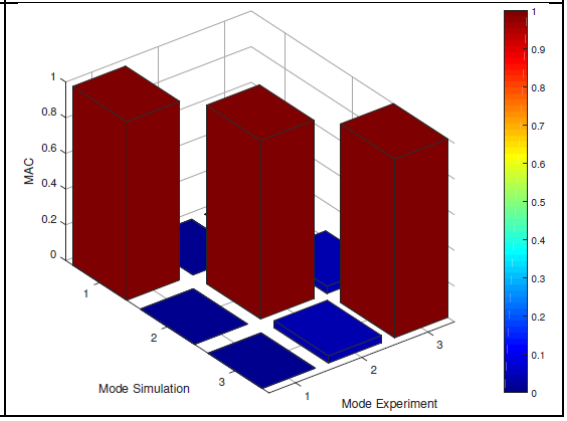
Its experimentally determined natural frequencies and stiffness coefficients, shown in Table 15, served as termination criterion for the parametrization.

Table 15 *Natural frequencies and stiffness coefficient of the 3 DOF mechanism (following* [135]*).*

Natural frequency			Stiffness coefficient		
ω_1	ω_2	ω_3	k_1	k_2	k_3
8.9 Hz	33.8 Hz	407.8 Hz	74.0 N/m	1094.1 N/m	165.2 N/m

The parameter identification was performed employing both, gradient-based and evolutionary algorithm. The results from each of the parameter identifications are shown in Table 16.

Table 16 *Comparison between the gradient-based algorithm and the evolutionary algorithm (following* [135]*).*

	Identification with gradient-based algorithm	**Identification with evolutionary algorithm**
ω_1	8.8 Hz	8.8 Hz
ω_2	33.8 Hz	34.1 Hz
ω_3	407.9 Hz	407.5 Hz
k_1	74.0 N/m	70.0 N/m
k_2	1094.1 N/m	1177.2 N/m
k_3	165.0 N/m	164.8 N/m
MAC		

Employing the gradient-based algorithm (nl2sol), the largest discrepancy obtained for the natural frecuencies and for the stiffness coefficients was 1.1% and 0.12%, respectively, considering them as highly accurate. The identification with the evolutionary algorithm has provided a maximal discrepancy of 0.6% for the natural frequencies and 7.6% for the stiffness coefficients.

The better results exhibited by the gradient-based method is primarily due to the satisfactory definition of the upper and lower boundaries for the computation. Nevertheless,

accurate outcomes would not have been easily reached, if these initial boundary limits had not be predicted beforehand. For such cases, an initial parametrization by means of the evolutionary algorithm is suggested to be performed firstly to estimate the interval of the parameter boundaries that can be employed as upper and lower limits for a second parameter identification using the gradient-based algorithm.

Accordingly to observations, the parametrization algorithm is further developed with:

- A combination of global and local optimization methods can be implemented for searching for the optimal parameter set. The evolutionary algorithm delivers several parameter sets with good global optimum. Based on these parameter sets, the correlation rate is increased with the local gradient-based method.
- Implementing weights and penalty functions to focus on certain reference values for a better correlation, allowing specific parameters to strongly dominate at the expense of the others. Additionally, a penalty function acts in the parametrization algorithm as soon as the requested constraints are violated.
- Parametrization of stiffness and damping coefficients separately. The restricted knowledge about damping parameters, mostly present as friction, makes its estimation difficult. Due to the fact that the damping coefficients usually have less influence on the modal parameters than the stiffness, the parametrization algorithm only considers the stiffness parameters in a first stage. However, by omitting the damping parameters, the MBS models provide purely real eigenmodes. For an adequate correlation between the EMA and the MBS results, it is necessary that both, the experimental and simulated natural frequencies and modes, are also purely real. To achieve this, the method of Fuellekrug [136] for mode transformation converts complex eigenmodes to real eigenmodes. Then, in a second stage, only the damping parameters are adjusted and the already computed stiffness remains unchanged. An outstanding problem during the transformation of complex eigenvectors is the condition that the amount of accelerometers should be equal to the amount of mode shapes, but usually the amount of accelerometers used during the EMAs is much higher than the amount of the determined mode shapes. By means of a modal truncation [137], the eigenvectors need to be truncated in order not to have to reduce the amount of measurement points.

The parametrization algorithm that includes the listed considerations was tested using the simple oscillating system shown in Figure 48.

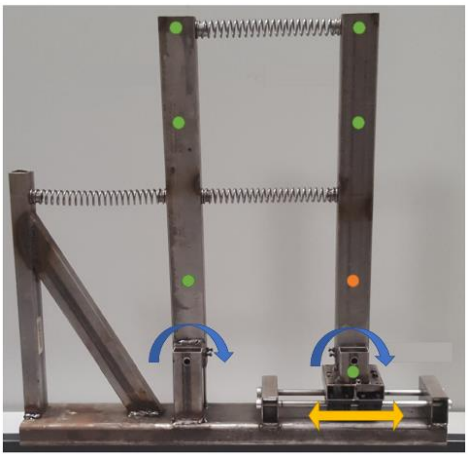
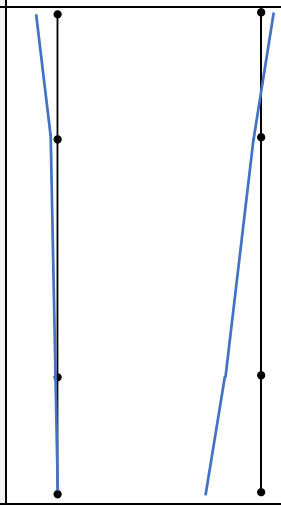
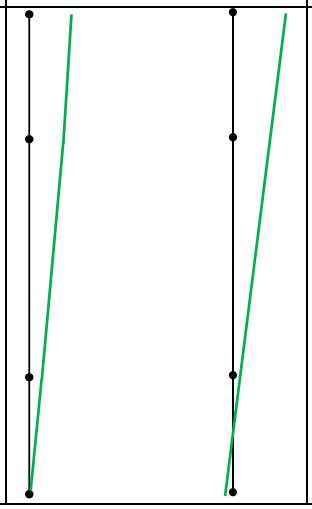
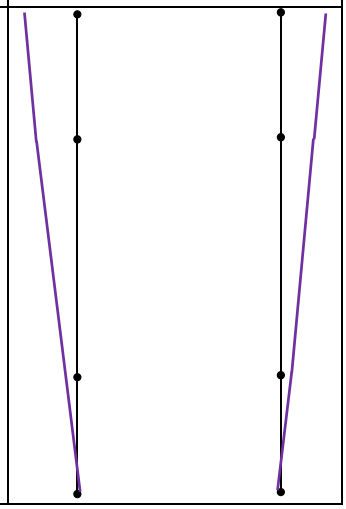


Figure 48 *Testing system for further development* [131].

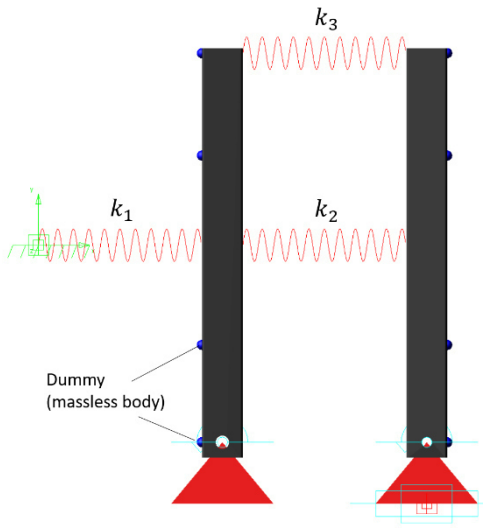
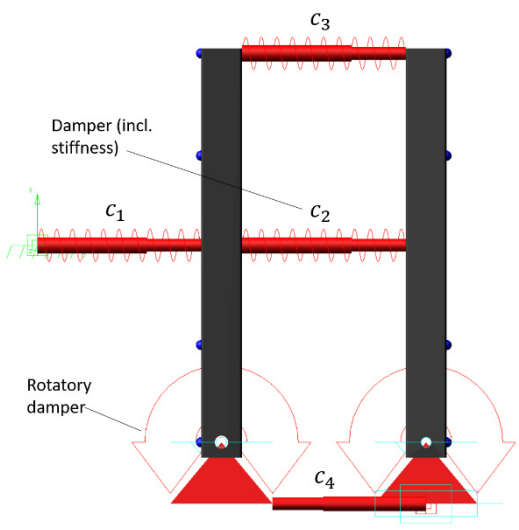
Its inherent modal parameters obtained by means of EMAs are presented in Table 17.

Table 17 *Modal parameters of the testing system acquired by EMAs (following* [131]*).*

	1st mode	**2nd mode**	**3rd mode**
Natural frequency	4.44 Hz	6.58 Hz	29.41 Hz
Damping ratio	4.00 %	1.10 %	0.36 %
Mode shape			

The MBS model of this system comprised the eight massless blue dummy bodies shown in Table 18, which were arranged accordingly to the real measuring points used during the EMA.

Table 18 *MBS model of the testing system* [131].

Undamped system	Damped system
	

At the beginning of the parametrization, the system was considered as undamped, taking into account only its three spring stiffness. With the help of the evolutionary algorithm, the limit value 2100 N/m was estimated for all spring's boundary constraints, employing it for the second attempt via gradient-based method.

The determined parameter set for its stiffness coefficients, its natural frequencies and the MAC correlation of all its eigenmodes are illustrated in Table 19.

Table 19 *Modal parameters obtained by the enhanced parametrization algorithm (following* [131]*).*

	Stiffness (N/m)		
	k_1=2778.4	k_2=2948.1	k=2389.0
	Natural frequency (Hz)		
	1st mode	2nd mode	3rd mode
EMA	4.4	6.6	29.4
MBS	4.4	6.6	29.4
MAC	1.0	1.0	1.0

Consequently, its three damping parameters were estimated. Since the damping parameters of the springs were completely unknown, their boundary limits were chosen relatively amply. The result from this parametrization is displayed in Table 20.

Table 20 *Damping coefficientes obtained during second run (following* [131]*).*

	Damping coefficients	
	EMA	**MBS**
1st mode	4.0 %	4.0 %
2nd mode	1.1 %	1.1 %
3rd mode	0.4 %	0.4 %

In conclusion, the parametrization method generates applicable and realistic results, if the reference values from the real system are experimentally accurately obtained and if the modeled MBS system is capable of reproducing the required same physical behavior.

The scripts implemented for the parametrization algorithm are presented in Annex A.1.

4.4 Development of stabilization strategies

In order to speed up the design and testing phase of the stabilization strategies, mechatronic co-simulations between the, to be developed, closed-loop controls (Matlab/Simulink) and the MBS model (built in MSC.ADAMS/View) are performed.

4.4.1 Mechatronic co-simulations for the stabilization strategies

To integrate the MBS model with the block diagrams (control systems) from Matlab/Simulink, the parameters to be controlled in the MBS model are defined as inputs and outputs variables. These variables read data from the MBS model during the simulation cycle, e.g., forces and torques generated in the robot manipulator joints, and also feed data back into the model, e.g., forces/moments applied by an actuator. This continuous data exchange enables a parallel, coupled simulation of mechanical and electrical/electronic components.

Figure 49 shows this principle: The closed-loop control of the external actuators in Matlab/Simulink sends the control signals to each of virtual actuators built in MBS model. Then, the MBS model returns measurements from virtual sensors to the feedback loops, such as force, angle, speed, acceleration, etc., at certain time intervals.

The representation of the emergency braking behavior of the mobile manipulator is indispensable for the good functioning of the proposed stabilization strategies. Therefore, a theoretical braking process of the mobile platform needs to be modeled as its trajectory in the MBS environment.

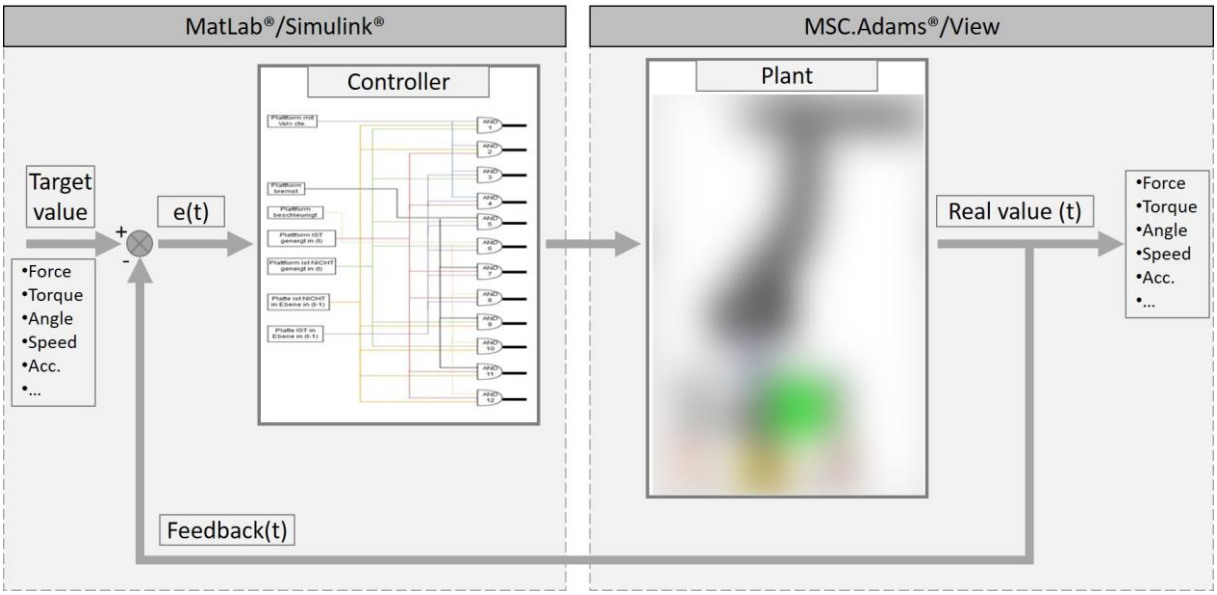


Figure 49 *Co-simulation between the developed stabilization strategies (in Matlab/Simulink) and the mobile manipulator as a multibody-system (in MSC.ADAMS/View).*

The procedure to perform the mechatronic co-simulations can be summarized as follows:

1. Define the inputs (from Matlab/Simulink) and outputs (sent to Matlab/Simulink) in the MBS model (MSC.ADAMS/View), such as forces, torques, position, speed, etc. The inputs defined in MSC.ADAMS/View should correspond to the outputs defined in Matlab/Simulink and reciprocal.
2. Export the transfer function from the MBS model of mobile manipulator (so-called ADAMS-Plant) and import it into Matlab/Simulink.
3. The simulation in Matlab/Simulink starts the interaction with the MBS model of the mobile manipulator by triggering the ADAMS solver. The inputs and outputs between both software platforms transfer the needed information during the simulation: The MBS model in MSC.ADAM/View acts according to the inputs sent by the control algorithm in Matlab/Simulink and this, in turn, reacts according to the feedback measured in the MBS model.
4. The evaluation of results can be done in both, Matlab/Simulink and MCS.ADAMS/View.

In order to guarantee reliable outcomes from the dynamic simulations triggered during the mechatronic co-simulations, a suitable solver able to find a dynamic solution for particular states of the model has to be chosen.

MBS software tools offer multiple numerical methods to solver the ordinary differential equations that describe the continuous state of dynamic systems. Two elementary types of solvers defined by the stiffness, as an efficiency concern, are available for dynamic simulations in ADAMS/Solver:

- "Stiff" algorithms use implicit backward differentiation methods (BDF) to solve differential and algebraic equations (so-called DAEs).
- "Non-Stiff" algorithms employ explicit formulations for solving common equations.

ADAMS/Solver provides four stiff integrators and a single non-stiff integrator. Following the statement presented by Shampine and Thompson [138], since the available non-stiff integrator did not converge on a solution during preliminary simulations, the problem may be stiff and stiff solvers should be employed, because they are designed for stiff solutions.

In order to select the suitable stiff solver in MSC.ADAMS/View that guarantees the computing state of the model during dynamic simulations, a solution stability analysis has to be carried out using the three stiff integrators WSTIFF, GSTIFF and BDF CONSTANT, whose properties are shortly summarized in Table 21.

Table 21 *Stiff integrators and its main properties* [139].

Integrator	Properties
GSTIFF	Standard integrator. Backward Differentiation Formulas. Uses fix coefficients for predictions and corrections.
WSTIFF	Backward Differentiation Formulas. Uses variable coefficients for predictions and corrections.
CONSTANT_BDF	Backward Differentiation Formulas. Uses fix coefficients for predictions and corrections. Its accuracy is defined by its maximal allowed increment. Each step is calculated by a non-local integration error. FORTRAN as programming language.

A deeper insight into the implicit solution methods of these stiff integrators can be find in [140] and [141].

Each integrator is characterized by indexes, which are defined as the number of times a differential-algebraic equation must be differentiated to get a system into an ordinary differential equations [142]:

- Index-3 (I3) is fast and ensures that the solution satisfies all modeled boundary conditions. However, integration errors are only monitored for movements of the system, and not for the calculation of velocities. Its convergence (Jacobian matrix) can be very limited when using small step sizes.
- Stabilized index[18]-2 (SI2), in conjunction with the integrators GSTIFF or CONSTANT BDF, takes into account the boundary conditions of equations of motion in the solution. Furthermore, the SI2 integrator monitors the integration error of velocity variables, leading to accurate results in the simulations. The convergence value remains stable at small step sizes, which increases the robustness of the corrector at small step sizes.
- Stabilized index[18]-1 (SI1), in conjunction with the integrators GSTIFF, WSTIFF or CONSTANT BDF, also takes into account the boundary conditions of equations of motion in the solution. In addition, as SI2, it monitors integration errors on Lagrange multipliers in the system. Its robustness is typically very similar to that of the SI2 index.

Based on solution stability analyses, a solver can be considered as accurate only if there are no differences between the simulation results from employing the same solver with the same index, by reducing its error tolerance. Therefore, a comparison of simulation outcomes using different error tolerances can determine if the numerical results comprise a good approximation for a true solution [143]. The implementation of a solution stability analysis for simulatios in the context of this work is outlined in Section 4.5.31.

[18] Reduction of the initial index-3, which is consider the most robust integrator but at the same time the most challeging for the numerical solution of the differential-algebraic equation [142].

Moreover, the mechatronic co-simulations of the stabilization strategies require that virtual actuators and their corresponding control are modeled in Matlab/Simulink. In order to implement a reliable and accurate control system for the stabilization strategies, a closed-loop control is needed.

The most relevant closed-loop control algorithms include cascade control, conditional feedback and state control [144]. The cascade control is the best suited for electric drive systems because their commonly complex systems to be controlled can be divided into smaller subsystems. It consists of a superordinate, slow outer control loop and, at least, one inner subordinate. The inner fast control loops promote the agile behavior of the control system, while the outer control loop only has to compensate residual dynamics, i.e. the inner control loops react against disturbances as if they were the only control loop presented.

Typically, the cascade control design starts with the inner control loop(s). Then, based on the behavior of this (these) inner control loop(s), the outer control loop is conceived.

The model of the DC motors (actuators) can be represented in the cascade control as black box models. They can be modeled with a transfer function of first-order lag (PT1) or second-order lag (PT2), with damping factor greater than one as in [145], with:

$$G_M(s) = \frac{K}{(T_1 \cdot s + 1)} \tag{77}$$

for first-order systems, and

$$G_M(s) = \frac{K}{(T_1 \cdot s + 1)(T_2 \cdot s + 1)} \tag{78}$$

for second-order systems. In Eq. *(77)* and Eq. *(78)*, K represents the process gain, T_1 and T_2 are the time constants for the first and second order lag, respectively.

The actuators (DC motors) should react optimally to dynamic changes in the system. PI-, PD- and PID-controllers can be implemented to approach a target value abruptly. The transfer function of a PID-controller, $G_C(s)$, can be expressed as follows:

$$G_C(s) = K_{PID} \cdot \left(1 + \frac{1}{T_N \cdot s} + T_V \cdot s\right) \tag{79}$$

where K_{PID} represents the controller gain, T_N is the reset time of the integral component of the controller and T_V is the controller derivative time. The selection of the suitable controller depends on the nature of the system dynamics.

4.4.1.1 Mechatronic co-simulations of linear drives

The orientation, the upper plate of the stabilization mechanism should adopt to compensate moments, is achieved through specific stroke lengths of the three linear drives proposed in Section 4.1.1. These required stroke lengths can be determined by its inverse kinematics [77].

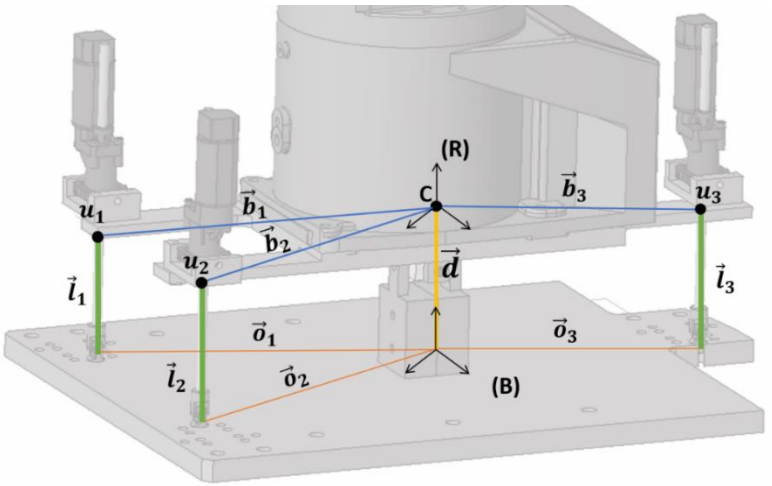


Figure 50 *Vectors for the estimation of the inverse kinematics (following* [146]*).*

For the calculation, the inverse kinematics approach for Stewart-platforms presented by Gattringer [146] is applied for the stabilization mechanism in Figure 50: point C symbolizes the location of the end effector (cardan joint); the constant that represents the distance between the two local coordinate systems R and B (one for each plate) is defined as $\vec{d}$ (in this case, this distance remains constant); vectors $\vec{o}_i$ are required to describe each of the cardan joints located on the lower plate and vectors $\vec{b}_i$ for the cardan joints located on the upper plate, relative to the local coordinate systems B and R, respectively; the green vector describes the stroke of the trapezoidal threaded rods, ${}_B\vec{l}_i$, which can be calculated via the vector chain as follows:

$$ {}_B\vec{l}_i \;=\; {}_B\vec{d}_C \;+\; A^T_{Cardan} \cdot \; {}_R\vec{b}_i \;-\; {}_B\vec{o}_i \qquad (80) $$

with the superscript B and R as origin of the local coordinate systems for the vectors and A^T_{Cardan} as the coordinate transformation via Cardan angles (see [146]).

The magnitude of the vector, being equivalent to the stroke length of the trapezoidal rod, is estimated with

$$ l_i = \sqrt{\left({}_B\vec{l}_i \right)^T \cdot \; {}_B\vec{l}_i} \; . \qquad (81) $$

Eq. *(81)* is implemented in Matlab/Simulink as inverse kinematic function. Both rotation angles of the upper plate of the stabilization mechanism are defined as its input parameters; further data such as distances between the coordinate systems, between the cardan joints and the local coordinate systems, etc., have also be custom-built into the function.

The resulting torque, velocity and displacement of the lineal drives, generated by the closed-loop control is implemented in the MBS model by means of variables operating on the respective actuators during the co-simulations. Forces and torques, angle and linear displacements as well as velocities and accelerations are measured/computed over time in the MBS model and sent as output signals to the closed-loop control of the actuation system. All these signals serve as real-time reference values that allow the control model to calculate the deviation and, thus, react in real-time against instabilities.

Additionally, each of the translational motion of the lineal drives is declared as input variable and the angle of the upper plate of the stabilization mechanism as output. Moreover, a jerk limitation in form of a low-pass filter with critical damping is required to avoid big

oscillations and to ensure a linear system behavior. Thus, the angle of the upper plate is pre-processed by the filter and, then, passed on to the inverse kinematics as a set point.

The entire concept is illustrated in Figure 51 and given in Annex A.2.

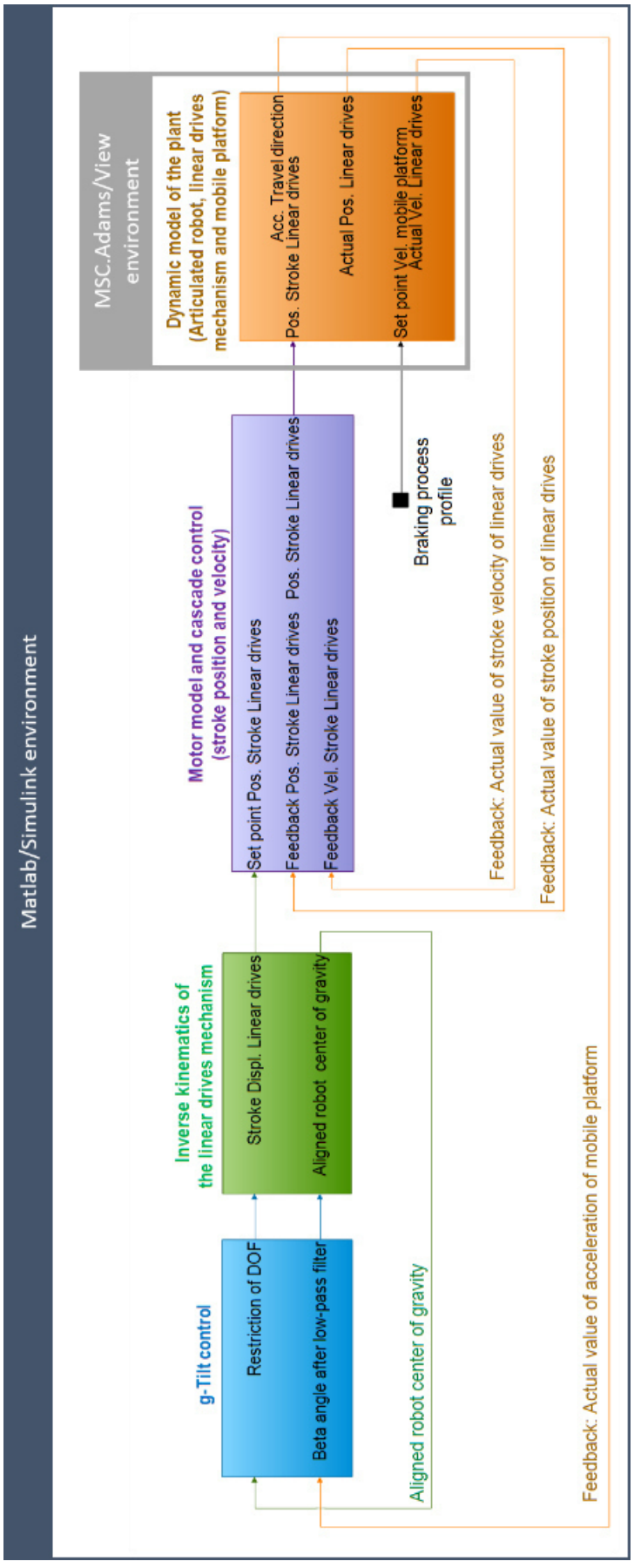


Figure 51 *Co-simulation of g-tilt control with the MBS of the mobile manipulator.*

4.4.1.2 Mechatronic co-simulations of gyroscope

The MBS model of the gyro mechanism must perform two different motions: the mass rotates around the vertical axis, and additionally, it is tilted around the horizontal axis (see **Figure 52** Rotary and tilting motors of the gyro mechanism.Figure 52).

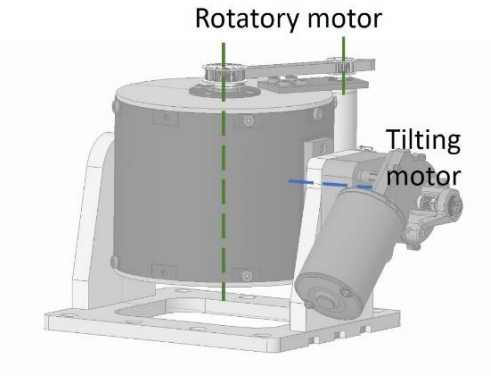


***Figure 52** Rotary and tilting motors of the gyro mechanism.*

For this purpose, a control algorithm for each motor has to be implemented in Matlab/Simulink. The mass rotational speed must remain constant during the simulations. This greatly simplifies the general control concept, since this rotational speed is set initially to the target value by a simple speed control and no changes are needed to be carried out afterwards. Therefore, only the tilting motion has to be controlled depending on the required deflection angle by means of a closed-loop position control.

Additionally, the control concept of the gyro mechanism detects accelerations and decelerations produced by the mobile platform, against which the gyroscope should automatically generate the compensation torque.

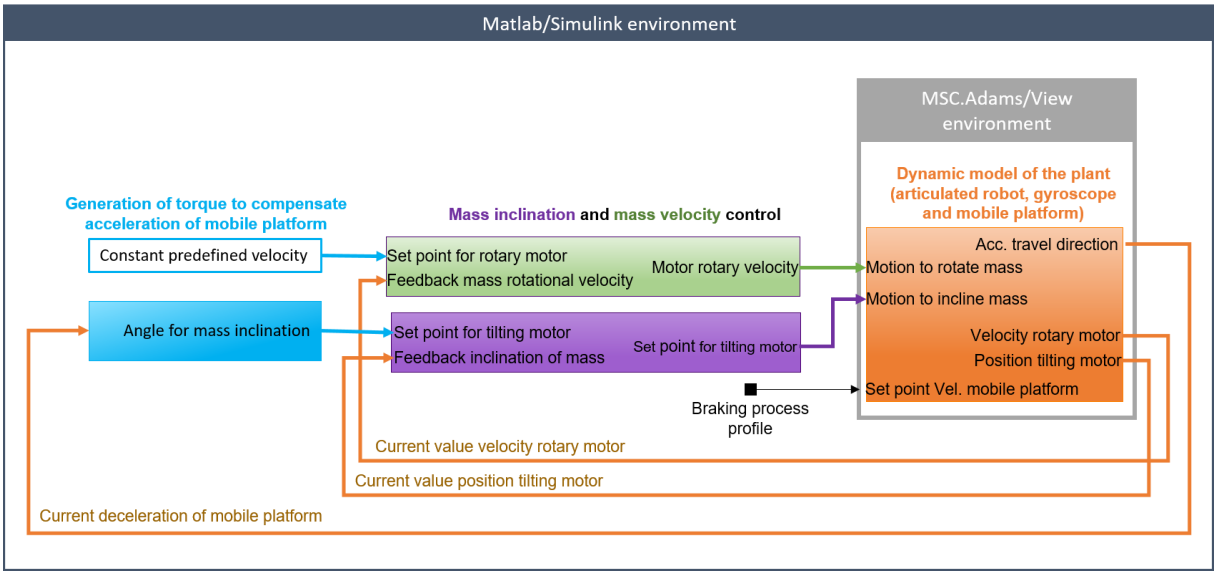


***Figure 53** Schema of the co-simulation for the gyroscope.*

Similar to the mechatronic co-simulations for the stabilization strategies in Section 4.4.1.1, while the motions of the gyro mechanism are implemented in the MBS model in MSC.ADAMS/View, the simulation of its control loop is carried out in Matlab/Simulink, establishing together the mechatronic co-simulation represented in Figure 53.

4.5 Implementation of stabilization strategies employing a testing system.

The following sections deal with the implementation of the stabilization strategies previously introduced (Section 4.4.1 and Section 4.4.2) employing a testing system, with the aim of demonstrating their effectiveness.

The adopted testing system consisted of the robot manipulator Mitsubishi RV-3AL mounted on the non-holonomic mobile platform Scitos X3 by MetraLabs, shown in Figure 54.

Figure 54 *Mitsubishi RV-3AL (left) and MetraLabs Scitos X3 (right) as testing system for further analyses.*

Although the robot manipulator does not represent the latest state-of-the-art in the robotic field, its weight and dimensions contribute to reproduce worst-cases scenarios, the approach should deal with. The robot manipulator is 660 mm high, 300 mm wide and weighs 60 kg.

Besides this, the Scitos X3 is 442.54 mm high, 470 mm wide and 710 mm long, with a mass of 58 kg and a maximum payload of 100 kg. The mobile platform is equipped with front and rear laser scanners that help to move safely in its environment: when an object or a person is detected in the field of one of the laser scanners, a speed reduction and a braking process is initiated to react in advance [147]. An abrupt braking process is performed only when the object is detected closer than 2 m radius.

The diagrams in Figure 55 displays the acceleration and braking process of the mobile platform according to the manufacturer. With an acceleration of 300 mm/s^2, its maximum speed (1000 mm/s) is reached in 3.67 s. When the mobile platform abruptly brakes, a maximum negative acceleration of 1000 mm/s^2 occurs during 0.16 s. A speed reduction, from 1000 mm/s to 200 mm/s, is automatically performed by the mobile platform when an object or person is detected within the outer warning field.

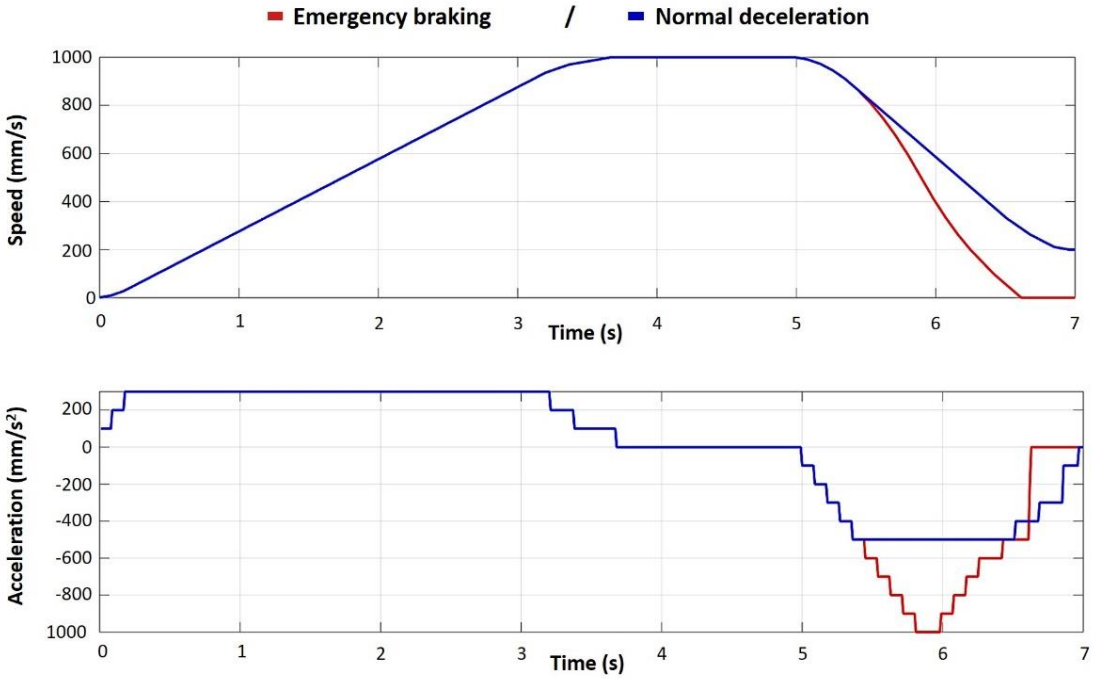


Figure 55 *Emergency brake and normal brake of the mobile platform.*

In order to detect a braking process based on the provided profile (Figure 55), an Inertial Measurement Unit (so-called IMU) was attached to the mobile platform. Figure 56 exemplifies the location of the IMU with respect to the mobile platform.

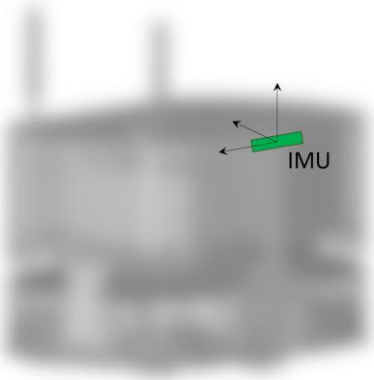


Figure 56 *Schematic sketch of the mobile platform and the IMU mounting location* [148].

Since the purpose of the IMU is to measure accelerations caused by the navigation of the mobile platform, the sensor can be arbitrarily placed on a horizontal surface plane of the mobile platform. However, it is recommended to mount the IMU in a surface where the least possible system vibrations can be perceived.

In the case of the testing system, the sensor axes were correlated with the vector that describes the platform motion and, additionally, follows this convention:

- "Roll axis", in which the mobile platform and the sensor both indicated a forward motion. Its measurements went from 0° to ±180°.
- "Pitch axis", perpendicular to the surface, which indicated the inertia of the system during linear movements. The measurements went from 0° to ±90° and, then, again to 0°.
- "Yaw axis", displayed the changes regarding the direction and orientation of the mobile manipulator. The measurements went from 0° to 360°.

The IMU included a 3D accelerometer, a 3D gyroscope and a 3D magnetometer [149] to measure:

- with the accelerometer, translational motion such as displacement, velocity, acceleration, jerk.
- with the gyroscope, rotational motion such as angular displacement, angular velocity, angular acceleration.
- with the magnetometer, the earth's magnetic field, which helped to estimate all the above mentioned parameters with a higher accuracy.

An evaluation of the range and accuracy of the measures acquired by the IMU was preliminary carried out: the mobile manipulator was tilted so that the mobile platform came into contact with the ground. It was observed that the sensor was able to detect all kind of alterations, even those caused by minuscule movements.

4.5.1 Experimental Modal Analyses of testing systems

The dynamic behavior of both, the adopted robot manipulator and the adopted mobile platform, were experimentally investigated in order to adjust and validate, as a next step, their corresponding MBS simulation models. This technique ensures a close-to-reality environment for the implementation of the proposed stabilization strategies by means of mechatronic co-simulations.

The EMAs presented in this section contribute to a deeper insight into the complete modeling process of robot manipulators and mobile platforms, lacking in the literature.

For the EMAs of both systems, the OR36 real-time analyzer (24-bit A/D converter integrated) by OROS was employed as acquisition instrument. The data processing was performed in Oros Modal[19] and NVGate multi-analyses[20] software.

Although the analysis was carried out in a frequency range from 0 Hz to 800 Hz, the most important frequencies for both systems were expected within a low frequency range, under 500 Hz.

The excitation and measuring devices for the experimental sets (see Table 22), i.e. the impulse hammer, tip and hammer head as well as accelerometers, were the same for all tests.

Table 22 *Excitation and measurement devices.*

Excitation stimulus	Measurement unit
Impulse hammer Dytran 5800B2 Impulse hammer Dytran 5805A	1D-Accelerometers Dytran 3225F1 3D-Accelerometers Dytran 3023A1

Their relevant parameters for the corresponding set up were chosen as shown in Table 23.

[19] https://www.oros.com/products/structural-dynamics/modal-analysis/
[20] https://www.oros.com/products/general-noise-and-vibration/software-platform-nvgate/

Table 23 *Configuration parameters for the experimental modal analyses.*

Parameter	Value (for robot manipulator and mobile platform sets)
Measured Response Frequency range	800 Hz
Impulse hammers – Tip	Soft impact tip
Impulse hammer (Dytran 5800B2 with 0.22 lb head weight) – Sensitivity, ±10%	100 mV/lbf
Impulse hammer (Dytran 5805A with 1 lb head weight) – Sensitivity, ±10%	1.0 mV/lbf
Accelerometer (Dytran 3225F1) – Sensitivity, ±10%	10 mV/G
Accelerometer (Dytran 3023A1) – Sensitivity,-10/+15%	10 mV/G

Aspects considered in this study, such as geometry, amount of measuring and excitation points as well as their corresponding arrangement in the structure of the robot manipulator and mobile platform, will help other researchers to experimentally analyze comparable mobile platforms and robot manipulators.

4.5.1.1 EMA of the mobile platform

First of all, the structure of the mobile platform was examined in order to select suitable excitation spots for the impulse hammer and measurement spots for the accelerometers, in such a way that all natural frequencies, natural modes and the natural damping will be covered by the experimental investigation.

Three excitation points (see green dots in Figure 57) were selected in order to guarantee for the application of stimulus in all coordinates of the Cartesian space. The 25 measuring points were placed on the rigid aluminum profiles and near to the screw joints, to ensure a good reproduction of the system oscillation and not to emulate the vibration of each single element.

All these spots built the basis for the geometry into the processing software OROS Modal and helped to graphically display the resulting mode shapes of the mobile platform (see Figure 58).

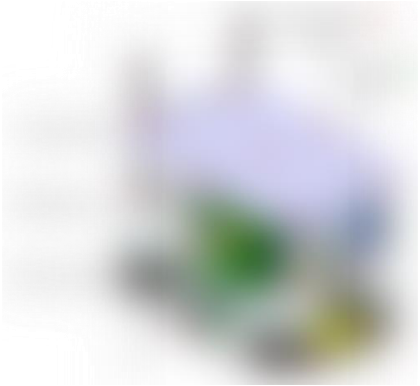


Figure 57 *CAD model of the mobile platform and its measurement points.*

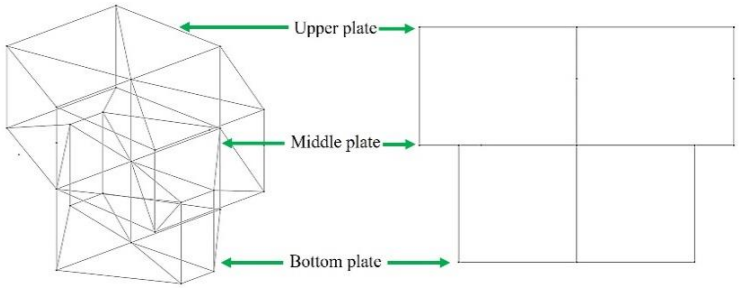


Figure 58 *Representation of the mobile platform in OROS Modal (based on the points in Figure 57).*

Each excitation stimuli was applied 5 times in the same spot, in the direction shown in Figure 59 (green arrows).

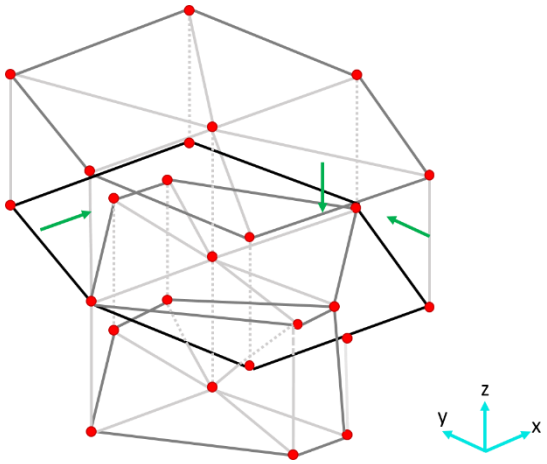
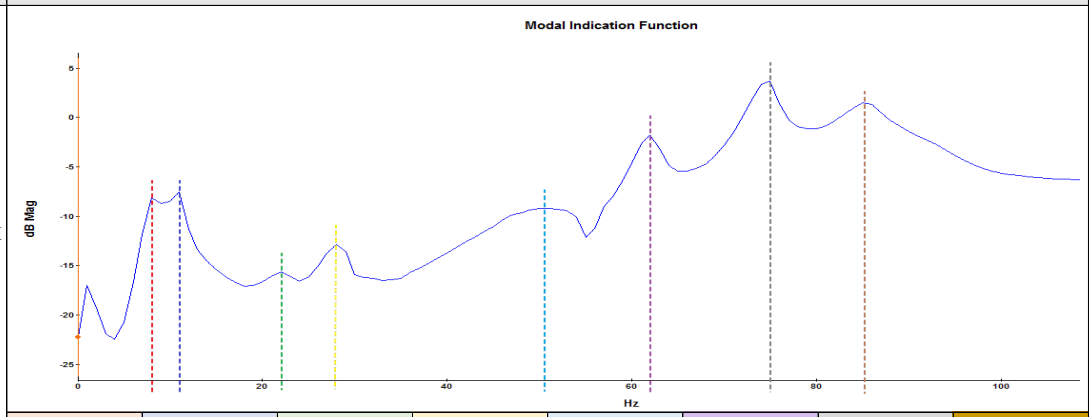
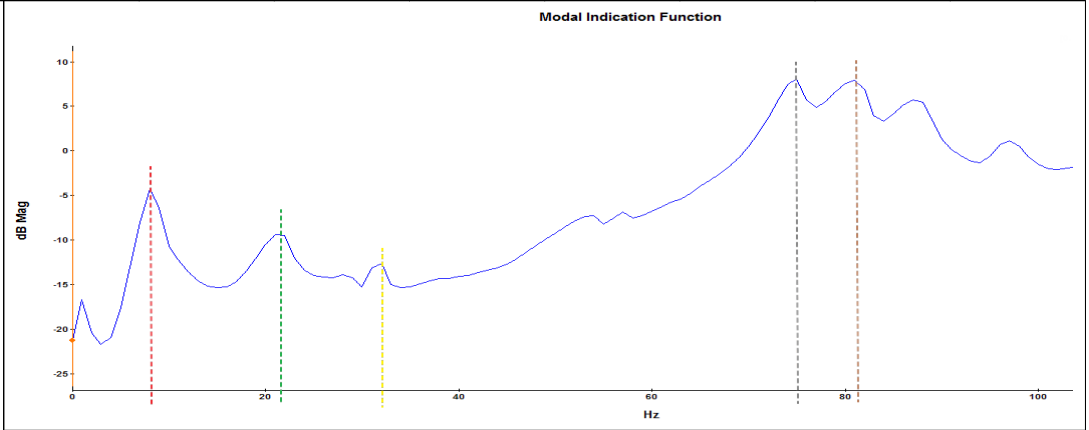
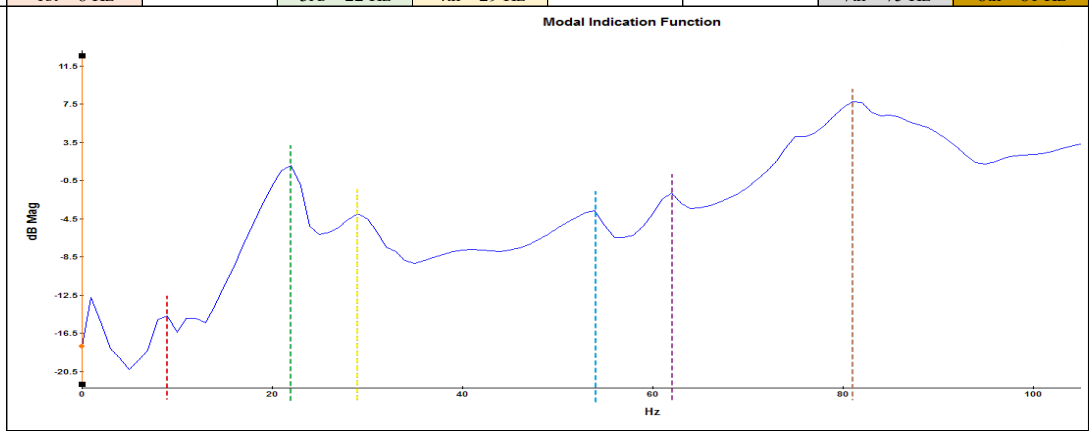


Figure 59 *Excitation (green) and measurement (red) spots for the mobile platform.*

Although the most important frequencies were expected within a low frequency range, two different sets were performed in order to check the repeatability of the EMAs and to delimit the frequency range in which the dynamic behavior of the mobile platform will be more precisely described: the first experimental set covered the frequency range from 0 Hz to 800 Hz to capture all natural frequencies and all eigenmodes as far as possible; the second experimental set examined the frequency range from 0 Hz to 100 Hz, to accurately identify the first fundamental natural frequencies and mode shapes.

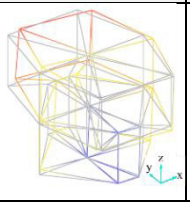
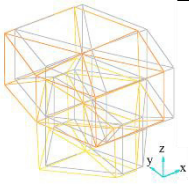
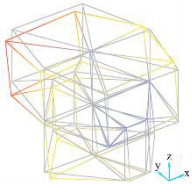
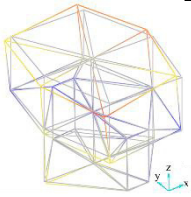
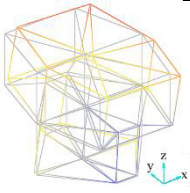
The results of the EMAs for the mobile platform are shown in Table 24. An overlapping method [150] allowed to identify its first eight natural frequencies under 100 Hz: The first, third and fourth natural frequencies (7 Hz, 21 Hz and 28 Hz) appeared in all tests. Following the overlapping method, the remaining values were obtained at 10 Hz, 54 Hz, 62 Hz, 75 Hz and 81 Hz, which were visible in no more than two different directions of excitation.

Table 24 *Identification of the Scitos X3 natural frequencies by means of overlaping method.*

Position of excitation	Identified natural frequencies
IH #1 Direction +X	 1st ≈ 8 Hz | 2nd ≈ 10 Hz | 3rd ≈ 21 Hz | 4th ≈ 28 Hz | 5th ≈ 52 Hz | 6th ≈ 62 Hz | 7th ≈ 75 Hz | 8th ≈ 85 Hz
IH #2 Direction +Y	 1st ≈ 8 Hz | 3rd ≈ 22 Hz | 4th ≈ 29 Hz | 7th ≈ 75 Hz | 8th ≈ 81 Hz
IH #3 Direction -Z	 1st ≈ 8 Hz | 3rd ≈ 22 Hz | 4th ≈ 29 Hz | 5th ≈ 54 Hz | 6th ≈ 62 Hz | 8th ≈ 81 Hz

The mode shapes corresponding to the first five natural frequencies are graphically represented in the Table 25.

Table 25 *Mode shapes and natural frequencies of the mobile platform obtained from the EMAs.*

Natural freq.	**ω_1=7.8 Hz**	**ω_2=10.3 Hz**	**ω_3=21.4 Hz**	**ω_4=28.0 Hz**	**ω_5=54.0 Hz**
Mobile platform mode shapes					

The grey lines in the geometry represent the initial state of the structure, the red lines the maximum amplitude attained, the yellow lines the proportionately smaller amplitudes and the blue lines the smallest one.

Experimental set no. 1 and experimental set no. 2 showed the same result for the frequency range up to 100 Hz: the first mode shape (7.8 Hz) from Table 25 consisted in a rotation of the platform about the Z-axis (yaw). The second mode shape (10.3 Hz) showed a translational movement in X-direction. The third one (21.4 Hz) exhibited a petty rotation around the Y-axis (roll) and another around the X-axis (pitch). The fourth and fifth mode shapes (28 Hz and 54 Hz) displayed a slightly tilting around the Y-axis (roll) and the X-axis (pitch), respectively. All axes are related to the light blue coordinate system.

4.5.1.2 EMA of robot manipulator

A similar procedure was performed for the robot manipulator with regard to the described analyses of the mobile platform.

The natural frequencies and mode shapes of the robot manipulator were determined for three different positions: home, fully extended arm in upright configuration and fully extended arm in horizontal configuration; all of them with and without energized motors. The described experimental set is illustrated in Figure 60.

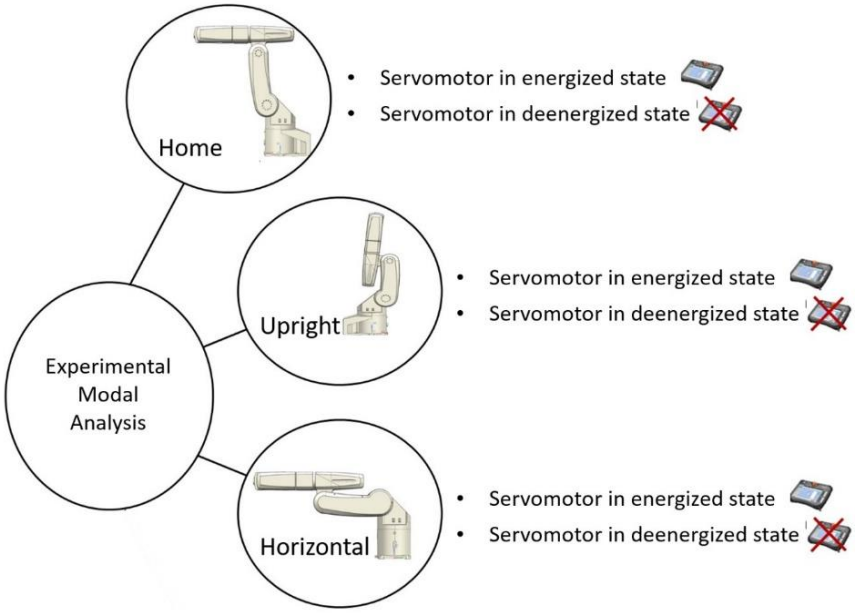


Figure 60 *Scheme of the performed experimental modal analyses for the robot manipulator.*

The results for all three different positions were comparable to each other. For this reason and for practical purposes, only the EMA of the robot manipulator carried out in Home configuration is presented below.

During the EMAs, two individual sets were performed: The first experimental set covered a frequency range from 0 Hz to 800 Hz, in order to capture all natural frequencies and, especially, all eigenmodes as far as possible. The second experimental set examined the frequency range from 0 Hz to 100 Hz to accurately classify these first fundamental natural frequencies and mode shapes.

Similar to the mobile platform, the excitation and measurement spots were chosen so that an appropriate identification of the modal parameters took place (see red and green dots in Figure 61). The simplification of the robot manipulator geometry in OROS Modal, shown in Figure 62, resulted in 3 excitation points and 20 measurement points.

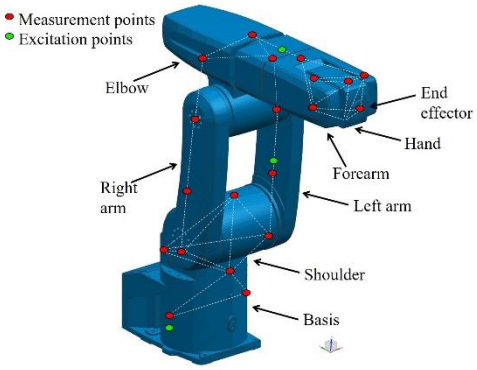


Figure 61 *CAD model of robot manipulator and its measurement points* [151].

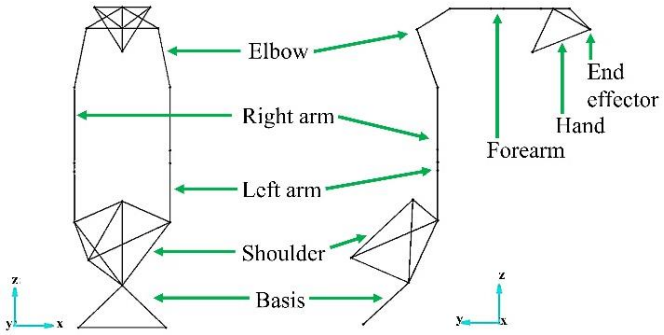


Figure 62 *Representation of the main parts of the robot manipulator in OROS Modal 2* [151].

Each excitation stimuli was applied 5 times in the same spot, in the direction shown in Figure 63 (green arrows).

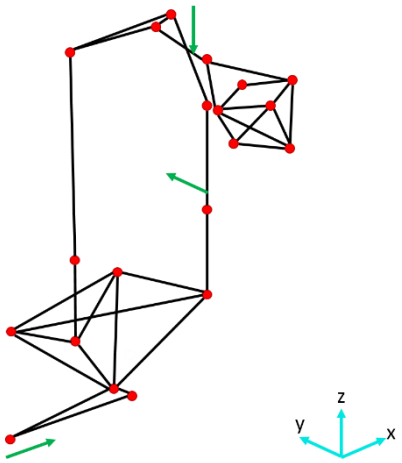


Figure 63 *Excitation (green) and measurement (red) spots for the robot manipulator.*

Table 26 presents the results obtained from the EMAs of the robot manipulator.

From Table 27, it can be observed that the first three natural frequencies (11 Hz, 21 Hz and 28 Hz) were found within all directions of excitation. Moreover, the natural frequency at 72 Hz and 87 Hz were only found in two separately excitation points.

Table 26 *First natural frequencies of the robot manipulator obtained from the EMAs.*

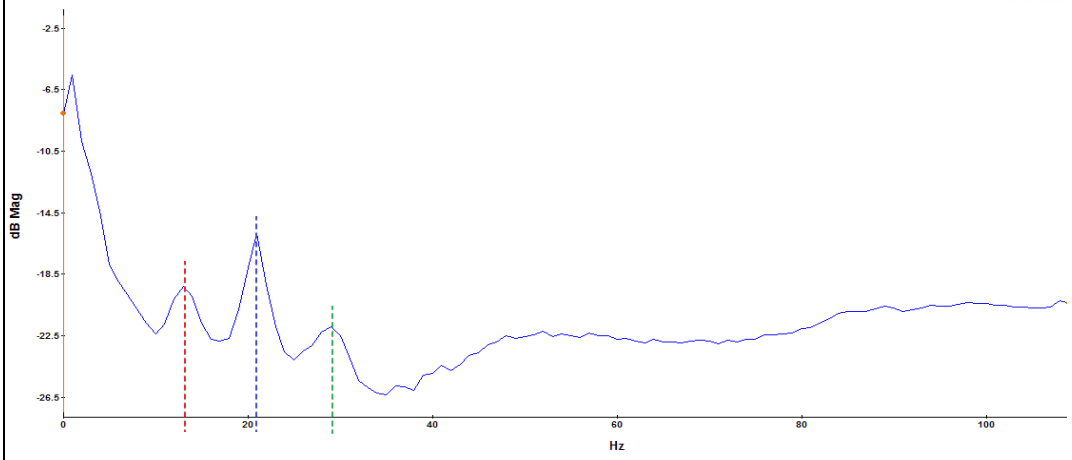
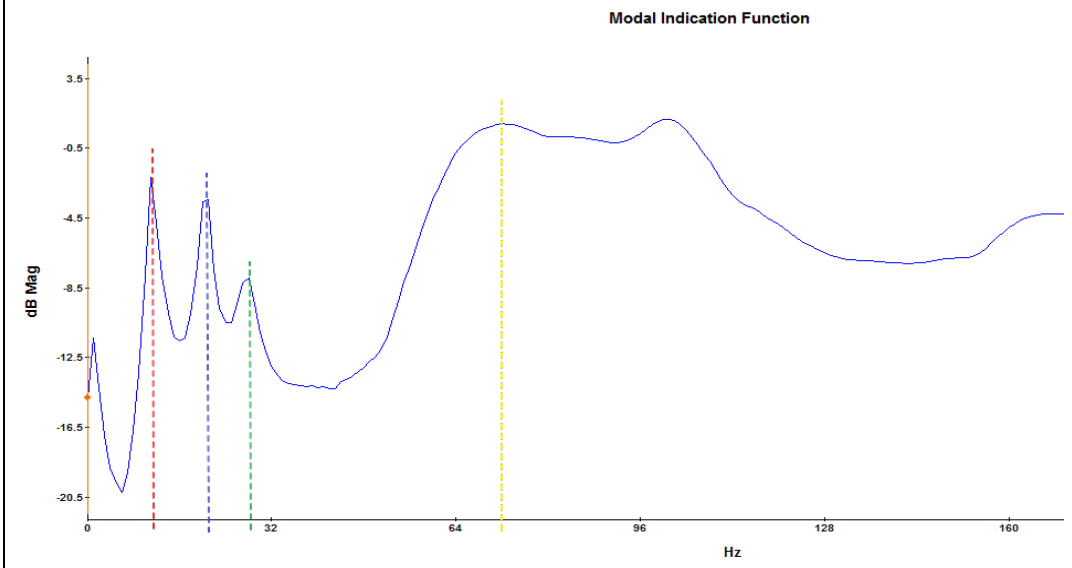
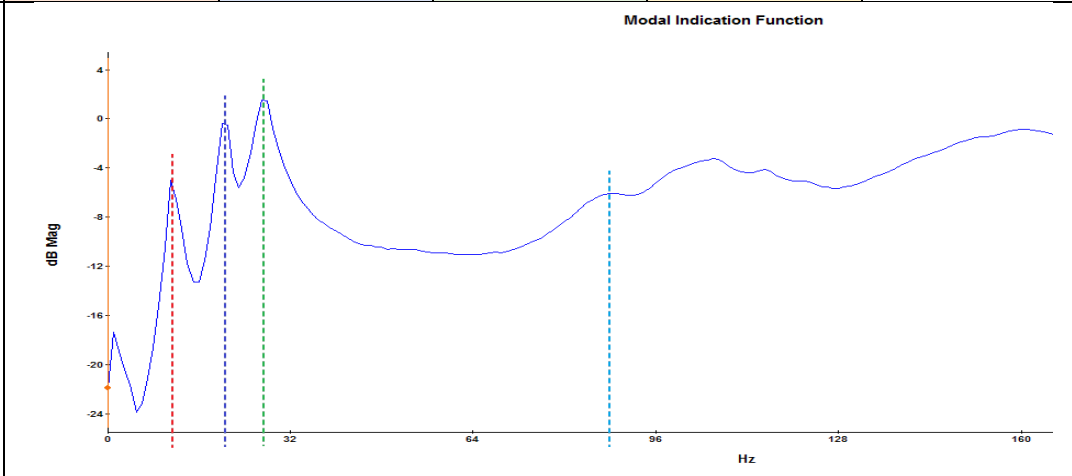
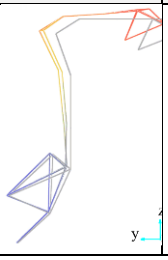
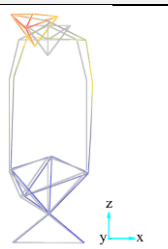
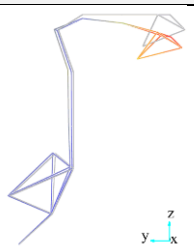
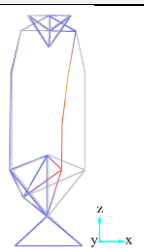
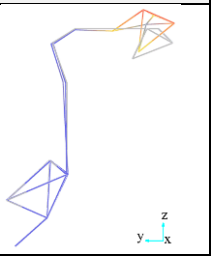
Position of excitation	Identified natural frequencies				
IH #1 Direction +X		1st ≈ 12 Hz	2nd ≈ 21 Hz	3rd ≈ 29 Hz		
IH #2 Direction +Y		1st ≈ 11 Hz	2nd ≈ 21 Hz	3rd ≈ 28 Hz	4th ≈ 72 Hz	
IH #3 Direction -Z		1st ≈ 11 Hz	2nd ≈ 21 Hz	3rd ≈ 27 Hz		5th ≈ 87 Hz

Table 27 *Mode shapes of the robot manipulator obtained from the EMAs.*

Natural frequencies and mode shapes	**ω_1=11.0 Hz**	**ω_2=21.0 Hz**	**ω_3=28.0 Hz**	**ω_4=72 Hz**	**ω_5=87 Hz**
EMA Robot manipulator					

Closer inspection of Table 27 shows that the first mode shape (at 11 Hz) was represented by a pitching movement (around the X-axis) generated by the elbow, the forearm and the end effector; the second and third mode shapes (21 Hz and 28 Hz) affected only the lightest four components, causing a slightly rolling movement around the Z-axis and a rotation around the X-axis (pitching) in the elbow, forearm, hand and end effector; the fourth mode shape (72 Hz) produced a deflection on the left arm; and finally, the fifth modal shape (87 Hz) also exhibited a slightly pitching movement (rotation around X-axis) but just starting in the hand and continuing with the end effector.

The next step was to examine the influence of the robot manipulator's electric drives and their control on the previously acquired modal parameters. Typically, self-locking transmission gears are located between the electric drives and the links of a robot manipulator. These gears could induce additional vibrations that have not been identified during the last measurement. For this reason, further sets of EMAs were performed with the robot manipulator brakes released and compared with the outcomes of the previous EMAs, which were performed with the motors switched off. Table 28 summarizes the first natural frequencies of the robot manipulator in home position with the motors switched on and off.

Table 28 *Natural frequencies of the robot manipulator in home position with its servomotor's brakes released and enganged.*

Home position				
Set	**Motor on/off**	**Natural frequency (Hz)**		
		ω_1	ω_2	ω_3
1		11	21	29
		11	21	29
2		13	21	28
		13	21	28
3		11	21	28
		11	21	28
4		11	21	28
		11	21	28

No significant changes of the robot manipulator system's dynamic behavior were observed between the two scenarios. It is therefore deduced that the state of the servomotors has no

influence on the oscillation behavior of the robot manipulator due to the self-locking gears installed in the transmissions.

4.5.2 Modeling the testing system as MBS

The MBS simulation model of the mobile platform and the robot manipulator should reproduce the essential dynamics of the real system. The results of their EMAs, in terms of natural frequencies and mode shapes, allow a reliable modeling procedure for the mobile manipulator as multibody-system.

Complementary to the EMAs exemplified previously, all the aspects considered in the following section will help other researchers to model comparable mobile platforms and robot manipulators as MBS.

4.5.2.1 MBS of mobile platform

The modeling process of the MBS of the mobile platform was based on the analysis of its system's dynamic behavior according to the information obtained from its EMAs.

First, the inertial parameters of the components that constitute its structure were estimated with all the information about geometries and materials contained in its 3D CAD model. Then, this 3D volume model was exported into MSC.Adams/View.

The most significant assembly for the modeling was the middle central profile shown in Figure 64, on which the drive wheels and the engines are suspended via an upper swing arm, a damper and a lower articulated strut. Due to the DOFs emerging by the connections of the individual elements, this assembly is largely responsible for the oscillation behavior in the lower frequency range of the entire mobile platform.

Figure 64 *Drive wheels and engines assembly of the mobile platform.*

Figure 65 presents a clear overview of the kinematics of the individual elements of the wheel suspension and the associated elements of the mobile platform.

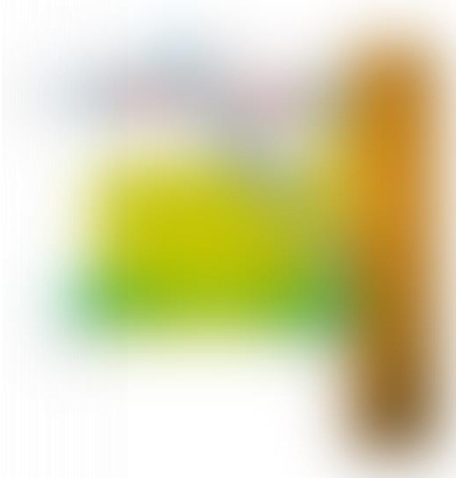


Figure 65 *Wheel suspension mechanism of the mobile platform.*

The MBS model contained low level of detail to get good computing performance, without affecting the reproduction of the real dynamic behavior by the exclusion of structural elements. The motors and their holders, the wheels and bearings holder as well as the individual components of the swing arms, the drive shafts and struts were combined into main rigid bodies. The remaining joint elements were classified based on their kinematic (geometrical relation) and their physical force properties (e.g. torsion springs) acting between two or more links [152].

Between the cylinder and the piston of the dampers, a linear spring that allowed motion along the center axis of the damper was defined. To constrain the forces affecting the main bodies, virtual massless elastic joints were implemented. Figure 66 illustrated this scheme for one of the two identical wheel suspension mechanisms of the mobile plaftorm.

Figure 66 *Graphical topology of the implemented joint elements for the wheel suspension mechanism of the mobile platform.*

With help of bushings and springs, all translational and rotational DOFs were described by means of the stiffness and damping assignments shown in Table 29 for their translational coefficients and in Table 30 for their corresponding rotational coefficients.

Table 29 *Adjusted translational stiffness and damping coefficients for the mobile platform.*

Joint elements	Translational stiffness coefficients (N/mm)			Translational damping coefficients (N·s/mm)		
	x	y	z	x	y	z
Bushing #1 Swing arm < > Al-profile	30.0	10000.0	400.0	0.01	0.01	0.01
Bushing #2 Swing arm < > Al-profile	40.0	10000.0	400.0	0.01	0.01	0.01
Bushing #3 Swing arm < > Al-profile	30.0	10000.0	400.0	0.01	0.01	0.01
Bushing #4 Swing arm < > Al-profile	40.0	10000.0	400.0	0.01	0.01	0.01
Bushing #1 Swing arm < > DC Motor	150.0	10000.0	400.0	0.01	0.01	0.01
Bushing #2 Swing arm < > DC Motor	250.0	10000.0	400.0	0.01	0.01	0.01
Bushing #3 Swing arm < > DC Motor	250.0	10000.0	400.0	0.01	0.01	0.01
Bushing #4 Swing arm < > DC Motor	150.0	10000.0	400.0	0.01	0.01	0.01
Bushing #1 Rigid strut < > Al-profile	40.0	10000.0	400.0	0.01	0.01	0.01
Bushing #2 Rigid strut < > Al-profile	40.0	10000.0	400.0	0.01	0.01	0.01
Bushing #1 Rigid strut < > DC Motor	100.0	10000.0	400.0	0.01	0.01	0.01
Bushing #2 Rigid strut < > DC Motor	100.0	10000.0	400.0	0.01	0.01	0.01
Spring #1&2 Cylinder < > Piston	-	-	600.0	-	-	0.01
Bushing #1&2 Piston < > DC Motor	1000.0	1.0×10^{6}	600.0	0.01	0.01	0.01

Table 30 *Adjusted rotational stiffness and damping coefficients for the mobile platform.*

Joint elements	Rotational stiffness coefficients (N·mm/deg)			Rotational damping coefficients (N·mm·s/deg)		
	x	y	z	x	y	z
Bushing #1 Swing arm < > Al-profile	5.72×10^{5}	2864.8	14323.9	0.6	0.6	0.6
Bushing #2 Swing arm < > Al-profile	5.72×10^{5}	5729.6	14323.9	0.6	0.6	0.6
Bushing #3 Swing arm < > Al-profile	5.72×10^{5}	2864.8	14323.9	0.6	0.6	0.6
Bushing #4 Swing arm < > Al-profile	5.72×10^{5}	5729.6	14323.9	0.6	0.6	0.6
Bushing #1 Swing arm < > DC Motor	5.72×10^{5}	5729.6	14323.9	0.6	0.6	0.6
Bushing #2 Swing arm < > DC Motor	5.72×10^{5}	2864.8	14323.9	0.6	0.6	0.6
Bushing #3 Swing arm < > DC Motor	5.72×10^{5}	2864.8	14323.9	0.6	0.6	0.6
Bushing #4 Swing arm < > DC Motor	5.72×10^{5}	5729.6	14323.9	0.6	0.6	0.6

Table 30 (cont.) *Adjusted rotational stiffness and damping coefficients for the mobile platform.*

Joint elements	Rotational stiffness coefficients (N·mm/deg)			Rotational damping coefficients (N·mm·s/deg)		
	x	y	z	x	y	z
Bushing #1 Rigid strut < > Al-profile	5.72×10^5	2864.8	45836.6	0.6	0.6	0.6
Bushing #2 Rigid strut < > Al-profile	5.72×10^5	2864.8	45836.6	0.6	0.6	0.6
Bushing #1 Rigid strut < > DC Motor	5.72×10^5	5729.6	28647.9	0.6	0.6	0.6
Bushing #1 Rigid strut < > DC Motor	5.72×10^5	5729.6	28647.9	0.6	0.6	0.6
Spring #1&2 Cylinder < > Piston	-	-	28647.9	-	-	0.6
Bushing #1&2 Piston < > DC Motor	5.72×10^5	5729.6	28647.9	0.6	0.6	0.6

A representation of the model of the mobile manipulator as MBS in MSC.Adams/View is shown in Figure 67.

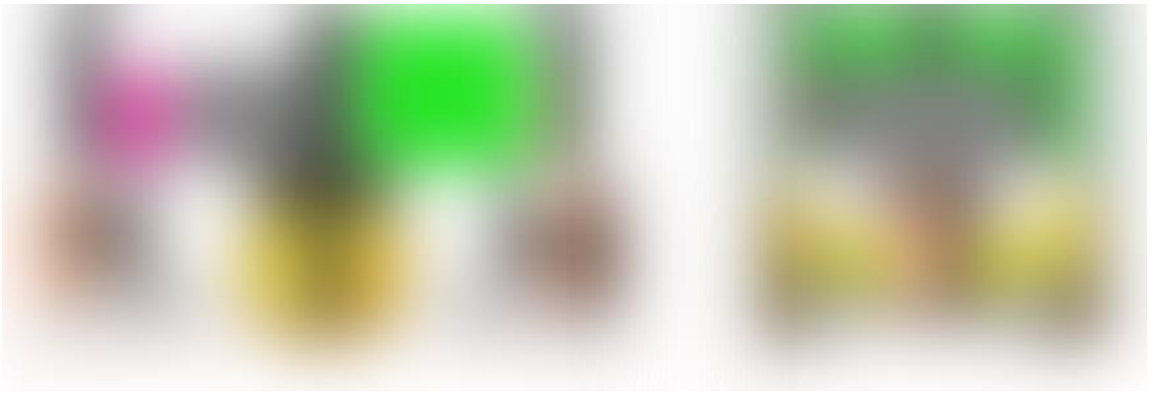


Figure 67 *MBS model of mobile platform.*

The parameters of the implemented ground contact forces were empirically adjusted. Particular attention was paid to the stiffness and damping values, since a performed sensitivity analysis exhibited that these values affected the most the impact behavior between the ground and the wheels of the mobile platform. Beginning with the stiffness and damping going towards $+\infty$ [153], their values were iteratively decreased to guarantee that no convergence errors and no large vibrations occurred during the simulations. All remaining parameters were empirically tuned based on the values suggested by the simulation software, as displayed in Table 31 [154].

Table 31 *Settings for the Coulomb's friction contact constraints between the wheels and the ground.*

Parameter for contact constraint	Value
Contact type	Point to plane
Normal force	Impact
Stiffness	1.0×10^4 N/mm
Force exponent	1.5
Damping	500.0 N·s/mm
Penetration depth	1.0×10^{-2} mm
Friction force	Coulomb
Coulomb friction	On
Static coefficient	0.76
Dynamic coefficient	0.8
Stiction transition vel.	100.0 mm/s
Friction transition vel.	1000.0 mm/s

All along the iterative manual adjustment of the stiffness and damping coefficients of the bushings and springs implemented in the MBS model, the visualization of the mode shapes in the simulations served as helpful qualitative analysis to identify which specific joint element had to be modified and how much.

Table 32 shows the derived natural frequencies and mode shapes for the mobile platform. The red traces in the illustrations represent the mode shapes in an overstated manner.

Table 32 *Modal parameters of the mobile platform obtained from MBS simulations* [151].

Natural freq. and modal shapes	**ω_1=8.9 Hz**	**ω_2=13.0 Hz**	**ω_3=21.5 Hz**
MBS Mobile platform			

The graphic in Figure 68 presents the calculation of the MBS reliability related to the EMA, using the "three-sigma limit" method [155]. The natural frequencies from EMA are represented by the black solid line. It can be seen that the first modal frequency of the MBS (red solid line) was not kept within the calculated limits (black and green dashed lines) as opposed to the second and third values, which were in the tolerated area.

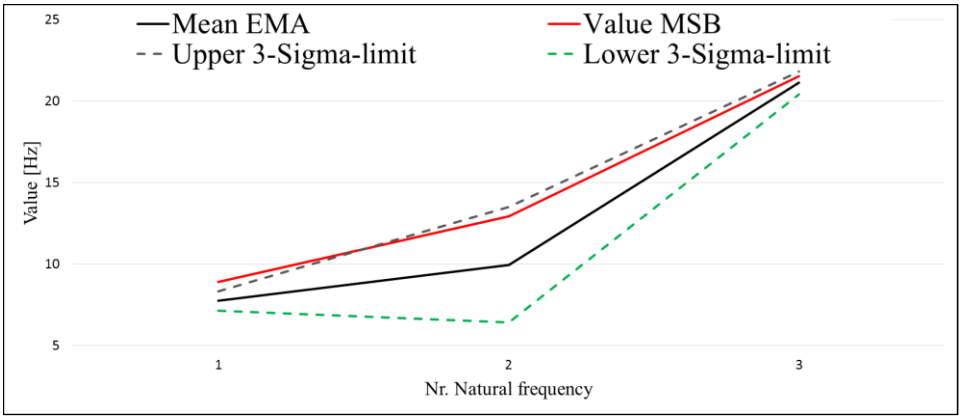


Figure 68 *Correlation between EMA- and MSB-mobile platform using 3-sigma limits* [151].

The deviation of the first natural frequency, about 15%, could have been caused by the model simplification procedure enforced during modeling the mobile platform. Another uncertainty existed with respect to the global damping parameters determined from the EMAs, which in practice cannot be directly applied in MSC.Adams/View. However, according to [156,157], the model can be considered as satisfactory for the purpose of simulating the braking process of the mobile manipulator, since it will be integrated in a closed-loop control system to enhance robustness against small disturbances in the process.

4.5.2.2 MBS of robot manipulator

The modeling process of the six-axis robot manipulator in MSC.Adams/View was very similar to the procedure presented for the mobile platform, but with a main difference: its available CAD model just contained information merely related to sizes and geometries, but not to aspects about materials, weight, center of mass or moment of inertia, which play a crucial role in the modeling process.

The bigger and heavier bodies of the robot manipulator mainly affect its dynamic behavior. For this reason, a material identification was carried out to estimate their relevant center of mass and moment of inertia and, thus, the density of the materials in the outer structure by performing some simple non-destructive tests. These material specifications were introduced into the 3D CAD model of the robot manipulator and were adjusted to match up the weight specified by the robot manufacturer. Then, this detailed model was exported into MSC.Adams/View.

Analogous to the mobile platform, the robot manipulator components with the higher mass and inertial moment were abstracted and merged into primary elements: Basis, shoulder, right arm, left arm, elbow, forearm, hand and end-effector. They were also modeled as rigid bodies for the same reason as the mobile platform: these main bodies did not present relevant deformations in comparison to their elastic joint elements. In the same way, each of these 8 basic elements were joined using three-dimensional massless bushings, at the place where in the real system massed gearboxes were located. The implemented bushings with respect to the main bodies of the robot manipulator are illustrated in red in Figure 69.

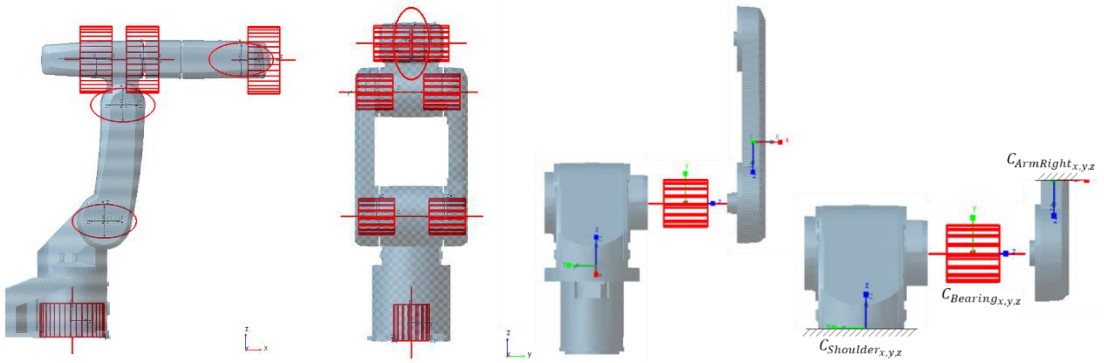


Figure 69 *6-DOF-limited bushings in the multibody-system simulation of robot manipulator.*

The inherent stiffness and damping parameters of these joining elements enable the dynamic oscillation of the robot manipulator. Likewise the mobile platform (see Section 4.5.2.1), the experimental outcomes from the EMA of the robot manipulator served as reference for the iterative adjustment of the stiffness and damping parameters in the MBS simulation model. The final values for each of the implemented bushings are presented in Table 33.

Table 33 *Adjusted stiffness and damping coefficientes for the bushings of the MBS robot manipulator.*

Parameter		**x-coordinate**	**y-coordinate**	**z-coordinate**
Bushing at "Base-Shoulder"				
Translational	Stiffness (N/mm)	2.4×10^{6}	2.4×10^{6}	2.5×10^{6}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	4.6×10^{9}	4.6×10^{8}	4.6×10^{8}
	Damping (N·mm·s/deg)	5.73	5.73	5.73
Bushing at "Shoulder-Upper arm left"				
Translational	Stiffness (N/mm)	2.1×10^{4}	2.0×10^{4}	2.5×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	5.0×10^{5}	9.0×10^{6}	5.0×10^{5}
	Damping (N·mm·s/deg)	5.73	5.73	5.73
Bushing at "Shoulder-Upper arm right"				
Translational	Stiffness (N/mm)	2.1×10^{4}	2.0×10^{4}	2.5×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	5.0×10^{5}	9.0×10^{6}	5.0×10^{5}
	Damping (N·mm·s/deg)	5.73	5.73	5.73
Bushing at "Upper arm left-Elbow"				
Translational	Stiffness (N/mm)	6.76×10^{3}	2.00×10^{4}	3.43×10^{3}
	Damping (N·mm·s)	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
Rotational	Stiffness (N·mm/deg)	4.50×10^{5}	1.00×10^{5}	1.50×10^{6}
	Damping (N·mm·s/deg)	5.73	5.73	5.73
Bushing at "Upper arm right-Elbow"				
Translational	Stiffness (N/mm)	6.8×10^{3}	2.0×10^{4}	3.43×10^{3}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	4.5×10^{5}	1.0×10^{5}	1.5×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at "Elbow-Forearm"				
Translational	Stiffness (N/mm)	2.3×10^{4}	1.3×10^{4}	1.2×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	2.9×10^{6}	5.7×10^{6}	5.2×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7

Table 33 (cont.) *Adjusted stiffness and damping coefficients for the bushings of the MBS robot manipulator.*

Bushing at "Forearm -Hand left"				
Translational	Stiffness (N/mm)	1.7×10^{4}	2.0×10^{4}	1.9×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	5729.6	5729.6	1.7×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at "Forearm -Hand right"				
Translational	Stiffness (N/mm)	1.7×10^{4}	2.0×10^{4}	1.9×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	5729.6	5729.6	1.7×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at "Hand-Finger"				
Translational	Stiffness (N/mm)	5.4×10^{3}	5.1×10^{3}	4.0×10^{3}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	1.2×10^{7}	5.7×10^{7}	1.7×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7

Table 34 displays with red traces the representation of the mode shapes. Higher frequency ranges were not considered, since only the oscillations where the main bodies move relative to each other are particularly relevant for the dynamic behavior of the robot manipulator in the mechatronic co-simulations.

Table 34 *Modal parameters of robot manipulator obtained from the MBS simulations* [151].

Natural freq.	ω_1=13.0 Hz	ω_2=22.5 Hz	ω_3=27.2 Hz	ω_4=68.3 Hz	ω_5=88.9 Hz
Mode shapes					

The MBS of the robot manipulator presented a good fitting of its natural frequencies. As can be seen from the graphic for "three-sigma limits" calculation [155] in Figure 70, the second natural frequency was slightly outside of the tolerance margin. However, the maximal deviation between this MBS and the EMA resulted about 10.6%, which is considered sufficient in this type of study [33,156].

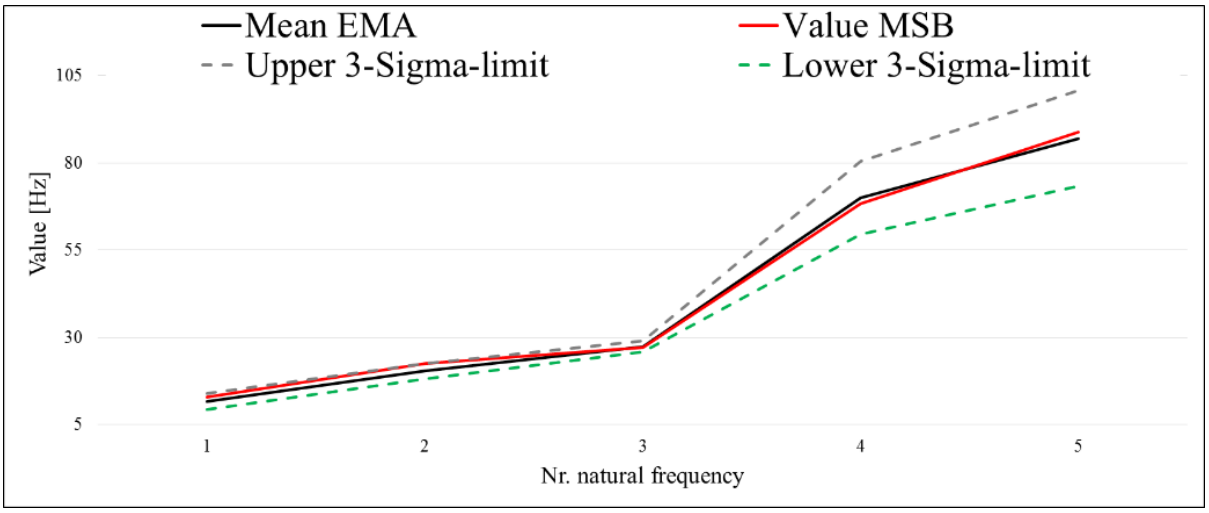


Figure 70 *Correlation between EMA- and MSB-robot manipulator using 3-sigma limits* [151].

The deviations between the MBS- and the EMA- results could be caused, on the one hand, by the model simplification carried out to decrease the complexity of the systems (e.g., rigid bodies were used to avoid large computational calculations). On the other hand, parametric uncertainties could be induced by lack of material homogeneity, tolerances in the geometry, unknown clearances between the robot manipulator elements, etc., having non-ideal real conditions for the MBS model.

It is important to emphasize that the premises for MBS modeling presented in this study were based on the nature of the dynamics of 6 DOF robot manipulators and mobile platforms with non-holonomic drive, both not exhibiting significant deformation of its mechanical structure. Table 35 provides an overview of these premises and their corresponding limitations for practical use in further systems.

Table 35 *Premises applicable purely to 6 DOF robot manipulators and mobile platforms with non-holonomic drive and without significant deformations.*

Adopted premise for modeling the robot manipulators and mobile platforms as MBS systems	Limitations/Remarks
Modeling the structure as rigid bodies.	Avoid if it is intended to study the deformation of the structure. In this case, the implementation of flexible bodies is crucial during modeling the system.
Merging multiple solid bodies to create a single-bodied part.	Use only if the motion of the main bodies relative to each other (and not its interaction) is of interest.
Merging bodies with different material properties.	By combining the bodies' density, center of gravity and moment of inertia into a single flexible body, local material-dependent deformations will not be reproduced.
Removing and replacing screws, screw nuts, washers and slot nuts by fix joints.	Avoid if the weight difference of these elements with respect to the rest of the bodies of the system is too small as themselves to affect the dynamic behavior of the system.

Table 35 (cont.) *Premises applicable purely to 6 DOF robot manipulators and mobile platforms with non-holonomic drive and without significant deformations.*

Adopted premise for modeling the robot manipulators and mobile platforms as MBS systems	Limitations/Remarks
Considering only natural frequencies and the mode shapes found in low frequency ranges.	Applicable only, if the oscillations generated by the interaction between the main bodies are particularly relevant for the dynamic behavior of robots, which is expected to be detected in the low frequency range. Higher frequency ranges have to be examined if it is intended to study the vibration of the structure.
Assuming that the stiffness of the robot manipulator bodies are significantly greater than the stiffness of its revolute joints.	Applicable only, if the structure does not present significant deformations or if the behavior of the revolute joint shows high friction indices.
Adjusting only the stiffness and damping coefficients for the friction contacts between the ground and the wheels.	Applicable only, if the impact behavior between the bodies or the evaluation of contact forces are irrelevant during the simulations.
Implementing the stiffness between two body centers of gravity using a spring-damper model.	This implies that the stiffness of the bearings itself is incorporated in the stiffness value of the joining elements. Avoid if it is proposed to analyze the action and reaction region of the structure where the joint is situated.

The use of these premises in systems, whose structure and/or dynamic behavior is not comparable with the robotic systems adopted in this work could lead to incorrect results and, thus, to a model that does not resemble the reality.

By contrast, the following criteria can be implemented in other different types of robot manipulators and mobile platforms:

- Interlink individual rigid bodies using physical connections (e.g. bushings) to build the system's kinematic chain.
- In order to constrain translational and rotational forces affecting the main bodies, virtually massless elastic joints with 6 DOF (e.g. bushings) can be implemented. By only adjusting the stiffness and damping coefficients of each of its DOF, the translational and rotational forces affecting the bodies can be reproduced in an accurate manner.
- If the mass of a joint element is already considered in the mass of the adjacent rigid bodies, its corresponding kinematic joint or physical connection in the MBS model can be considered as mass-free.
- In order to obtain accurate mode shapes in the simulation models, equip the MBS model with dummy massless bodies to represent the exact location of the accelerometers employed in the EMAs.

Further information about the general multibody approach for dynamic systems can be found in [124,158].

4.5.2.3 Optimization of MBS of the robot manipulator using the parametrization algorithm

Since the robot manipulator is the subsystem that will be driven by the stabilization mechanism and due to its high center of gravity, its dynamic behavior plays an important role during the development of the stabilization strategy. It must be ensured that the MBS model of the robot manipulator describes the real system more accurately, with a maximum deviation of about 5% between the natural frequencies and mode shapes obtained by means of EMAs and the MBS simulations.

Therefore, the parametrization algorithm introduced in Section 4.3.2.1 re-estimated the stiffness and damping parameters of the bushings modeled in the MBS. For this purpose, the dynamic behavior of the real robot manipulator had to be reanalyzed experimentally via EMA, but this time including the following aspects:

- The base of the robot manipulator was firmly screwed to a steel plate, which in turn was fixed to the ground.
- In order to better identify the robot manipulator's oscillation behavior, more one-dimensional accelerometers had to be attached to the structure; only in this way, all spatial directions could be registered.
- The system was excited at the forearm, upper arm and elbow in the frequency range up to 500 Hz. A too strong excitation could cause a non-linear system behavior [159].

For the following identification processes, only the first four natural frequencies and their corresponding eigenmodes were used as reference values, since the global eigenmodes from the EMAs (see Section 4.5.1.2) showed that the main oscillations were caused by the upper links. Therefore, only the four bushings located on the upper links were parametrized first. The forearm and elbow were merged to one rigid body, eliminating their associated bushings.

The MBS model was complemented with dummies, located where the accelerometers were attached to the individual bodies of the real system. The MBS model of the robot manipulator adapted for its adjustment employing the parametrization algorithm is displayed in Figure 71.

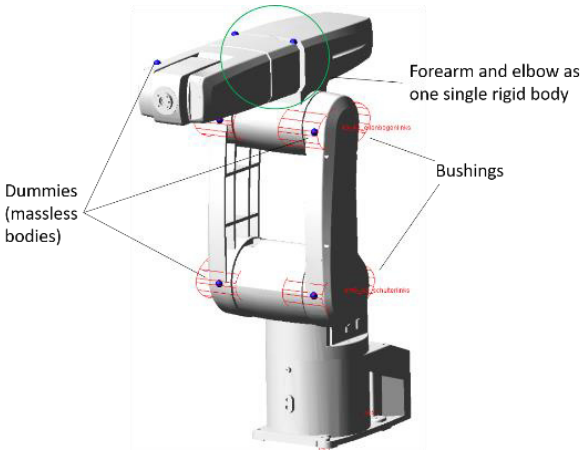


***Figure 71** MBS model for robot manipulator suited for the DAKOTA algorithm* [131].

The first parametrization was performed by means of the evolutionary algorithm, set with relatively wide outer boundaries. Then, using the gradient-based method, the parameter identification set a higher weight for the eigenmodes than for the natural frequencies, in order to attempt a good MAC correlation. The results of the parametrization procedures are shown in Table 36.

Table 36 *Modal parameters for the robot manipulator computed by the parametrization algorithms (following* [131]*).*

	1st mode	**2nd mode**	**3rd mode**	**4th mode**
EMA	11.6 Hz	14.6 Hz	21.2 Hz	27.0 Hz
First estimation with evolutionary algorithm	11.1 Hz	15.3 Hz	20.2 Hz	28.6 Hz
Deviation	4.0 %	5.0 %	4.5 %	5.7 %
MAC	0.7	0.5	0.9	0.7
Subsequent identification with a gradient-based algorithm	11.7 Hz	15.0 Hz	20.6 Hz	26.9 Hz
Deviation	1.0 %	2.7 %	2.8 %	1.0 %
MAC	0.9	0.8	0.8	0.8

The values in bold in Table 37 are the parameters corresponding to the four bushings of the robot manipulator that were parametrized by the algorithms.

Table 37 *Stiffness and damping coefficientes for the bushings of the MBS robot manipulator computed by the parametrization procedures (values in bold).*

Parameter		**x-coordinate**	**y-coordinate**	**z-coordinate**
Bushing at “Base-Shoulder”				
Translational	Stiffness (N/mm)	2.4×10^{6}	2.4×10^{6}	2.5×10^{6}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	4.6×10^{9}	4.6×10^{8}	4.6×10^{8}
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at “Shoulder-Upper arm left”				
Translational	Stiffness (N/mm)	**5.0×10^{3}**	**4.4×10^{4}**	**5.0×10^{3}**
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	**8.7×10^{5}**	**1.4×10^{6}**	**0.0**
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at “Shoulder-Upper arm right”				
Translational	Stiffness (N/mm)	**570.0**	**6.5×10^{3}**	**4.8×10^{5}**
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	**3.6×10^{5}**	**5.5×10^{5}**	**3.0×10^{6}**
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at “Upper arm left-Elbow”				
Translational	Stiffness (N/mm)	**123.1**	**238.7**	**1.6×10^{3}**
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	**8.7×10^{5}**	**1.3×10^{4}**	**1.6×10^{4}**
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at “Upper arm right-Elbow”				
Translational	Stiffness (N/mm)	**831.7**	**6.4×10^{3}**	**9.0×10^{3}**
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	**2.0×10^{6}**	**2.3×10^{5}**	**2.8×10^{5}**
	Damping (N·mm·s/deg)	5.7	5.7	5.7

Table 37 (cont.) *Stiffness and damping coefficientes for the bushings of the MBS robot manipulator computed by the parametrization procedures (values in bold).*

Bushing at "Elbow-Forearm"				
Translational	Stiffness (N/mm)	2.3×10^{4}	1.3×10^{4}	1.2×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	2.7×10^{6}	5.7×10^{6}	5.2×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at "Forearm -Hand left"				
Translational	Stiffness (N/mm)	1.7×10^{4}	2.0×10^{4}	1.9×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	5729.6	5729.6	1.7×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at "Forearm -Hand right"				
Translational	Stiffness (N/mm)	1.7×10^{4}	2.0×10^{4}	1.7×10^{4}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	5729.6	5729.6	1.7×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7
Bushing at "Hand-Finger"				
Translational	Stiffness (N/mm)	5.4×10^{3}	5.1×10^{3}	4.0×10^{3}
	Damping (N·mm·s)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Rotational	Stiffness (N·mm/deg)	1.2×10^{7}	5.7×10^{7}	1.7×10^{6}
	Damping (N·mm·s/deg)	5.7	5.7	5.7

The developed algorithm can be employed for the parameterization of stiffness and damping coefficients of any robot manipulator that has been modeled under the criteria presented in Section 4.5.2.2. In case, the MBS model of the robot manipulator is not built in the simulation tool MSC.Adams/View, the interface between DAKOTA and MSC.Adams/View, described by, e.g., the script *run_adams.acf* and the file *simmodell.adm*, need to be adapted.

Once both MBS systems, the mobile platform and the robot manipulator, were successfully adjusted, they were unified into a single model for the mobile manipulator. This new MBS system has to be integrated with the actuation mechanisms, the linear drives (Figure 72) and the gyro stabilizer (Figure 73), whose task is the dynamic compensation of external forces to assure the system stability and, thus, to prevent the mobile manipulator from tip over by abrupt acceleration and deceleration events.

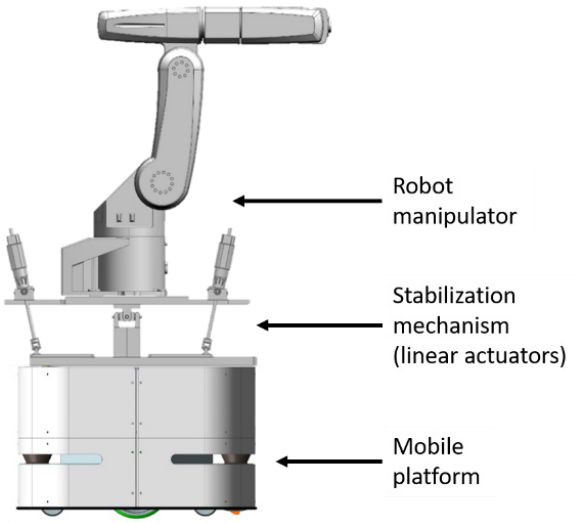


Figure 72 *Linear actuators as external stabilization mechanism for the testing system comprised by the robot manipulator mounted on a small footprint mobile platform.*

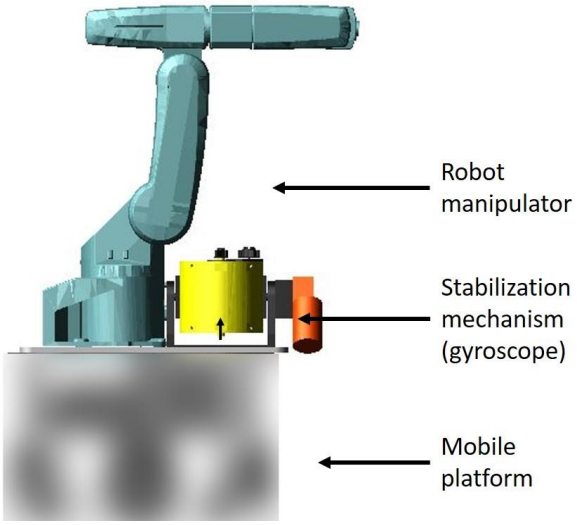


Figure 73 *Gyroscope as external stabilization mechanism for the testing system comprised by the robot manipulator mounted on a small footprint mobile platform.*

4.5.3 Mechatronic co-simulation of testing system

The effectiveness of the three stabilization strategies proposed in approach A (referred to as "inclining/tilting", "conservation of angular momentum" and "gyroscopic effect") is assessed by implementing series of mechatronic co-simulations, which perform critical braking profiles for the mobile platform. The following sections present an overview of the proceedings carried out particularly for the testing system.

4.5.3.1 Solution stability analysis of solvers

The required solution stability analysis for the in MSC.Adams/View available solvers (see Section 4.4) were carried out using the MBS model of the mobile manipulator, capable to represent a dynamic critical state of the system due to the oscillations generated by the spring suspension located in the drive wheels.

During mechatronic co-simulations, each stand-alone sub-simulator exchanges its output variables at a certain communication interval. The sub-simulators consider their inputs as constant over the integration time of each communication time step, although in reality their values vary continuously. As a result, large comminucation intervals can lead to errors in the numeric estimations and affect the simulation accuracy. In this work, the communication interval was defined following the technique for the sampling time estimation in control engineering, which states that the sampling time must be at least ten times higher than the smallest time constant presented in the control system [160]. Hence, considering the time constants of the control systems obtained in Section 4.5.3.3 and Section 4.5.3.4, the data transmission interval for all mechatronic co-simulations was set to 0.001 s.

Three different error tolerances were considered into the solution stability analysis. The stability of the integrators (and their corresponding tolerances) was evaluated regarding the linear acceleration the mobile platform adopted during the MBS simulations. Their corresponding outcomes are outlined in the Table 38, Table 39 and Table 40, respectively:

- Integrator GSTIFF with index I3 and with index SI2:

Table 38 *Comparison of different indexes and tolerances for integrator GSTIFF (following* [112]*).*

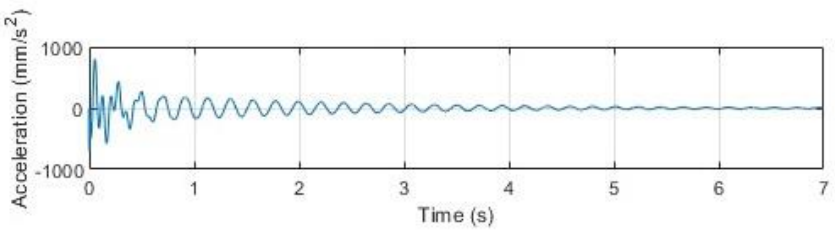
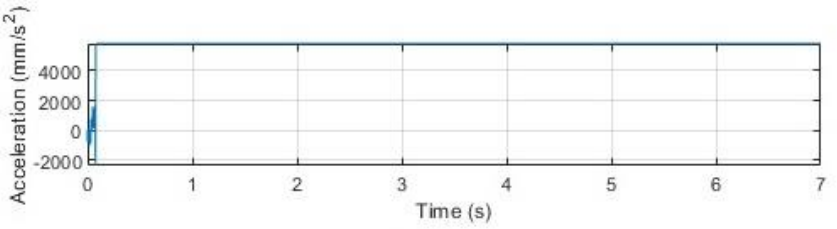
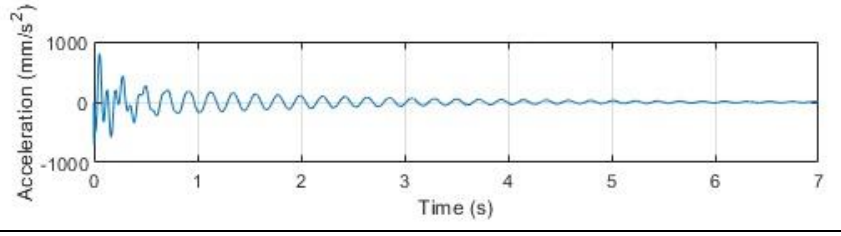
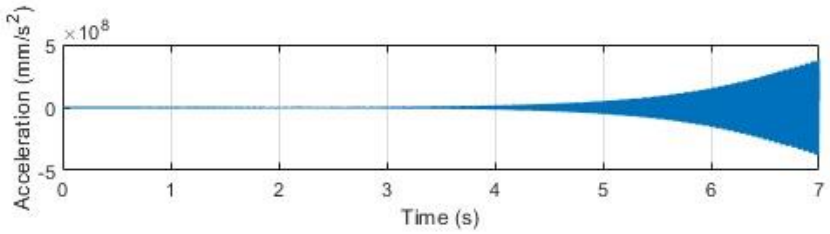
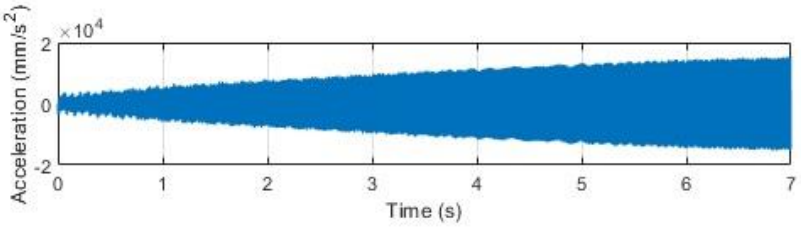
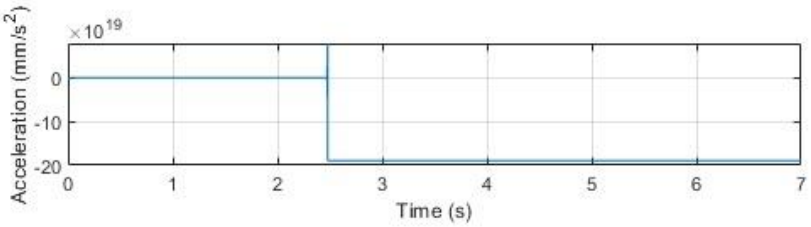
Tolerance	**Index I3**
1×10^{-3}	
1×10^{-5}	
1×10^{-2}	

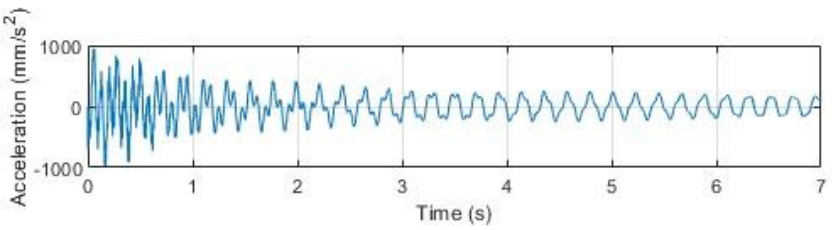
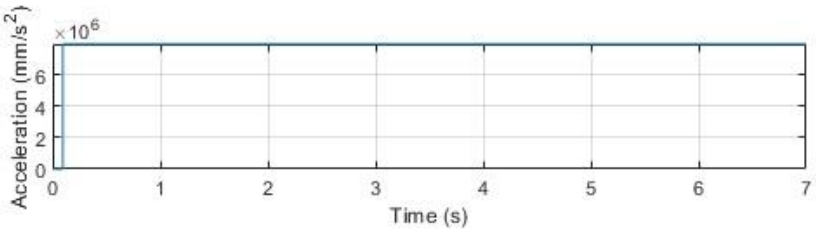
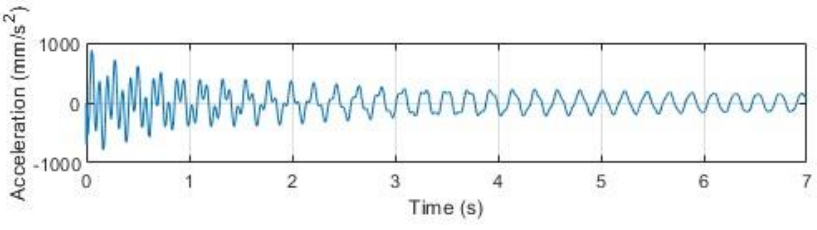
Table 38 (cont.) *Comparison of different indexes and tolerances for integrator GSTIFF ([112]).*

Index SI2	
1×10^{-3}	
1×10^{-5}	
1×10^{-2}	

The results with the different error tolerances made the GSTIFF integrator with index-3 (I3) or with index-2 (SI2) unsuitable for the simulation, since the output signals for each index (I3 or SI2) do not exhibit the same course among the different tolerances.

- Integrator WSTIFF with index I3 (only available):

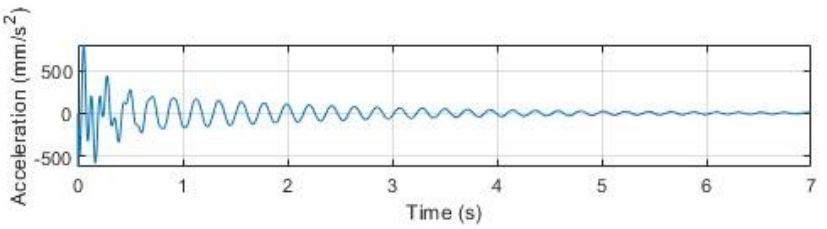
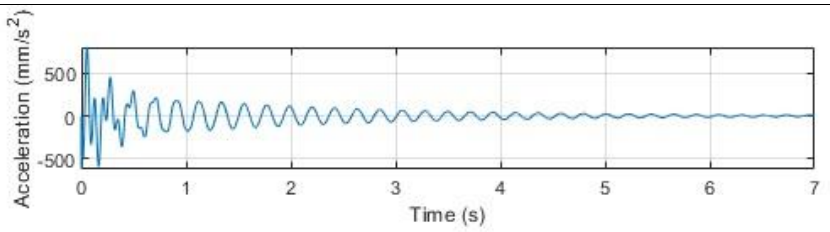
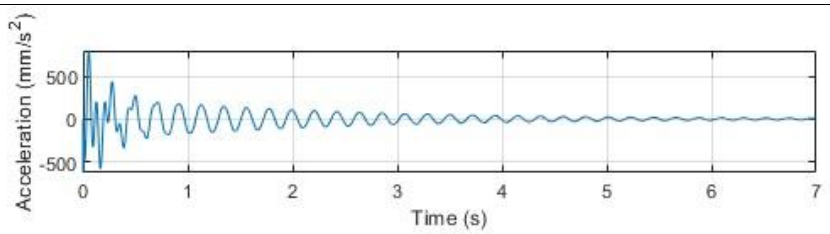
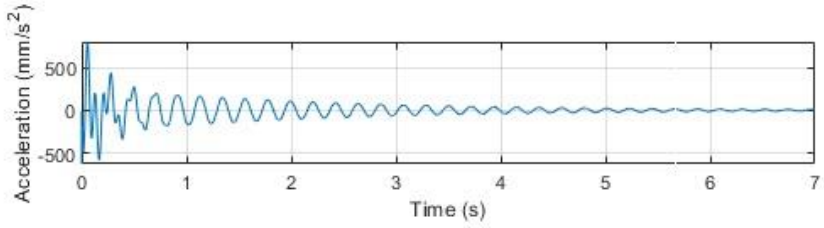
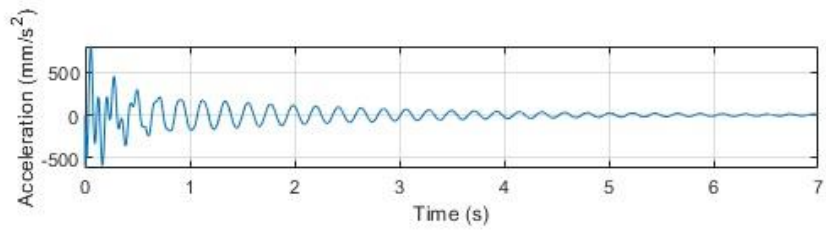
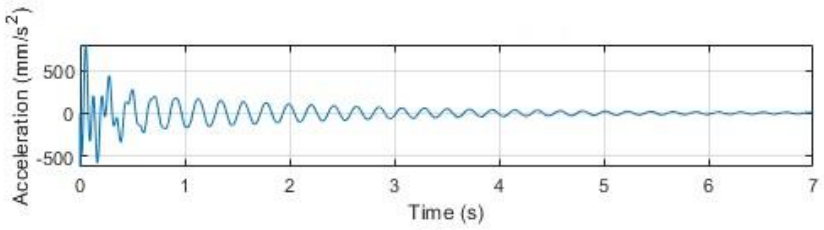
Table 39 *Comparison of different tolerances for integrator WSTIFF with Index-I3 (following* [112]*).*

Tolerance	**Index I3**
1×10^{-3}	
1×10^{-5}	
1×10^{-2}	

Here, as with the GSTIFF integrator, the acceleration did not produce equivalent results for the graduated tolerance values. Thus, this integrator was also unsuitable for the simulations.

- BDF CONSTANT integrator with an index I3 (top) and an index SI2 (bottom):

Table 40 *Comparison of different tolerances for integrator BDF constant (following* [112]*).*

Tolerance	Index I3
1×10^{-3}	
1×10^{-5}	
1×10^{-2}	
Index SI2	
1×10^{-3}	
1×10^{-5}	
1×10^{-2}	

As illustrated in Table 38, the results for all tolerance values showed only minimal differences between the acceleration graphs. Therefore, the BDF CONSTANT integrator is

considered as the suitable solver to achieve the best approximation for the dynamic multibody simulation of the particular system. The faster responses were obtained with the I3 index and with an error tolerance of 1×10^{-3}, which were employed for the further co-simulations.

4.5.3.2 General simulation statements and setups

Figure 74 shows how the force on the front support wheels of the mobile platform increases depending on the applied braking deceleration: the mass of the entire system, before the disequilibrium occurs, is distributed among the six wheels. If the back-support wheels lift-off, then only the drive wheels and the front wheels support the whole system weight (indicated with the white line in the graphic on Figure 74). And, if the drive wheels lift additionally, the entire weight is held only by the front wheels, pointed out with the black line in the graphic on Figure 74.

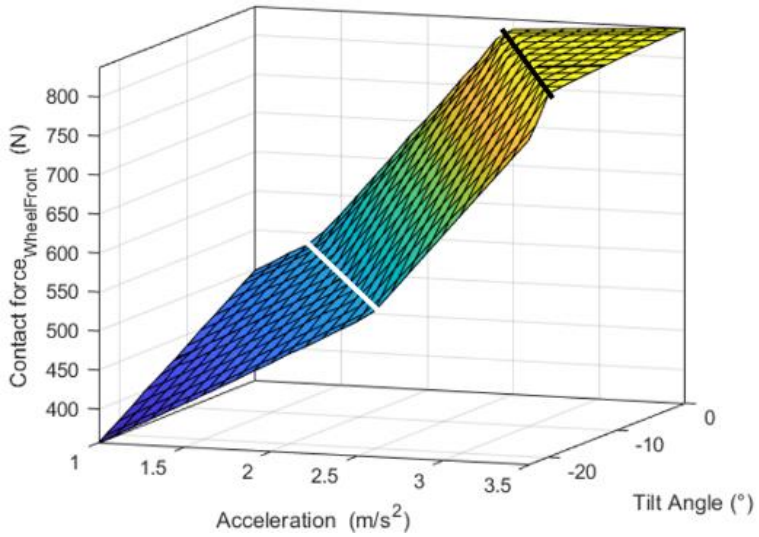


Figure 74 *Braking force influence on the front support wheels as a function of the travelling deceleration and the tilt angle generated by the linear actuators* [113].

Figure 74 reveals that if the robot manipulator has no tilt angle and no stabilization strategy, the system tips over at a deceleration of 1.4 m/s^2. If the robot manipulator adopts its equilibrium position (COG aligned with the axis of the universal joint at approx. 11.86°), a deceleration of 2 m/s^2 causes both rear support wheels to lift off.

The mobile platform is accelerated until it reaches a maximal velocity, particular for each scenario to be tested; then, the mobile platform travels with a constant linear velocity for a short time to ensure a stable steady state before the braking process starts; after that, the platform brakes abruptly.

The mechatronic co-simulations of both strategies “Inclining/Tilting” and “Conservation of angular momentum” employed the same velocity curve for the mobile manipulator during the simulations. To demand a particularly difficult testing scenario, two braking processes were carried out for all simulations, with a deceleration of 2 m/s^2 and with 2.1 m/s^2, respectively, in order to make the linear drives act in its technical extremes twice, for consecutive periods. These braking processes are presented in Figure 75.

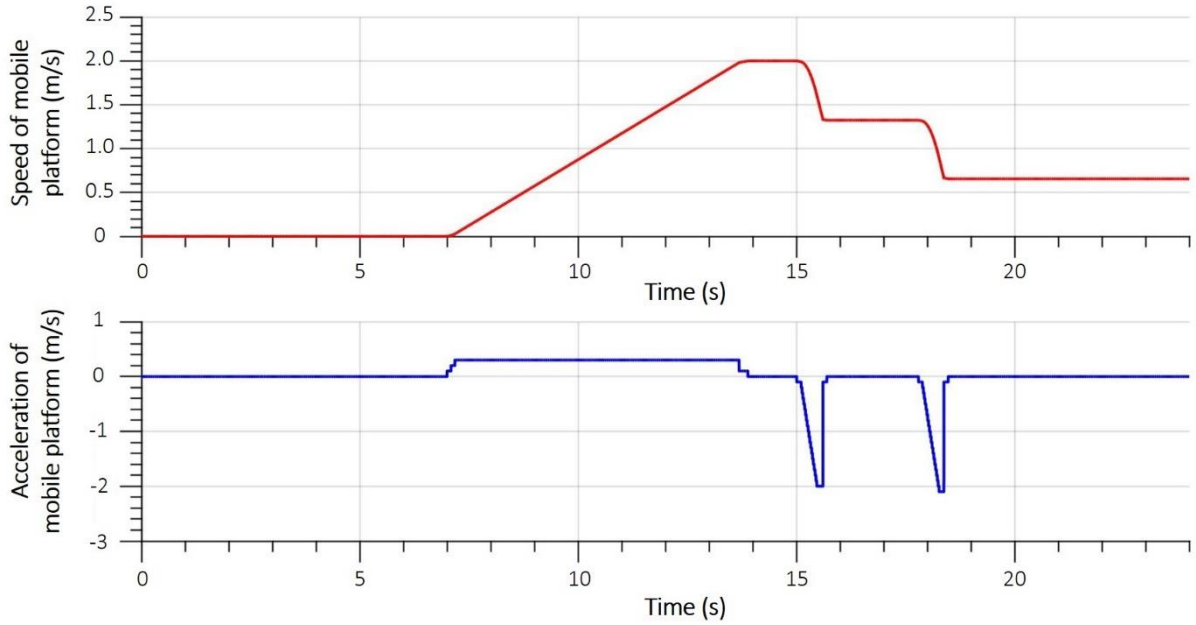


Figure 75 *Braking processes implemented for the linear drives strategy simulations* [112].

On the other hand, due to the simplicity of the gyro mechanism and its control, more extensive and diverse braking processes were implemented during the mechatronic co-simulations. The acceleration and deceleration slopes were based on the simple profile provided by the manufacturer (curves shown in Figure 76), being adapted depending on the scenarios to be tested.

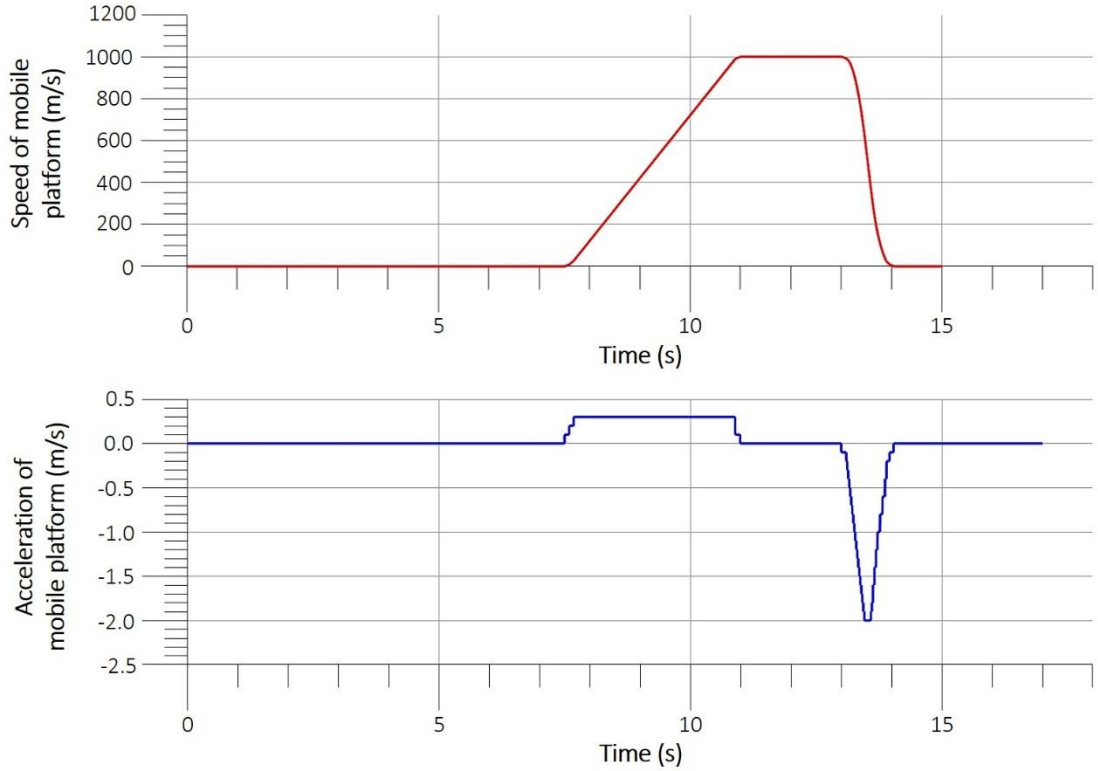


Figure 76 *Braking process which served as the basis profile for the gyro stabilizer simulations* [112].

4.5.3.3 Mechatronic co-simulation of stabilization strategies employing linear drives

For the mechatronic co-simulations of the external mechanism, the Linear Drive SFL with DC-Motor [161] was adopted. Using the information supplied by the manufacturer ([161,162]) and benchtop measurements on the real motor together with simple data fitting following the approach in [163], the motor was identified as a second-order system which, according to Eq. *(78)*, resulted in

$$G_M(s) = \frac{489}{(0.01337s + 1)(0.01025s + 1)}. \tag{82}$$

Taking into account the gear transmission ratio of 1:10 [164], the motor transfer function can expressed as

$$G_M(s) = \frac{48.9}{(0.01337s + 1)(0.01025s + 1)}. \tag{83}$$

Since the system does not contain an I-component, PI-controllers can be used to reach stationary accuracy for the speed control loop (inner control loop) [165]. By employing only P and PD components, a permanent control deviation would be observed in the system [166]. The time constant for the required PI controller for the speed control loop was estimated with the rules according to Kuhn [167,168], since the system dynamic behavior (transfer function) was available from the experimental setups.

$$\sum T = T_1 + T_2 \tag{84}$$

$\sum T$ was obtained by searching the point where the areas A_1 and A_2 of the step response graph are equal. As in Figure 77, the resulting sum of the time constants for the case considered is $\sum T = 0.02\ s$.

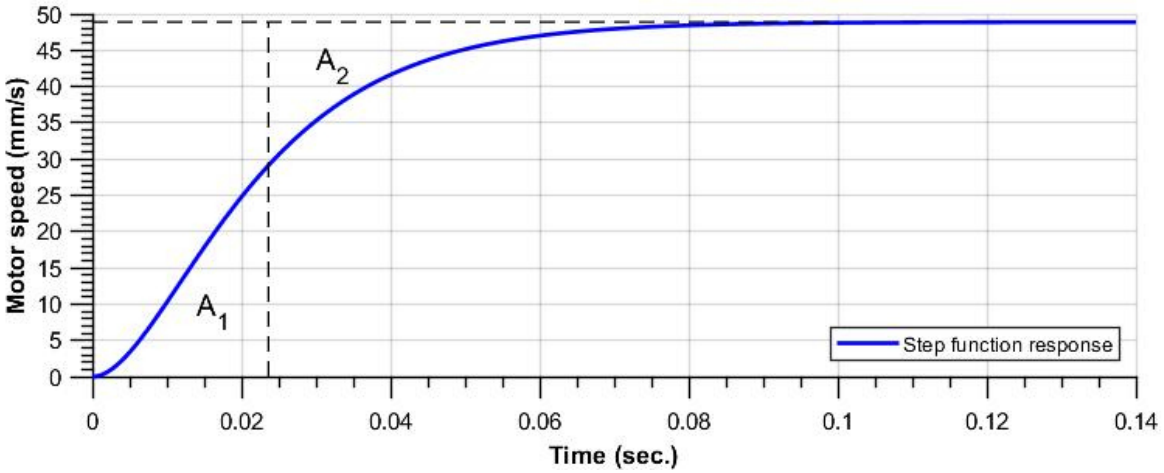


Figure 77 *Empirical estimation of the motor time constant* [112].

Then, the controller gain K_c and the reset time T_N were estimated employing the equations for setting rules of a PI controller according to Kuhn [160], as follows

$$K_c = \frac{1}{K_s} = \frac{1}{48.9} = 0.02045 \tag{85}$$

where K_s is the gain factor of the controlled system.

$$T_N = 0{,}7 \cdot \sum T = 0.7 \cdot\ 0.02362 = 0.01653. \tag{86}$$

For a PI controller, the proportional factor, K_P, can be considered the same as the controller gain,

$$K_P = K_c = 0.02045 \tag{87}$$

and the integral factor, K_I, can be then obtained with

$$K_I = \frac{K_P}{T_N} = \frac{0.02045}{0.01653} = 1.23685 \tag{88}$$

$$\frac{K_I}{K_P} = \frac{1.23685}{0.02045} = 60.48143. \tag{89}$$

By employing these parameters, the step response reached its target value after 0.132 s and had an overshoot of 3.8%. Based on a root locus analysis, the proportional gain K_P was minimally increased from 0.02045 to 0.02192 and, thus K_I to 1.3257. After that, the pole of the PI controller migrated to -54.3 1/s. As a result, the step response reached the target velocity value after 0.117 s with 5% overshoot.

After the velocity control loop (inner controller) had been prepared, the position controller (outer control loop) was designed. The steady-state accuracy of the position control loop was already ensured, since the velocity must be integrated to get positioning values. Thus, if the system did not overshoot at all, a P controller could be employed to ensure that the position is approached accurately. With a proportional gain K_P of 15.582, its step response took 0.367 s until the stationary target value was reached and showed no overshoot, representing a proper controlling behavior.

A comprehensive review of the design of the control system implemented for the particular case of the testing system lies beyond the scope of this study. The designed control is illustrated in Annex A.3.

Finally, the so-called motions were implemented for the drive wheels to reproduce the acceleration and braking process of the mobile platform.

The corresponding assignment of signals for the mechatronic co-simulations are indicated in Table 41.

Table 41 *Inputs/outputs for mechatronic co-simulations of the linear drives´stabilization strategies.*

Input signals for MBS (MSC.Adams/View)	Output signals from MBS (MSC.Adams/View)
• Stroke position of linear drive 1 • Stroke position of linear drive 2 • Stroke position of linear drive 3 • Angle of upper plate of stabilization mechanism • Linear travel velocity of mobile platform	• Acceleration of linear drive 1 • Acceleration of linear drive 2 • Acceleration of linear drive 3 • Global X-acceleration of mobile platform • Global Y-acceleration of mobile platform • Global Z-acceleration of mobile platform • Force between ground and left support wheel back • Force between ground and right support wheel back • Force between ground and left drive wheel • Force between ground and right drive wheel • Force between ground and left support wheel back • Force between ground and right support wheel back
Output signals from Controls (Matlab/Simulink)	Input signal for Controls (Matlab/Simulink)

4.5.3.4 Mechatronic co-simulation of stabilization strategy using gyroscope

First, estimations about the effectiveness of the gyroscopic system as stabilization strategy were carried out using Eq. *(67)* in order to proceed with its detailed design. Starting with a maximal deceleration during the braking process (a) of −1 m/s^2, a compensation torque (M_c) of approx. −40 N·m had to be applied by the gyroscope to the whole system to achieve a stabilization during the standard braking process.

First, a very simplified model of a gyroscope was built as MBS system in order to estimate the design parameters required to compensate the estimated torque. The rotational speed (ω_g) of the mass was implemented in MSC.Adams/View by a motion on the upper axis of rotation. The tilting speed (ω_p) of the mass was imposed as two motions on the lateral axes of rotation.

After the first analytical and simulative estimations were performed, a gyro stabilizer consisting in a mass with a moment of inertia of I=1414.1 kg·mm^2 and a fix rotational speed of n_{motor}=5000 min^{-1} was designed, so that the required compensation torque to stabilize the mobile manipulator was achieved. The final model consisted of the mass (with its associated moment of inertia), the cage, the supports and the motors (gears) is displayed in Figure 78.

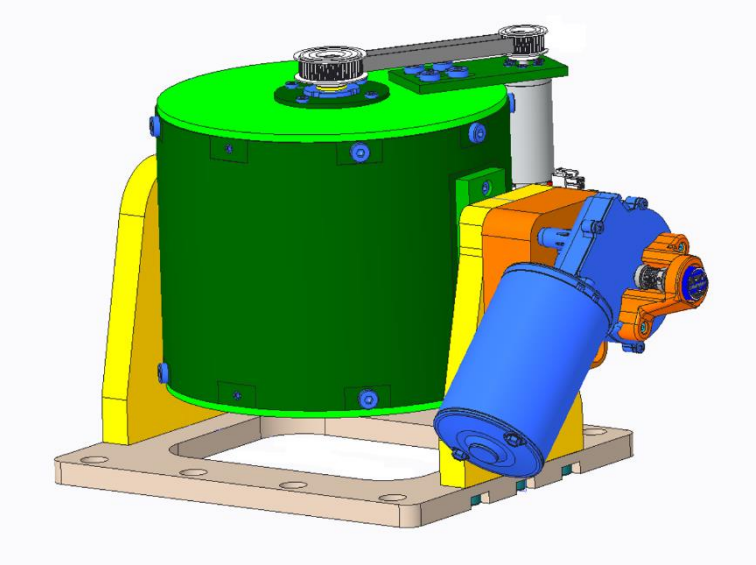


Figure 78 *Designed gyroscope for the further analyses.*

Two light weight servo motors with high positioning accuracy, [169] for the rotary motor and [170] for the tilting motor, were selected. Their corresponding control algorithms had to be implemented in Matlab/Simulink. The mass rotational speed could be initially set to the target value by a simple speed control. The precession moment should be controlled depending on the required deflection angle of the tilting motor by means of a position control.

Since not all motor parameters were provided by the manufacturers, the step response of both, the tilting motor and the rotary motor, were experimentally determined. The rotary motor was identified as first-order system and the tilting motor as first-order system with an additional integration element [171]. Both motors were modeled as black box in Matlab/Simulink according to Eq. *(77)* as follows

$$G_{TM}(s) = \frac{4.63}{(0.063 \cdot s + 1)} \tag{90}$$

for the tilting motor, and

$$G_{RM}(s) = \frac{523.6}{(22 \cdot s + 1)} \tag{91}$$

for the rotary motor.

Due to the fact the experimental determination of the controlled system showed that the tilting motor itself contained an I-component, a PD-controller was implemented. Its controller design parameters were established according to the Tietze-Schenk tuning method [172,173], in which the derivative time T_V of a PD-controller is determined by

$$T_V = \frac{T_p}{2 \cdot \pi} \tag{92}$$

using the measured period duration of the step function response, T_p=0.099 s, and according to Eq. *(92)*, T_V resulted in 0.015 s. The value for the corresponding proportional-derivative gain, K_{PD}, after the controller setting was 0.0225.

For the rotary motor, a PI-controller was designed. Its reset time, T_N, was estimated as 22 s and its proportional-integral gain, K_{PI}, as 7.15×10^{-4}.

A comprehensive review of the design of the control systems implemented for the particular case of the testing system lies beyond the scope of this study. Their corresponding block diagrams can be seen in Annex A.4.

The maximum torque generated by the gyroscope as well as the angular position and velocity of its tilting motion were set as the evaluation parameters for the motor closed-loop controls.

The input and output parameters implemented for the mechatronic co-simulations are described in Table 42.

Table 42 *Signal assignment for the mechatronic co-simulations of the stabilization strategy employing the gyro effect.*

Input signals for MBS (MSC.Adams/View)	Output signals from MBS (MSC.Adams/View)
• Position of tilting motor • Rotational speed of tilting motor • Rotational speed for rotary motor • Linear travel velocity of mobile platform	• Tilting angle of gyroscope • Tilting speed of gyroscope • Rotational speed of rotary motor • Generated torque measured in the upper plate of the mobile platform • Force between ground and left support wheel back • Force between ground and right support wheel back • Force between ground and left drive wheel • Force between ground and right drive wheel • Force between ground and left support wheel back • Force between ground and right support wheel back
Output signals from Controls (Matlab/Simulink)	Input signal for Controls (Matlab/Simulink)

4.5.4 Validation of stabilization strategies employing testing system

To ensure an accurate comparison between the validation scenarios, the contact forces and distances between the ground and each wheel of the mobile platform were used as evaluation criterion during the mechatronic co-simulations: contact forces greater than 0 N or distances equal to 0 mm mean that the mobile manipulator is not tipped-over. In other words, the higher the contact force distributed across all the wheels, the more stable the robotic system is.

4.5.4.1 Evaluation of stabilization strategy "Inclining/Tilting" employing testing system

For the stabilization strategy "Inclining/Tilting", the mobile platform has been accelerated until its maximum velocity (yellow area in Figure 79), followed by a short traveling path with constant speed (green area in Figure 79). Immediately after the IMU-measurements in the MBS model revealed that the system achieved a stable steady state, the mobile platform braked to stop the system abruptly (red area in Figure 79). Just before the mobile platform braked, the robot manipulator was tilted backwards. Tilting the robot manipulator backward shifts its COG to the back, which keeps the system more stable while braking. Figure 79 shows the robot manipulator over time, during acceleration and braking of the mobile platform (yellow curve in the graph).

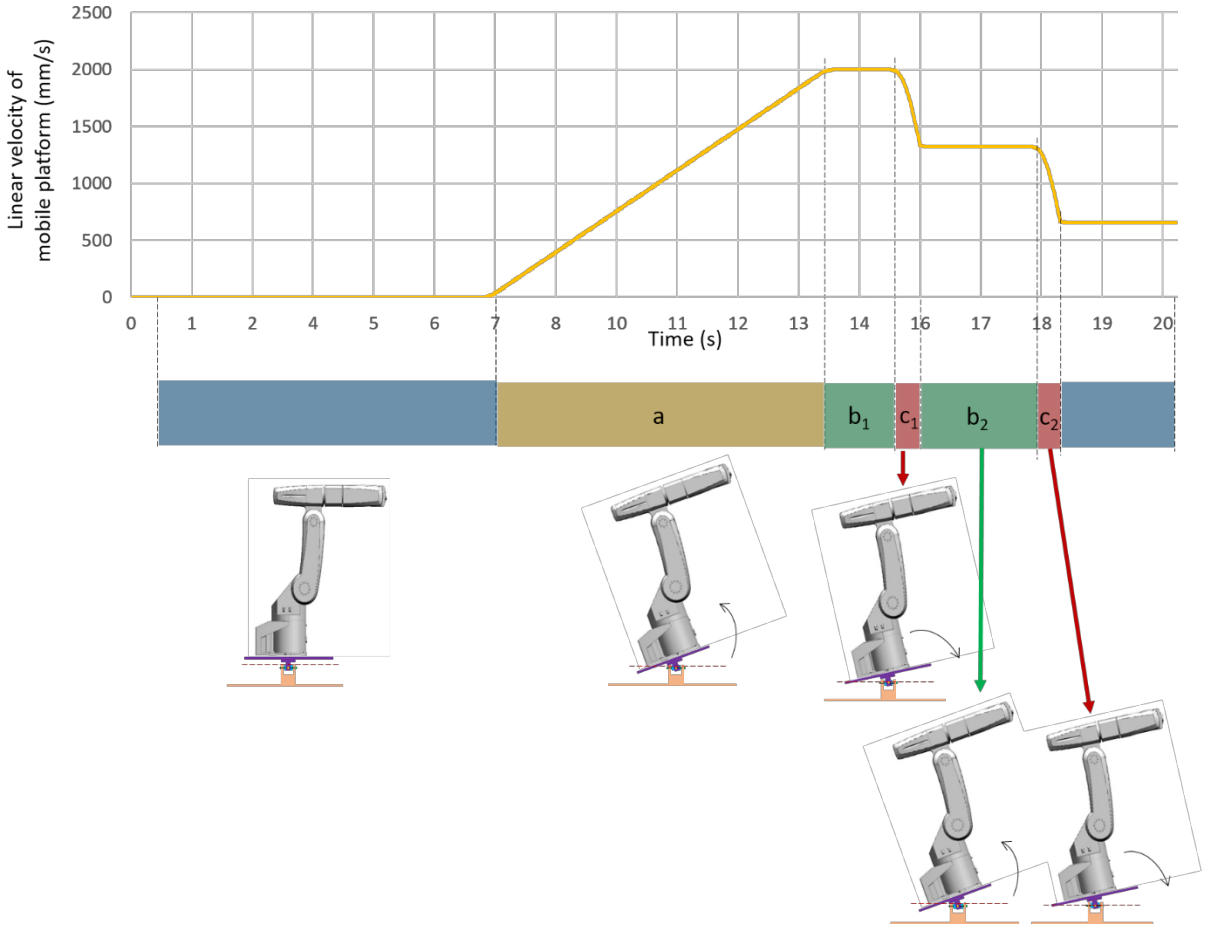


Figure 79 *Inclination/tilting of the manipulator during braking process* [148].

Simulations without the stabilization strategy demonstrated that during the accelerations of 2 m/s^2 and 2.1 m/s^2, the rear support wheels lifted off the ground, lacking of any ground contact forces. In comparison, the stabilization strategy revealed that even a deceleration about

2.1 m/s^2 did not exhibit instabilities if the mobile manipulator was tilted approx. −4° before the brake occurred. In this scenario, the contact force between the back support wheels and the ground was 42.6 N, as displayed in Figure 80.

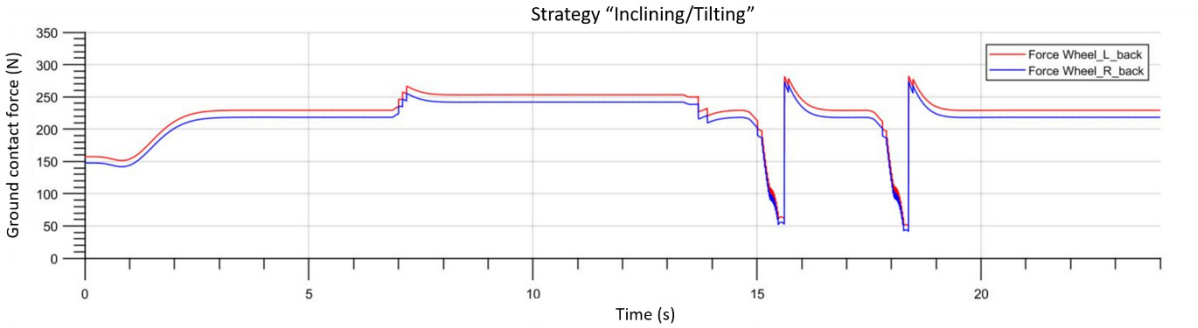


Figure 80 *Contact force measured with the "inclining/tilting" stabilization strategy* [112].

Nevertheless, the drawback of this strategy is that the countermeasure has to be done prior to the braking operation. In fact, the exact value of the braking deceleration cannot be easily known beforehand and, thus, the angle set point the robot manipulator should have adopted before the brake occurs is unknown. In that regard, the stabilization strategy is only applicable if the mobile manipulator is equipped with warning field devices, such as safety laser scanners, which help to estimate the eventual braking acceleration based on the current velocity of the mobile platform when an object has been detected.

Furthermore, there will always be a superposition of the impulse with the effect shown in the next section.

4.5.4.2 Evaluation of stabilization strategy "Conservation of angular momentum" employing testing system

During the braking process of the mobile platform, the robot manipulator´s COG was turned from the equilibrium position in the direction of travel of the mobile platform (opposite to the last explained concept), as represented in the diagram in Figure 81. As a result, the angular acceleration of the robot manipulator counteracted the braking acceleration of the mobile platform following the principle of conservation of angular momentum. After the braking process was finished, the robot manipulators´ COG was returned to the equilibrium position.

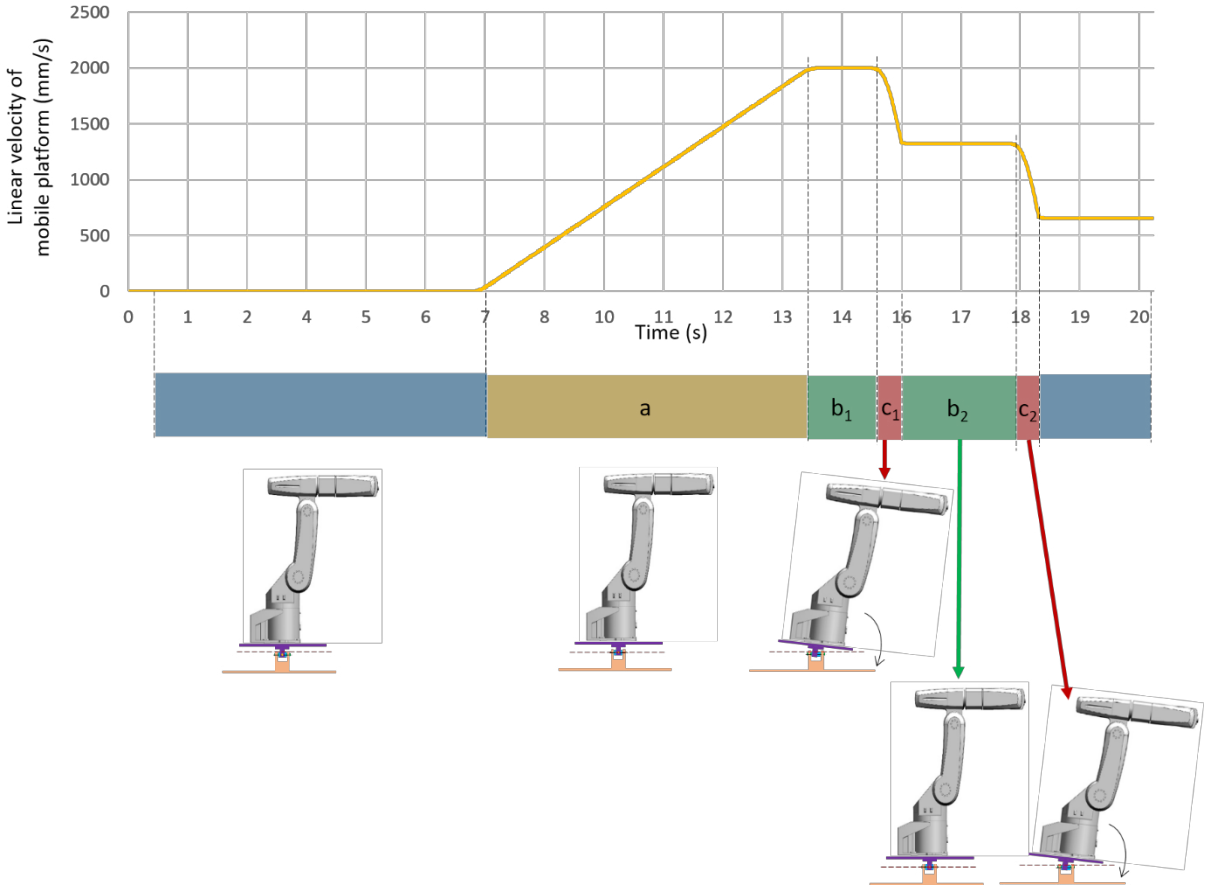


Figure 81 *Conservation of angular momentum of the manipulator during braking process. Based on* [148].

In order to objectively compare the effectiveness of this strategy with the strategy "inclining/tilting", the same traveling profile for linear drives from Section 4.5.4.1 was used, but with opposite sign because the robot manipulator is now impelled in the direction of travel of the mobile platform.

The outcomes of the stabilization strategy based on the conservation of angular momentum are shown in Figure 82.

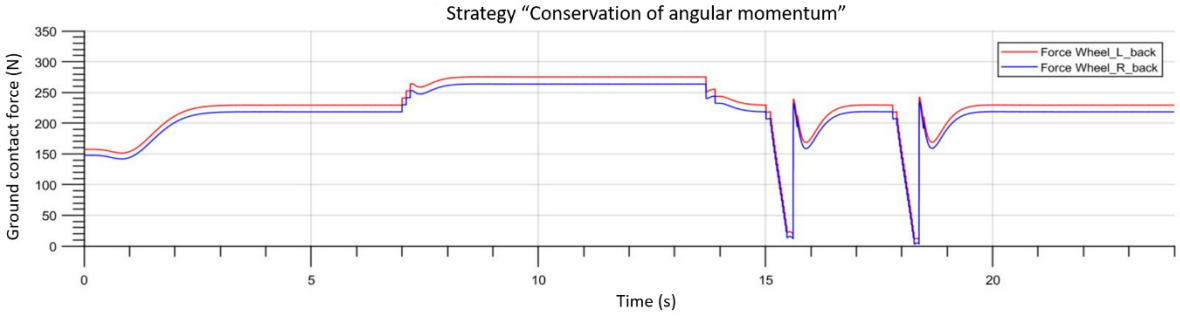


Figure 82 *Contact force measured with the "conservation of angular momentum" stabilization strategy* [112].

It can be observed that effect of the stabilization strategy is critical and needs to be enhanced at some point in the braking profile with an additional short angular momentum. In practice, this technique is more suitable when braking process appear unforeseen, or when the robot manipulator is already tilted to the back.

4.5.4.3 Evaluation of stabilization strategy "Gyroscope" employing testing system

The yellow curve in Figure 83 illustrates an example of a basic braking process implemented for the mechatronic co-simulations with the gyro stabilizer and its corresponding moment of action. The mass of the gyro stabilizer acquired a rotational speed of 5000 min^{-1} by the beginning of the simulations (t=0, blue area in Figure 83). While accelerating the mobile manipulator, the mass of the gyroscope was tilted counterclockwise by 0.78 rad (Figure 83-yellow) and, during its deceleration, the gyroscope was returned to its starting position (0°), tilting it by −0.78 rad (Figure 83-red).

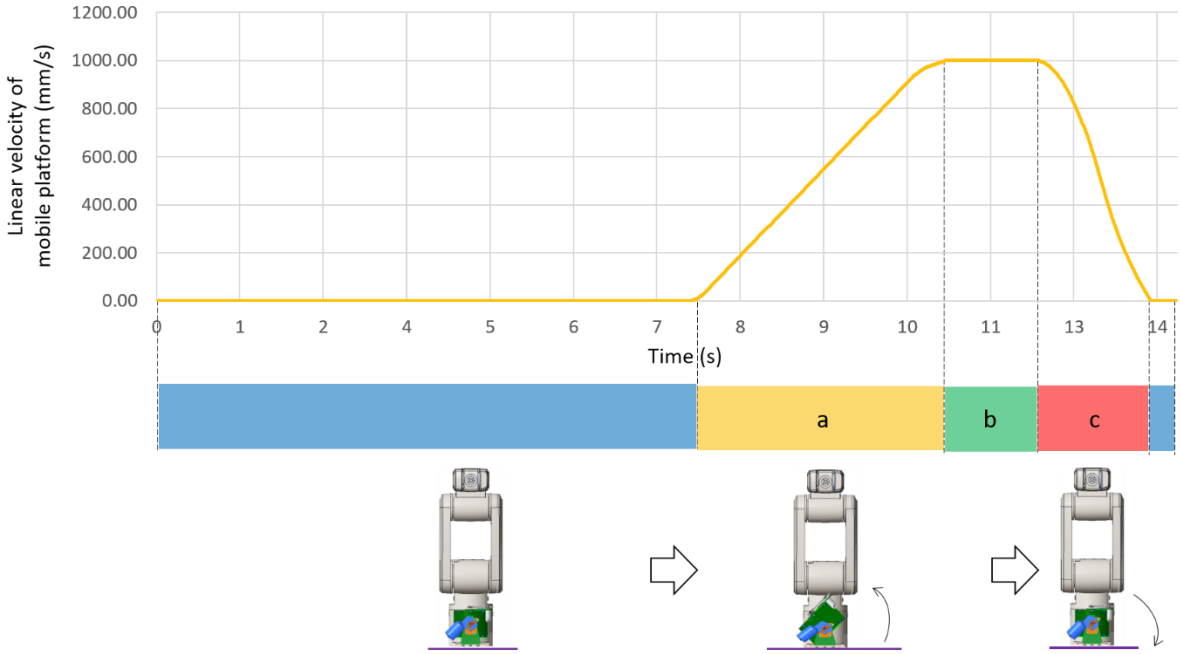


Figure 83 *Gyroscopic stabilizer action during braking process* [148].

During the acceleration and braking process of the mobile platform, the speed of the tilting motor was set to ω_p=0.60 rad/s. The gyro stabilizer was able to generate the precession torque (M_p) of approx. 60 N·m, shown in Figure 84. The first peak in Figure 84 indicates the inclination of the flywheel mass of the gyroscope at +0.78 rad. The second and higher peak illustrates when the flywheel mass tilts to the opposite side, at -0.78 rad. The asymmetry of the curves is inferred by an unbalanced mass distribution due to the rotary motor attached to the flywheel mass.

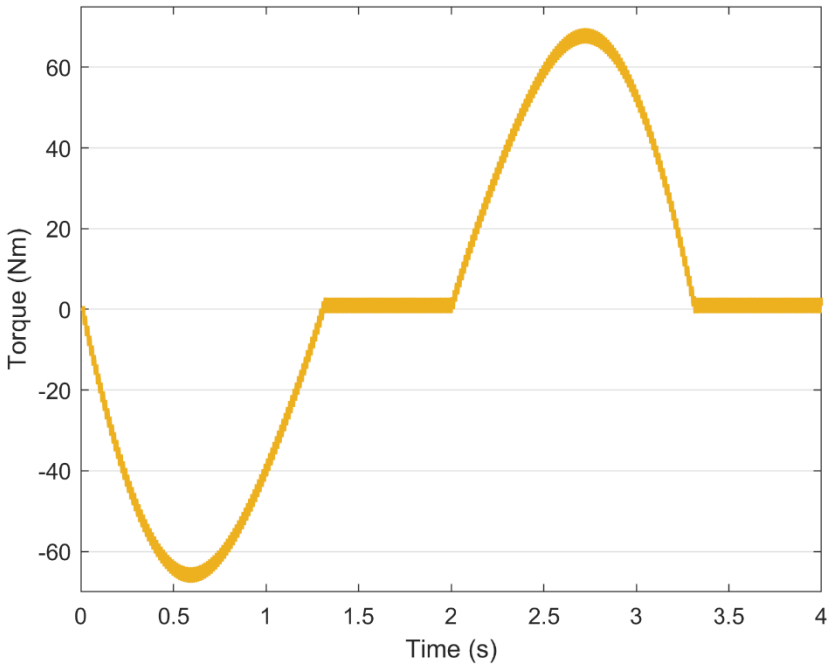


Figure 84 *Torque achieved by the simple gyroscope model* [148].

In order to assess the effectiveness of the proposed gyroscope stabilizer, diverse scenarios were implemented by means of mechatronic co-simulations. For each of the scenarios introduced in the following pages, the basic braking process shown in Figure 76 was adapted to produce the intended acceleration and deceleration behavior of the mobile platform. The measured distance between the ground and the shaft axis of the rear wheels (37 mm) served as reference for the evaluation of the gyroscopic effect, since it represents the state when all the wheels of the mobile platform have physical contact with the ground. If this distance increases above the reference (37 mm), a tip-over of the mobile manipulator is detected during the braking process.

For the purpose of analysis, the mechatronic co-simulations were first carried out without the activation of the gyro stabilizer, in order to identify the events at which the mobile manipulator experienced instabilities. Then, further mechatronic co-simulations under the same circumstances were performed with the activated gyro stabilizer.

- Scenario 1 and scenario 2 performed a braking process from 1.1 m/s (scenario 1) or rather from 1.2 m/s (scenario 2) to 0 m/s within 1.0 s to evaluate how small changes in the deceleration of the mobile platform affect the performance of the gyroscope. Without the stabilization mechanism, these decelerations produced a significantly larger lift-off of the rear wheels from the ground (>25 mm) compared to each other.
  The co-simulations exhibited a complete compensation of the instabilities by tilting the gyroscope 0.78 rad in 1.3 s for both scenarios, as shown in Figure 85 (top) and Figure 85 (bottom), respectively.

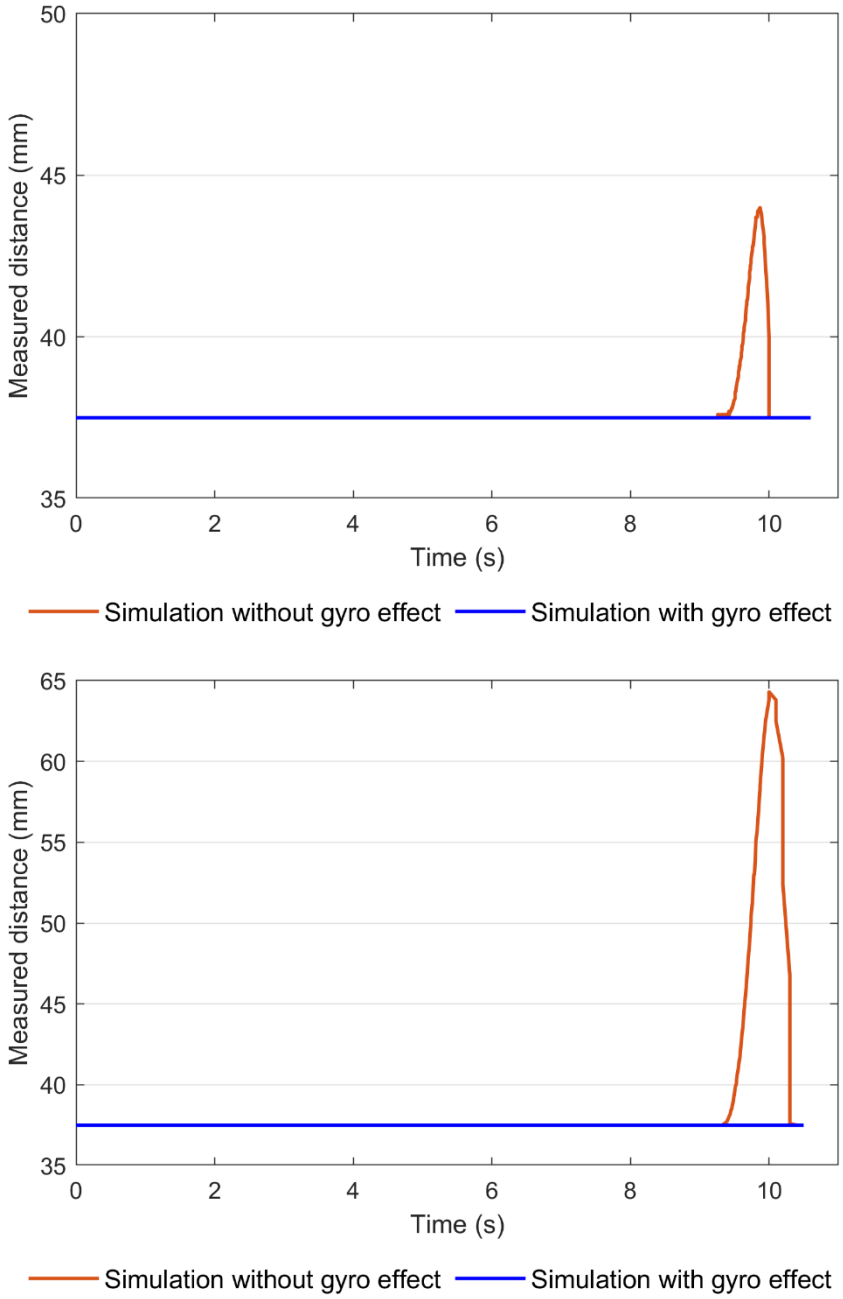


Figure 85 *Results from scenario 1 (top) and scenario 2 (bottom)* [174].

- Scenario 3 examined the minimum angle the gyroscope needs to compensate the braking process of scenario 2 (1.2 m/s to 0 m/s). For this purpose, the target value for the tilting angle of the gyroscope was increased iteratively, starting from 0.17 rad, until the lift-off of the rear wheels of the mobile platform was completely prevented.
  Figure 86 compares the results from the co-simulations with the different tested tilting angles. It can be seen that with a tilting angle of 0.33 rad, the mobile platform had physical contact with the ground throughout the entire braking process.

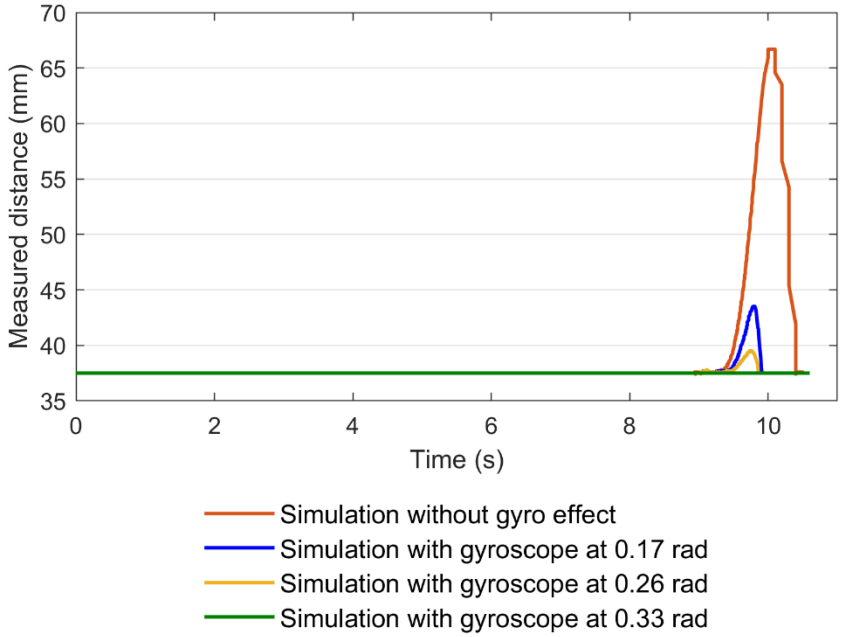


Figure 86 *Results from scenario 3* [174].

- Scenario 4 investigated the minimal time span at which the mobile manipulator can completely stopped without suffering any instabilities if the gyro stabilizer is positioned at 0.78 rad in 1.3 s during the braking process.
  The simulations showed that the minimum possible braking time the mobile platform could induce from a velocity of 1.0 m/s to 0 m/s was 0.8 s. Likewise, if the mobile platform traveled at a speed of 1.2 m/s, the minimum possible braking time was 0.9 s. Both corresponding results are shown in Figure 87.

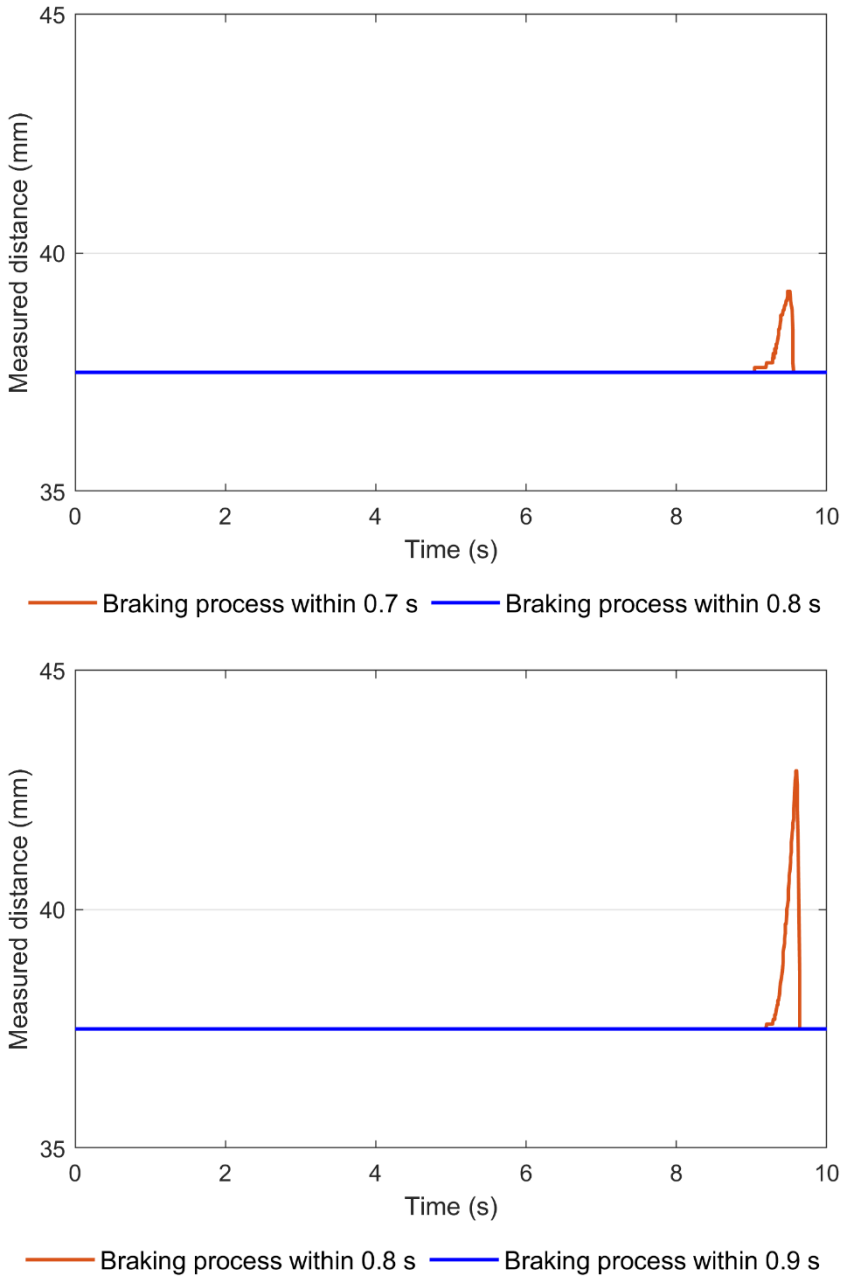


Figure 87 *Results from scenario 4 for a braking process from 1 m/s to 0 m/s (top) and from 1.2 m/s to 0 m/s (bottom)* [174].

- Scenario 5 aimed to evaluate the effectiveness of the gyro stabilizer under three directly successive extreme accelerations (from 0 m/s to 1.2 m/s within 1 s) and braking (from 1.2 m/s to 0 m/s within 1 s) processes.
  Figure 88 illustrates how the gyroscope ensured the stability of the mobile manipulator for all of them and even no instabilities occurred during turning back the gyroscope to its initial state after each deceleration (no peaks in green curve).

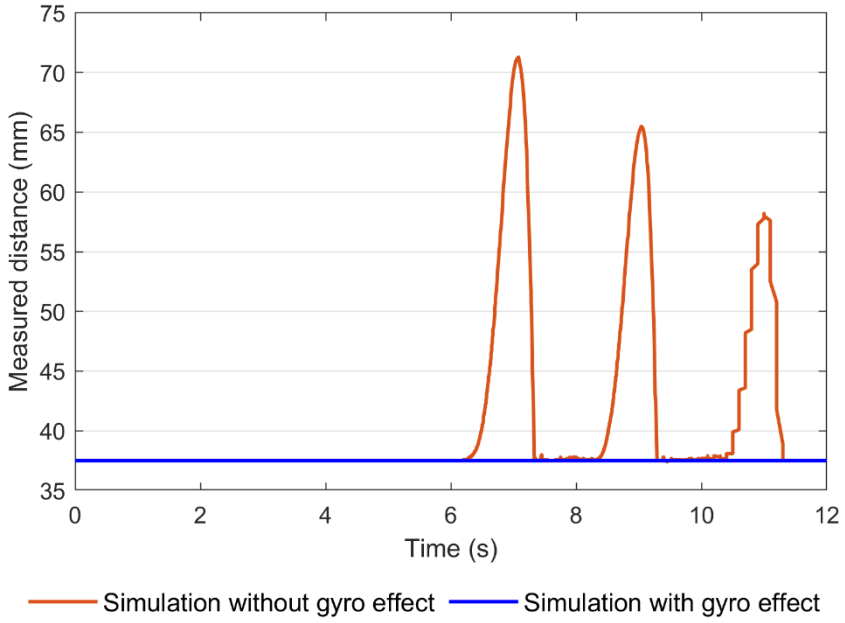


Figure 88 *Results from scenario 5* [174].

- Scenario 6 simulated the same conditions as scenario 2 (deceleration in 1 s from 1.2 m/s to 0 m/s), but this time, the robot manipulator adopted two worst-case positions: its arm was vertical and horizontal aligned, thus shifting the system's COG to be affected by the worst-case deceleration profile.
  As illustrated in Figure 89, the gyro stabilizer was also able to compensate instabilities for both worst-case positions of the robot manipulator.

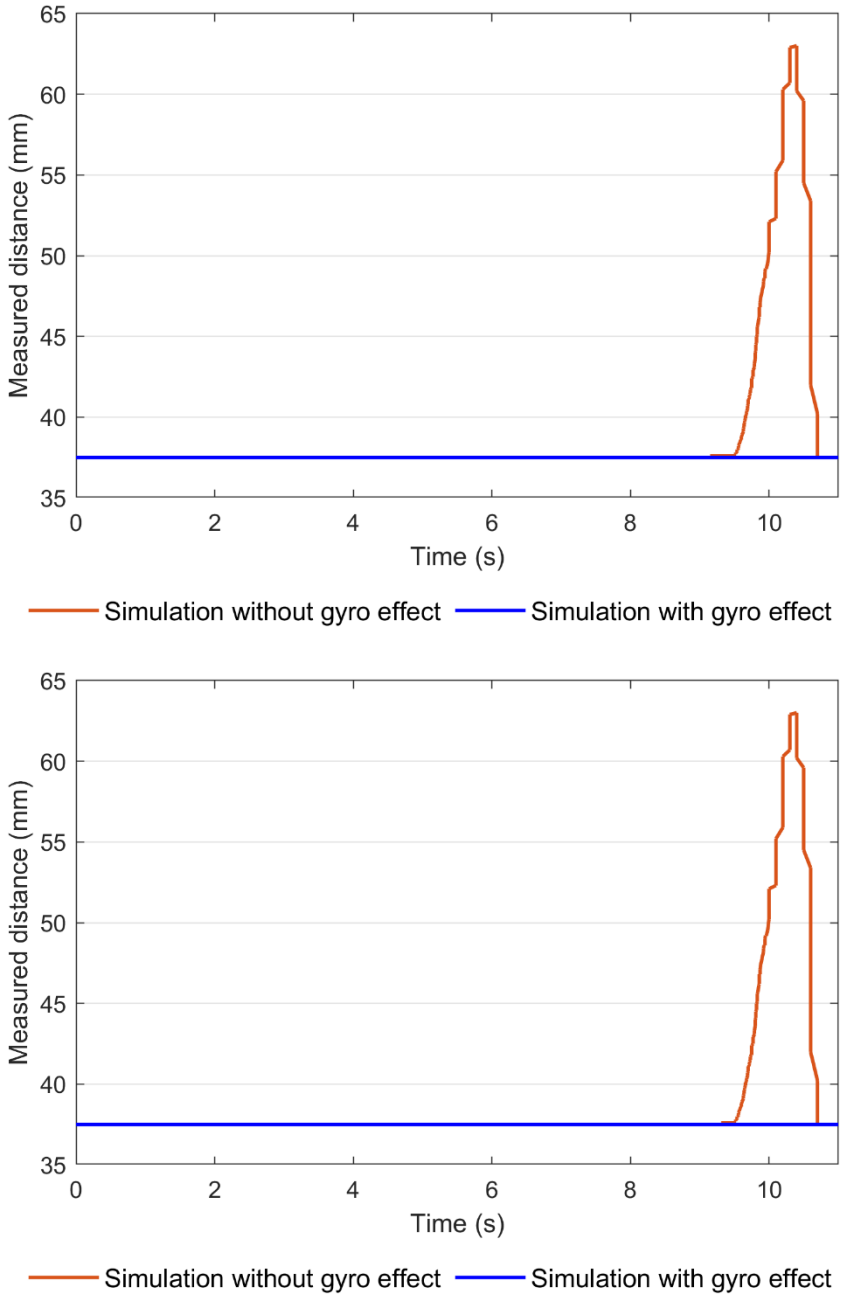


Figure 89 *Results from scenario 6 for the horizontal worst case position (top) and the vertical worst case position (bottom)* [174].

All these scenarios demonstrated that the designed gyro stabilizer can be adopted as a capable stabilization mechanism for the testing system mobile manipulator.

4.5.5 Comparison of the stabilization strategies results

The outcomes of the stabilization strategies that employ linear drives as external stabilization mechanism are summarized in Table 43. With respect to the "conservation of angular momentum" method, the maximum measured contact force between the wheels of the mobile platform and the ground dropped to 12.4 N. Moreover, for the strategy "inclining/tilting", the ground contact force only dropped to 53.1 N, by employing the same braking profile of −2.0 m/s^2. This results in a difference of 40.7 N for the compensation of instabilities.

Using as deceleration profile −2.1 m/s^2, the ground contact force measured during the co-simulations for the strategy "conservation of angular momentum" went down to 3.6 N and can, thus, be defined as unsuitable for this case. On the other hand, co-simulations for the strategy "inclining/tilting" indicated that the ground contact force only dropped to 42.6 N all along the braking process, resulting in a difference of 39.6 N generated force between the two mentioned strategies.

Table 43 *Comparison of the simulations for the stabilization strategies using the linear drives mechanism.*

Stabilization strategy	**Maximum measured ground contact force**	
Max. braking deceleration of mobile platform	−2 m/s^2	−2.1 m/s^2
Inclining/Tilting	53.1 N	42.6 N
Conservation of angular momentum	12.4 N	3.6 N

The main advantage of the strategy "inclining/tilting" is that higher contact forces between the wheels of the mobile platform and the ground were reached in comparison to the strategy based on the "conservation of angular momentum" principle. However, in order to obtain a positive effect for the strategy "inclining/tilting", the linear drives should shift the robot manipulator's COG before the braking process starts. However, thanks to the laser scanners mounted on the employed mobile platform, the stabilization strategy could be able to react prior to a deceleration. If an object was detected in a critical area in front of the mobile manipulator, the robot manipulator would be tilted backwards immediately.

During the stabilization strategy based on the "conservation of angular momentum" principle, the linear drives tilted the robot manipulator in the direction of travel of the mobile platform to compensate on time a braking process. It was possible to react on time depending on the acceleration signal of the mobile platform and, thus, generate the required angular acceleration for the compensation.

For an adequate compensation of instability moments regardless of the strategy, it is important to mention that the braking process of the mobile platform should last no longer than the counteracting angular acceleration applied to the robot manipulator.

Moreover, the described effects suggested the implementation of a gyro stabilizer as further stabilization strategy. Mechatronic co-simulations demonstrated that the gyro mechanism was able to compensate instability states at the right moment when the mobile platform decelerates, explained by its short reaction time and its big impact to the system

stability. In comparison to the “inclining/tilting” strategy, the gyro stabilizer did not depend on a prediction mechanism that prior estimates the mobile platform braking profile.

5 Approach B: Stabilization strategy for mobile manipulators with full access to the robot controller

The stabilization strategies described in Chapter 4 require an additional external mechanism to compensate instabilities, either through linear drives or a gyroscope, since they focus on mobile manipulators that have closed-source operating systems.

The second approach introduced in Chapter 5, uses the mobile manipulator itself as stability actuator, detecting instabilities and performing the countermeasures to prevent tip-over even in changing environments.

An important aspect considered in this approach B, is the fact that the mobile manipulator should independently detect instabilities and independently trigger the countermeasures to prevent the tip over. The strategy to avoid tipping over has to be designed in such a way that the mobile manipulator experiences the least possible loss of time for the overall task. The following sections provide details on the proposed concept, its development and validation employing a robot manipulator testing system.

5.1 Stabilization strategy incorporated in the robot internal control system

The dynamic effects emerged from the robot manipulator and/or mobile platform motions may lead the overall system to become unstable and start to tip over under the following scenarios:

1) Instabilities can occur when the mobile platform is immobile and the robot manipulator is moving. In other words, the target position is approached only by using the robot manipulator.

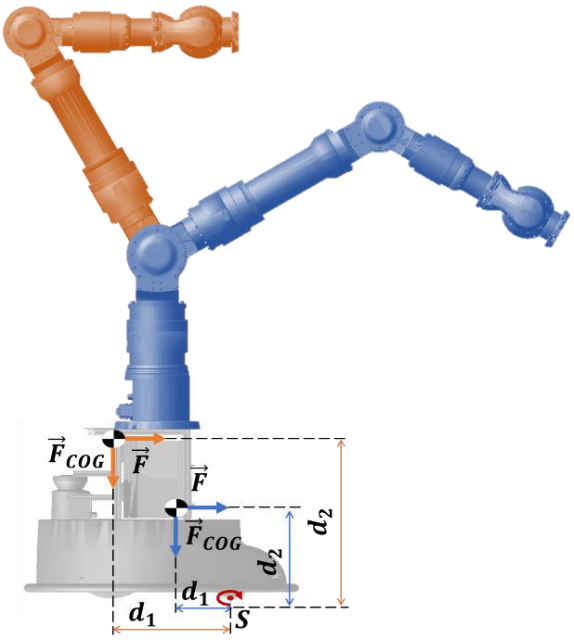


Figure 90 *Tilting moment for mobile manipulator in scenario 1.*

The moment M_S, which is formed around the tilting edge marked by S in Figure 90, results primarily from the location of the total $\vec{F}_{COG}$. Employing distance d_1, the force $\vec{F}_{COG}$, or the total mass m_{total} with the acceleration due to gravity $\vec{g}$, the moment M_S can be calculated as

$$M_S = -\left\|\vec{F}_{COG}\right\| \cdot d_1 = -m_{total} \cdot \|\vec{g}\| \cdot d_1. \tag{93}$$

Changing the robot manipulator's arm position affects the overall center of gravity (COG) and, as a result, the lever arm in relation to the tilt edge S is shifted. In this case, the robot manipulator starts to tip over when the distance to the tilting axis d_1 becomes smaller than zero and, thus, the direction of rotation of $\vec{F}_{COG}$ changes.

2) Instabilities can also occur when the mobile platform and the robot manipulator move simultaneously, e.g., if the Tool Center Point (TCP) is outside the available workspace and the mobile platform receives the signal to move to a predetermined location. If the robot manipulator has assumed an unfavorable position during the braking and acceleration process of the mobile platform, a tip-over also occurs. This is because the acceleration or deceleration results in forces that affect the overall system. However, as shown in Figure 91, in addition to the force $\vec{F}_{COG}$ and the lever arm d_1, the force $\vec{F} = m_{total} \cdot \|\vec{a}\|$ needs to be taken into account.

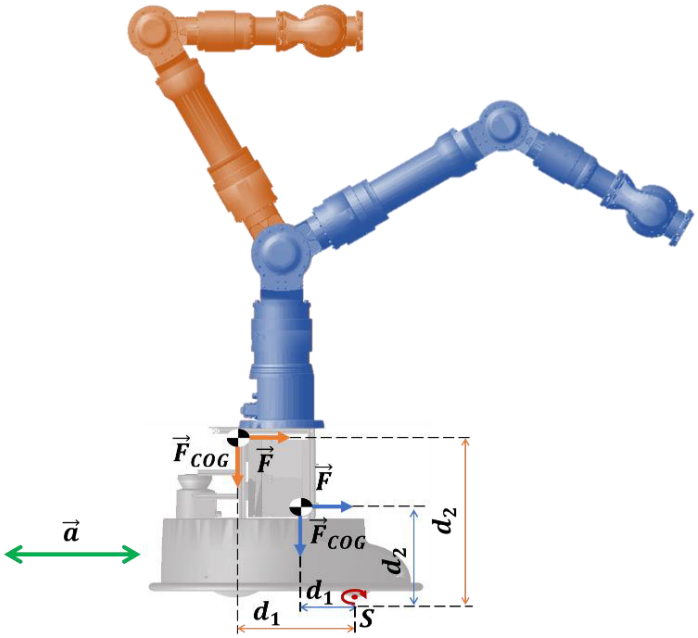


Figure 91 *Tilting moment for mobile manipulator in scenario 2.*

This force is originated from the total mass of the system m_{total} and the applied acceleration $\vec{a}$, and generates a moment with the lever arm d_2

$$M_S = -\left\|\vec{F}_{COG}\right\| \cdot d_1 + \left\|\vec{F}\right\| \cdot d_2 = -m_{total} \cdot \|\vec{g}\| \cdot d_1 + m_{total} \cdot \|\vec{a}\| \cdot d_2. \tag{94}$$

In this case, instabilities can occur if the stand moment M_S describes that $\left\|\vec{F}_{COG}\right\| \cdot d_1$ is smaller than the moment generated by $\left\|\vec{F}\right\| \cdot d_2$.

With focus on increasing $\left\|\vec{F}_{COG}\right\| \cdot d_1$, a mechanical approach (adding mass or extending the footprint of the mobile platform) cannot be pursued as a solution for mobile manipulators because a higher overall mass negatively impacts on battery life and additional supports limit its mobility.

Therefore, for scenario 1 where the mobile platform stands still while the robot manipulator is moving, a concept consisting of an optimization of the theoretical workspace of the robot manipulator is proposed to avoid a tip-over of the mobile manipulator. In addition, the configuration space of the robot manipulator must be adapted so that its joints and links do not move beyond the limits of the optimized working space. For the optimized workspace, it is important to cover a volume as large as possible so that the robot manipulator has maximum room to move.

In scenario 2, the mobile platform moves to a defined position if the given target coordinates for the robot manipulator's TCP is located outside the optimized workspace (from scenario 1). The robot manipulator could either move at the same time or remains in its position while the mobile platform moves through space. Here, a repositioning of the robot manipulator's arm should act as stabilization approach, when an instability is detected. In the proposed algorithm, the robot manipulator retracts into a safe area while maintaining the orientation of the TCP as follows:

1. The algorithm waits for the target position/orientation.
2. After receiving the target point, it is checked whether this point is situated inside or outside the optimized workspace (from scenario 1).
3. If the target point is located within the optimized workspace, the path is calculated directly and the robot manipulator moves to this target position.
4. In case the target point is outside the optimized workspace, the mobile platform moves near to the target point. During the displacement if the system is in a critical state, the repositioning of the arm of the robot manipulator is initialized. The repositioning procedure is to be executed as a loop, until the mobile manipulator exhibits a stable state.
5. As soon as the mobile platform stops, the robot manipulator can move to the target position, which should be located inside the optimized workspace.

The entire concept is represented in a simplified way in the flow chart in Figure 92.

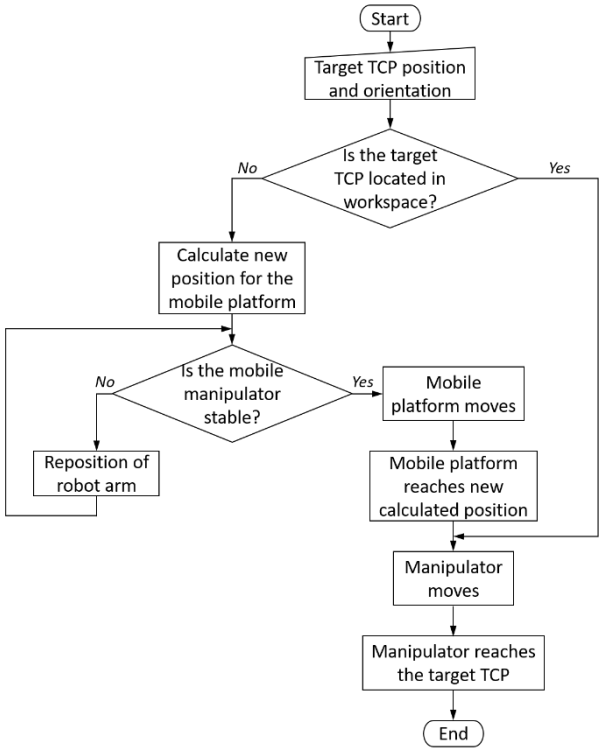


Figure 92 *General description of the algorithm corresponding to the approach B (following* [175]*).*

The central goal of this part of the work is to develop and implement the stabilization strategy shown in Figure 92 based on a ROS framework, and empirically demonstrate its effectiveness on a real autonomous mobile manipulator using the proposed countermeasures against tip-over.

5.2 Methodology: Modeling a close-to-reality system of the mobile manipulator

This second approach is also developed in a simulative environment. But this time, ROS modeling and simulation tools are employed instead of MBS models, since a further interaction between the robot control and path planning is needed. Regardless of this technical distinctness, the model of the mobile manipulator needs to be close-to-reality, in order to ensure that the performance of the proposed strategy is also reflected in the real system.

In comparison with the modeling procedure employed for approach A, the model of the mobile manipulator is not set up regarding its modal criterion, but rather its geometrical and inertial parameters. This assumption is carried out because the system itself is arranged as closed-loop control and, thus, its current states are instantly well-known, i.e. unexpected behaviors can be detected all the time.

In order to analyze the performance of the stabilization strategy under different conditions before it is implemented in the real world, the algorithms are developed and evaluated by means of simulations in ROS environments.

The real mobile manipulator can be represented in the simulation tool Gazebo, which is based on a robot description that visualizes the mobile manipulator in a virtual environment. For this, the ROS package *gazebo_ros_pkgs* provides the necessary interfaces to simulate the mobile manipulator in Gazebo, implementing ROS messages and services in real time with the virtual robot [176]. The mobile manipulator description is contained in the so-called URDF

files (Unified Robot Description Format) as XML-format. The URDF files also includes information about individual components and their geometry, the joints/connections between components and their constraints, the alignment of individual components as well as material, physical and collision properties that support the visualization of each component. For the simulation model of the mobile manipulator, it is very important that the dynamical parameters and general information contained in the URDF-file fits properly the real system regarding the geometry, bodies mass, inertia tensor, COG vector and its kinematic chain.

The geometry of the mobile platform and the robot manipulator as a complete CAD volume model is required for its robot definition. It is crucial to simplify the model while keeping its nominal weight and its theoretical COG. In particular, the dimensions and the material volume that are located far away from axes of rotation, have to be mapped realistically, since these have the greatest influence on the inertia tensor [177]. Small electrical components (circuit boards, cables, etc.) as well as mechanical fastening elements might not be taken into consideration.

All components of the mobile manipulator can be modeled as rigid bodies because deformations during the simulations are not expected. One important aspect to consider during the modeling of the mobile platform is the friction coefficient, which should avoid occasional slipping of the wheels, as in the real world.

Both robot definitions (robot manipulator and mobile platform) need to be assembled into an entire module. In a separate URDF-file, a 6 degrees of freedom (DOF) massless joint has to be attached at the point that represents the spot at which both subsystem are connected to each other. This point, P in Figure 93, helps to represent the components of the forces and moments acting to and deriving from both subsystems, the mobile platform and the robot manipulator.

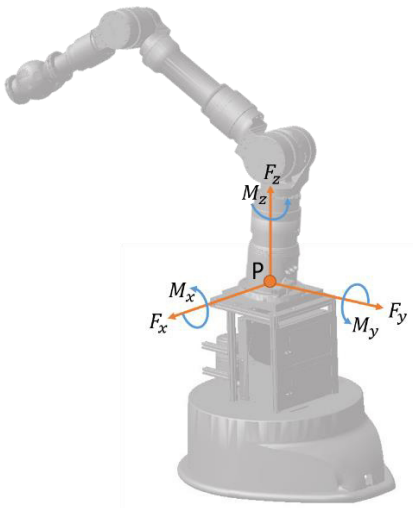


Figure 93 *Representation of 6 DOF massless joint implemented in a URDF-file.*

In addition, since the mobile platform should move relative to the world (global) coordinate system, a virtual planar joint between the world and the mobile manipulator has to be added in MoveIt! to describe this motion constraint.

RViz as 3D representation of the robotic system, MoveIt! (configured to work with ControlIt! [178]) as control tool and Gazebo as virtual world, assist the development of the algorithms in a close-to-reality virtual environment. They provide the same functionalities of a real mobile manipulator, mostly the data transfer for the differential drives (servo motors) and the navigation sensors (e.g., laser scanners [52]).

5.3 Development of stabilization strategy

The first step consists in determining whether the target position of the Tool Center Point (TCP) could be reached just by moving the robot manipulator arm. For this, the algorithm estimates the current TCP position and orientation of the robot manipulator in its world coordinate system (fixed on the mobile platform) and the current position and orientation of the mobile platform, with respect to its dynamic workspace. Then, the robot manipulator's inverse kinematics is computed to verify if the target TCP position can be reached by the robot manipulator links. If so, the robot manipulator is controlled to reach the target.

If not, it means that the target TCP position is located outside the current robot manipulator workspace, and that the mobile platform has to move in the direction of the target TCP. The vectors with respect to the mobile platform driving plane to the target TCP position, which is located outside the workspace, is computed and assigned as displacement set point for the mobile platform. In this case, moving both individual subsystems simultaneously would be the most effective way to reach the target TCP position as fast as possible.

The general principal limitation is that large forces and torques generated by certain configurations/motions of the robot manipulator and/or by accelerations of the mobile platform tend to overturn the whole robotic system. Therefore, no matter which behavior is adopted to reach the target position, the compensation of the generated forces and moments by the stabilization strategy is indispensable.

5.3.1 Building a simulation setup in ROS environment

Section 2.4 provided a brief overview of the basic concepts of the Robot Operating System (ROS), which facilitates the understanding of the paragraphs below.

To operate in the simulation environment, Gazebo has to be started and immediately be followed by RViz via MoveIt!. Afterwards, the communication link between Gazebo and MoveIt! has to be established. Then, the following steps need to be executed:

1. Load a predefined map for the world space in Gazebo and RViz.
2. Define the TCP target position in the space.
3. If required, the mobile platform moves towards the TCP target position using the ROS package *move_scitos*.
4. The robot manipulator is then approached to the TCP target position by MoveIt! using the ROS package *move_arm*.

Both ROS packages, *move_scitos* for the motion of the mobile platform and *move_arm* for the motion of the robot manipulator, are included into a superordinate package called *tcp_goal*. This logic is enclosed in the main function that incorporates the data about the current position and orientation of the mobile manipulator. Based on the target TCP position, the algorithm establishes autonomously, if the target position is located inside the workspace of the robot manipulator and, thus, can be reached without needing to move the mobile platform. On the other hand, it calculates how much the mobile platform must be relocated in space. Hence, depending on the current and the target TCP position/orientation, the main program calls the corresponding functions to move the mobile platform, if necessary. Then, the main program waits until the target position/orientation for the mobile platform has been reached. Once it happens, the main program calls the function to move the robot manipulator, thus reaching the end target TCP position.

The computer unit must be able to manage all processes in parallel as a multitasking system. To fulfil this requirement, the Whole Body Operational Space Control (WBOSC)[178] is implemented. This algorithm allows to control multiple tasks of the mobile manipulator at the same time by defining restrictions and particular priorities to each task. WBOSC is also implemented in ROS via ControlIt! It simultaneously gets the information sent by the sensors from each rotational joint and computes the motion path.

The main problem of the described motion logic is that the tilting stability of the mobile manipulator is not guaranteed for all points located in the robot manipulator theoretical workspace and for scenarios when the mobile platform moves. To avoid human beings and machines dangerous situations, algorithms for the autonomous tilting detection and avoidance have been developed.

5.3.2 Stabilization strategy

The entire algorithm for the stabilization strategy is divided into two main parts: The first section addresses the problematic regarding the tip-over detection and then, the second part, examines the tip-over avoidance query.

5.3.2.1 Tip-over detection algorithm

Generally speaking, a mobile manipulator tends to tip over its tip-over axis. A tip-over axis is described by the outer edge of its footprint. Depending on the shape of the footprint, the mobile manipulator could tip over the different axes that correspond to the outer edge of its footprint.

The tilting shape of a mobile manipulator is defined by linking the connecting lines of the contact points between the wheels of the mobile platform and the ground. Thus, for a mobile platform with three wheels, the triangle sides constitute the tilting stability axes for the mobile manipulator. Each of these axes is represented with a unit vector, as shown in Figure 94. Their direction corresponds to the shape of a closed chain.

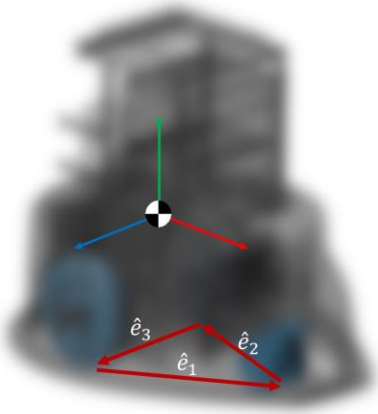


Figure 94 *Tilting shape and mobile platform COG (following* [95]*).*

The technique Force Angle stability measure (FA), seen in Section 3.2, served as a starting point for the implementation of the tip-over detection algorithm. Since the position of the mobile manipulator's COG changes depending on the configuration of the robot manipulator, the COG of the overall system has to be continuously calculated by the ROS package *robot_kinetics_pkgs* [179]. In order to detect if the mobile manipulator is prone to tip-over, the

algorithm computes all forces acting on the robot manipulator at any point of time. The resulting vector intersects the plane on which the tilting shape lies. Then, the distance between the projection of the intersection point and the tilting axes are determined. If the intersection point is situated outside the tilting shape (Figure 94), the mobile manipulator has a risk to tip-over.

The sensitivity of this algorithm can be adjusted by increasing or decreasing the offset that describes the distance between the COG and the footprint, as shown in Figure 95. The greater the offset, the more sensitive the tip-over detection because the resulting force vector points outside the footprint earlier.

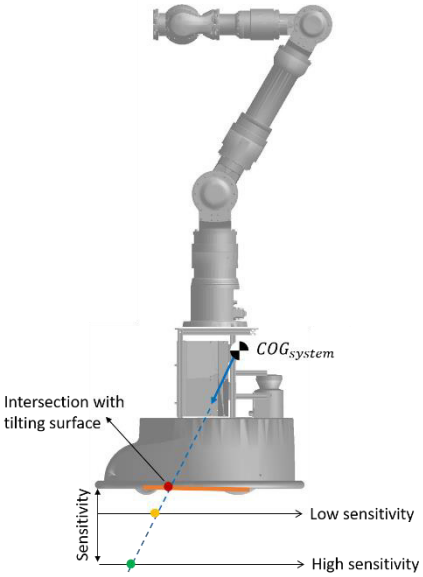


Figure 95 *FA point of intersection (red) between resulting force and footprint (following* [148]*).*

The Force-Angle stability measure monitors the angle between the resulting force and the mobile manipulator tip-over axes, but only examines magnitude and direction of the total acceleration measured by sensors located on the mobile platform. Additionally, the stability state is represented as a binary value, distinguishing between stable and unstable, but no statement is made about the degree of risk.

Consequently, this approach is enriched by employing the Moment Height Stability (MHS) method, introduced in Section 3.2, which not only considers the forces measured by Inertial Measurement Units (IMU) situated on the mobile platform, but also contemplates the internal forces emerged during the robot manipulator's motion.

In order to implement the MHS technique, the mobile manipulator model (including mass, COG and inertia tensor) in combination with the motions of the robot manipulator (position, velocity and acceleration of all joints) is employed to determine the inverse dynamics of the system by applying the recursive Newton-Euler algorithm (RNEA) of R. Featherstone [55], seen in Section 2.5. The inverse dynamics is the foundation of the dynamic tip-over detection using the MHS technique, since its components represent how the robot manipulator affects the mobile platform in terms of the forces (including inertial) and torques generated only by the robot manipulator's motion itself. The RNEA algorithm adopted to solve the inverse dynamics in this study was already implemented by R. Smits [180,181] in ROS, based on the R. Featherstone statements.

To this effect, the ROS topic *joint_states* provides the current joint positions and joint angular velocities of the robot manipulator. The joint angular accelerations are not directly

available from the topic, but they are implemented as an extra ROS node, which continuously calculates them by using the backward-difference approximation (only the current and last value for the angular velocity are available). The topic *joint_state_acceleration* publishes then all the information regarding each joint required for the inverse dynamics.

As a first step, the velocity and acceleration of all robot manipulator's links, beginning from the basis to the end effector (TCP), as well as the emerging forces and torques that induced those accelerations are estimated. Subsequent, all forces and torques transferred from the parent link to the child link across the kinematic chain are also computed by combining equations for the linear and rotational motion of rigid bodies given by Newton and Euler [55]. Besides, an IMU has to deliver information about the 3D-linear acceleration of the moving mobile platform and the vector of the gravitational acceleration regarding the mobile platform, since the acceleration/deceleration generated during the starting off/braking process is crucial for the estimation of a stability value.

Based on the outcomes of the inverse dynamics (RNEA) as well as the measurements from the IMU and the location of the COG related to the mobile coordinate system, the dynamic stability can be estimated by means of the MHS method following these steps:

1. The system has to be first split into the two parts that are physically connected by the point P, as represented in Figure 96: the mobile platform and the robot manipulator. The components of $\vec{F}$ and $\vec{M}$ represent how the robot manipulator affects the mobile platform when the robot manipulator moves.

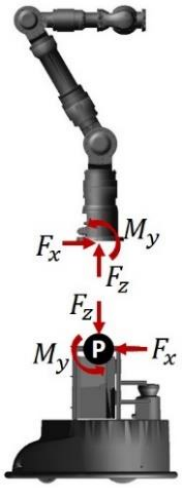


Figure 96 *Connection point P of the mobile platform with the robot manipulator* [148].

2. The acceleration values acting on point P has to be estimated and represented in the coordinate system of the mobile platform.
3. The joint forces acting on point P are also computed by means of inverse dynamics (RNEA) and correspondingly projected to the mobile platform coordinate system. Additionally, all inertial forces caused by the linear acceleration of the mobile platform acting on its COG has to be considered into the algorithm.
4. The tilting shape is defined, as in the FA method.
5. From the sum of the forces affecting the system, a total moment $M_{i,T}$ is calculated for each vertex of tilting shape by the cross product between the resultant force F_M generated by the robot manipulator and the (inverted) connection vectors of the tilting shape $\vec{e}_{side}$, adding the resulting moments of the articulations M_d. Figure 97 assists the formulation of the corresponding equations.

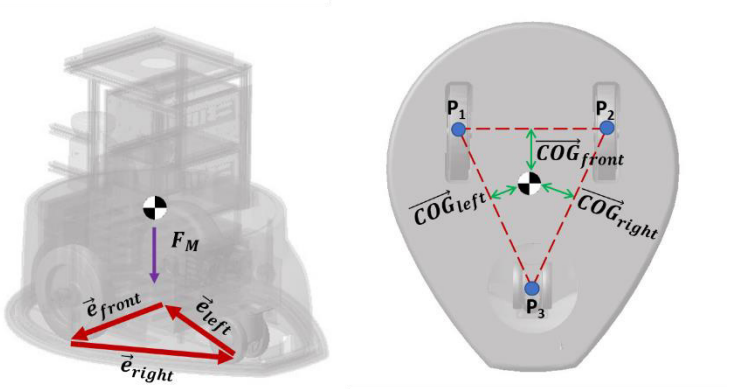


Figure 97 *Vectors for the projected torque calculation.*

$$\vec{M}_{1,T} = (\vec{e}_{right} \times \vec{F}_M) + \vec{M}_d \tag{95}$$

$$\vec{M}_{2,T} = (\vec{e}_{left} \times \vec{F}_M) + \vec{M}_d \tag{96}$$

$$\vec{M}_{3,T} = (\vec{e}_{front} \times \vec{F}_M) + \vec{M}_d. \tag{97}$$

In addition, all moments acting on the COG of the mobile platform have to be estimated by the cross product between the force and the corresponding lever arm as follows

$$\vec{M}_{1,Basis} = (\overrightarrow{COG}_{right} \times \vec{F}_{Base}) \tag{98}$$

$$\vec{M}_{2,Basis} = (\overrightarrow{COG}_{left} \times \vec{F}_{Base}) \tag{99}$$

$$\vec{M}_{3,Basis} = (\overrightarrow{COG}_{front} \times \vec{F}_{Base}). \tag{100}$$

The total moment regarding each tilting edge is calculated as the addition of both moments:

$$\vec{M}_{v1} = \vec{M}_{1,T} + \vec{M}_{1,Basis} \tag{101}$$

$$\vec{M}_{v2} = \vec{M}_{2,T} + \vec{M}_{2,Basis} \tag{102}$$

$$\vec{M}_{v3} = \vec{M}_{3,T} + \vec{M}_{3,Basis}. \tag{103}$$

6. These resulting moments $\vec{M}_{vi}$ have to be projected on the respective tilting axis by means of scalar product between them and the vectors of the tilting shape. Given $\overrightarrow{p_1}, \overrightarrow{p_2}$ ... $\overrightarrow{p_n}$ as contact points of the mobile platform with the ground (wheels), the unit vectors can be calculated as follows

$$\vec{M}_i = \vec{M}_{vi} \cdot \hat{e_i} \tag{104}$$

with

$$\hat{e_i} = \frac{\vec{p}_{i+1} - \vec{p}_i}{\|\vec{p}_{i+1} - \vec{p}_i\|}. \tag{105}$$

The resultant moment $\vec{M}_i$ comprises all forces and moments acting on P (caused by the robot manipulator) as well as all forces and moments acting on the COG of the mobile platform (caused by inertias, linear accelerations/decelerations from starting off and braking process, gravitational forces, etc.).

7. Considering that positive moments faced inside the unit vectors of the tilting shape, positive moments mean that the sum of all stability moments regarding an edge is bigger than the sum of all the tilting moments. The dynamical stability value α_i relating to the same i edge, for which the moment is calculated, can be then computed with help of the mass moment of inertia of the mobile manipulator regarding the i edge, I_{vi}.

$$\alpha_i = (I_{vi})^{\sigma_i} \cdot M_i \tag{106}$$

where

$$\sigma_i = \begin{cases} +1 \; for \; M_i > 0 \\ -1 \; otherwise. \end{cases} \tag{107}$$

The MHS coefficient is defined as the critical value (smallest dynamical stability value for α) regarding an edge i of the tilting shape. In other words, the smallest value of α_i regarding an edge represents the edge with the most critical stability state. $\alpha = \min(\alpha_i)$ can thus be interpreted as:

α>0 System is stable
α=0 System is critically stable
α<0 System tends to tip over edge i

α_i reacts very sensitively to the height of the system COG, i.e. under the same conditions, a higher position of the COG leads to an increased level of vulnerability for the system stability against tilting over. Therefore, the MHS equation is complemented as follows

$$\alpha_{i-cm} = (h_{COG})^{\lambda} \cdot \min(\alpha_i) \tag{108}$$

being

$$\lambda = \begin{cases} -1 \; for \; \min(\alpha_i) > 0 \\ +1 \qquad otherwise. \end{cases} \tag{109}$$

The continuous computation of the stability value α_{i-cm} during the mobile manipulator operation allows the detection of tip over risks and its critical tilting edge. The stability value based on the MHS method is the basis for triggering the tilting avoidance countermeasures.

5.3.2.2 Tip-over avoidance algorithm

Once the tip-over detection algorithm is able to identify a possible overturn of the mobile manipulator, it is decisive to actively react against these instabilities by means of a tip-over avoidance algorithm.

A reduction of the traveling speed of the mobile platform can contribute to prevent a system tip-over, however, it involves significant cycle time loss. Shifting the system COG, whereby the robot manipulator takes another configuration/position, is probably the best and most efficient method to avoid a tip-over of the mobile manipulator. Therefore, in case unstable states are detected while the mobile platform is moving, a repositioning of the robot manipulator's arm takes place.

A prerequisite for tip-over avoidance procedure is that the initial orientation of the robot manipulator's TCP remains unchanged during the repositioning process.

5.3.2.2.1 Workspace optimization

The reachable workspace of a robot manipulator describes all points in the space that its end-effector can reach in at least one of its orientations within its mechanical configurations.

On the other hand, the dexterous workspace is the subset of the reachable space in which the end-effector is able to reach the points with all its possible orientations [182–184].

The workspace of a stationary robot manipulator is perfectly known and delimited by its own kinematics. In contrast, the workspace of a mobile manipulator is infinitely large, variable and restricted by walls and other barriers. Mobile manipulators have a global and a local workspace: the local workspace is situated directly around the robot manipulator and describes all points that can be reached by its end-effector without having to move the mobile platform; the global workspace is characterized by the space the end-effector can reach when the mobile platform also moves [183,185].

Besides that, the configuration space (so-called C-Space) contains all possible poses restricted by the joint properties, in which each of its single point defines a unique configuration of the robot manipulator kinematic chain [183]. The dimension of the configuration space corresponds to the minimum number of parameters needed to specify a pose of the robot manipulator (its degree of freedom [186]). Moreover, all joint positions that are impermissible within the configuration space are represented by the collision (obstacle) space C_{obs}. The so-called free space is, then, the difference of the collision space and the configuration space.

The proposed stabilization algorithm implements all these workspaces in ROS as follows:

- The reachable workspace of the robot manipulator has to be stored in the general robot description.
- The robot manipulator´s configuration space, including its permissible joint range, are defined as variable on the robot description. The joint movements can be set up using the MoveIt! wizard.
- The configuration space has to be constrained with collision matrixes, so that no collisions with the own robot manipulator's links occur [187].
- Obstacles in the workspace are included to simulate the items that normally exist in a real environment. They are taken into account during the motion planning [188].
- Based on the defined configuration and collision space, a collision-free path is created. A comprehensive selection of algorithms for the motion planning can move the joints within this free space employing MoveIt! Because of its high success rate and its fast computation, the solver RRT (rapidly-exploring random tree) is implemented to search for a collision-free path within the configuration space in this approach [189, 190].

In order to implement the workspace optimization (illustrated with the green area in Figure 98-b), the entire theoretical workspace (illustrated with the orange area in Figure 98-a) is divided into two parts: a critical and a non-critical workspace.

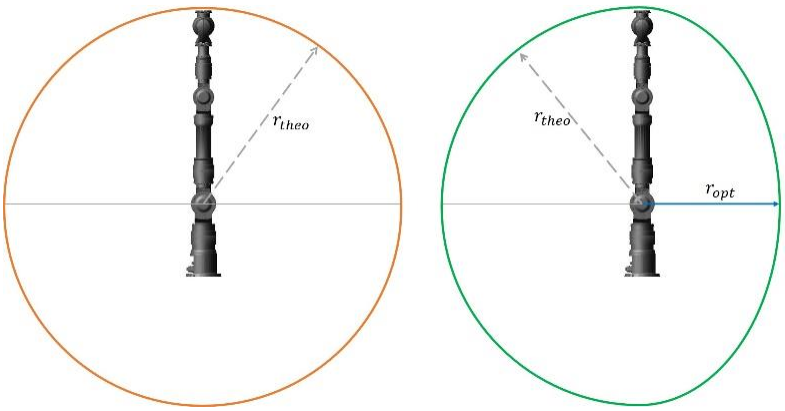


Figure 98 *a) Theoretical and b) Non-critical workspace optimization (following* [175]*).*

1. The non-critical area is described by a hemisphere with the original theoretical workspace r_{theo} as its maximum radius, and
2. The critical workspace is proposed as an ellipsoid with maximum focal point r_{opt}. The ellipsoid is chosen because its geometric shape builds a sphere very closely, resulting in minimal modifications to the original workspace and minimal reductions regarding its volume.

The non-critical area is represented with the equation for ellipsoids:

$$\frac{(x - x_0(t))^2}{a^2} + \frac{(y - y_0(t))^2}{b^2} + \frac{(z - z_0(t))^2}{c^2} = 1. \qquad (110)$$

Considering that the rotary axis of the front wheels of the mobile manipulator coincides with one of its tilting shapes (see Figure 97), the risk of tip over the front two wheels tends to be much higher in comparison with both lateral wheels. Consequently, the optimized radius, r_{opt}, can be considered for the a-direction in the Eq. *(111)*, and the entire theoretical radius, r_{theo}, can be used for the b- and c-directions. The generally valid equation is reformulated as follows

$$x, y, z, r \in \mathbb{R}^3 \left| \frac{(x - x_0(t))^2}{r_{opt}^2} + \frac{(y - y_0(t))^2}{{r_{theo}}^2} + \frac{(z - z_0(t))^2}{{r_{theo}}^2} \right| \leq 1. \qquad (111)$$

Since the optimized workspace must define the volume at which the mobile manipulator adopts all positions without making it tip over, the tilting moment M_{static} with respect to point A shown in Figure 99 is employed to define the dimensions of the ellipsoid.

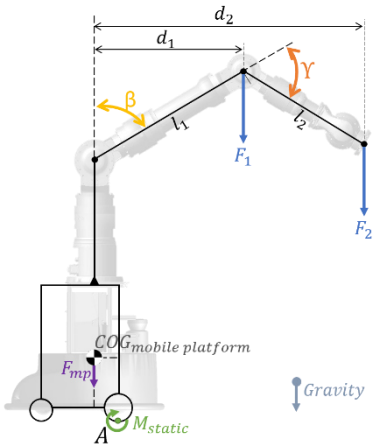


Figure 99 *Free body diagram of mobile manipulator (following* [175]*).*

$$M_{\text{static}} = \sum_{i=0}^{n} F_i \cdot d_i. \tag{112}$$

Nearly all robot manipulators possess 6 or 7 articulated joints. It can be generalized that from the three main joints and their corresponding links of robot manipulators (J1 or waist joint, J2 or shoulder joint and J3 or elbow joint), the positions that mainly affect the tilting moment M_{static} are characterized by its second and third link: in Figure 99, the lever arm of forces F_1 and F_2 to the center of rotation A changes depending on angles β and ϒ.

Thus, the moment generated by force F_1 is determined by changes in the angle β of the robot manipulator. Likewise, the moment produced by force F_2 is determined by both, β and ϒ, as follows:

For F_1,

$$d_1 = \mathrm{l}_1 \cdot \sin(\beta) \tag{113}$$

For F_2,

$$d_2 = \mathrm{l}_1 \cdot \sin(\beta) + \mathrm{l}_2 \cdot \sin(\beta + \Upsilon). \tag{114}$$

Considering the whole angle interval of β for the first link (l_1) and only limiting the angle interval ϒ for the second link (l_2) implies that the robot manipulator would be able to adopt the greatest possible arch for reaching the target point, without suffering any instabilities.

Once the optimized workspace is defined, it is also necessary to determine whether the robot manipulator's TCP is located within the pre-defined critical volume (in front of the robot manipulator) or within the non-critical volume (behind the robot manipulator– this area is less critical, as the robot manipulator is further away from the edge it might tip over). For that purpose, the vector between the mobile platform and the robot manipulator's end-effector $\overrightarrow{TCP}$ has to be calculated and then transformed into the coordinate system of the mobile platform.

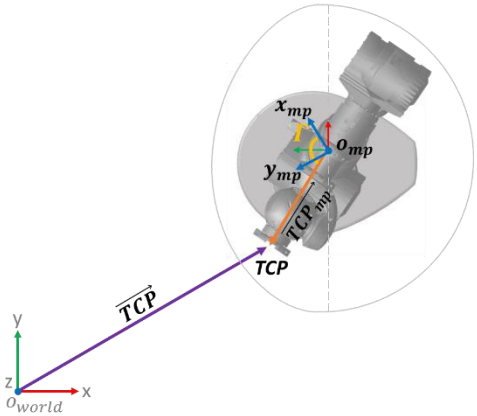


Figure 100 *The angle Γ defines if the TCP is located within the critical volume (following* [175]*).*

As shown in Figure 100, the scalar product between the direction vector $\vec{x}_{\text{mp}}$ and vector $\overrightarrow{\text{TCP}}_{\text{mp}}$ is used to infer the value of angle Γ as follows

$$\Gamma = \cos^{-1}\left(\frac{\overrightarrow{TCP}_{\text{mp}} \cdot \vec{x}_{\text{mp}}}{\left\|\overrightarrow{TCP}_{\text{mp}}\right\|}\right). \tag{115}$$

This angle Γ provides information about the position of the TCP relative to the mobile platform. This way, it can be distinguished whether the equation for the hemisphere or for the ellipsoid should be used for the workspace during the operation of the mobile manipulator. In general, if:

- Γ <90°, the TCP is located within the critical volume and, thus, the ellipsoid should be employed as the permitted workspace for the motion planning.
- Γ >90°, the TCP is located within the non-critical volume and, thus, the hemisphere defined by the theoretical local workspace should be employed for the motion planning.

By means of this function, the limits of the reachable working space is also implemented in ROS: the transformation between the map and the mobile platform is first retrieved. This can be used to determine the vector between the mobile platform and the TCP, which is then transformed into the coordinate system of the mobile platform. The subsequent calculation checks whether the TCP lays in front of the robot manipulator, in the critical volume, or behind it, in the non-critical volume. The function can then distinguish whether the spherical equation or the equation of the ellipsoid is being used.

5.3.2.2.2 Repositioning of robot manipulator

The robot manipulator and the mobile platform moving simultaneously is the second relevant scenario that has to be examined. There, dynamic forces and moments are generated by the robot manipulator joints and by the mobile platform motion itself. Therefore, as soon as an instability is detected by the algorithm presented in Section 5.3.2.1, a further algorithm should actively shift the system COG in such a way that the value from MHS exhibits more stability, in other words, α_{i-cm} increases its value.

The effortless way to implement it would be the reposition of the arm of the robot manipulator into a fixed home/safe posture before the platform starts to move. However, this method involves significant loss of time for the original task to be performed: even if the TCP is only just outside the non-critical volume, the robot manipulator has to execute the complete trajectory to the home/safe position before the mobile platform starts moving. Therefore, a prerequisite for the stabilization strategy presented in this approach is that the robot manipulator moves as little as possible during the reposition, in order not to unnecessarily deviate it from its original path.

A similar analysis to that in Section 5.3.2.2.1 is carried out to determine the dynamic critical torque $M_{dyn,}$ using the schema in Figure 101.

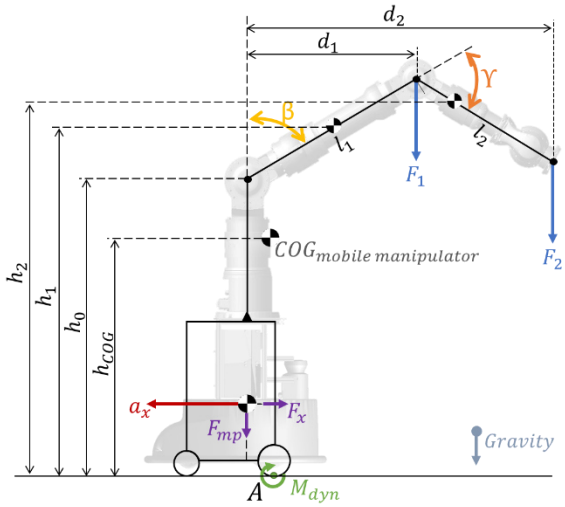


Figure 101 *Free body diagram of mobile manipulator considering dynamic forces and moments generated by operation movements (following* [175]*).*

$$M_{\mathrm{dyn}} = \sum_{i=0}^{n} F_i \cdot d_i - \|\vec{a}\| \cdot \mathrm{m}_{\mathrm{total}} \cdot h_{\mathrm{COG}}. \tag{116}$$

The height of the COG of the individual components, h_i, depends on the angles β and Υ as

$$h_{i=1} = h_0 + \cos(\beta) \cdot h_{1-JointCOG} \tag{117}$$

where h_0 represents a constant, corresponding to the height up to the first joint, and the value $h_{1-JointCOG}$ describes the height of the COG for the first link, with respect to its own coordinate system located in its pivot point. The height of the COG for the second member is, then, calculated with help of both angles (β and Υ) as well as with the length l_1 of the first link as follows

$$h_{i=2} = h_0 + l_1 \cdot \cos(\beta) + h_{2-JointCOG} \cdot \cos(\beta + \Upsilon). \tag{118}$$

Changes in the dynamic tilting stability, α, can be studied in relation to the robot manipulator joint variables, q_i, which describe the relative rotational motion between its contiguous links. They characterize the coordinates of the spaces in which the location of all the links of the robot manipulator are represented [191].

Through the gradient method, it is possible to find out in which direction in space the vector attached to the TCP acquires higher tilting stability by applying a fixed increment $\Delta\pm q_i$. This increment is added to and subtracted from each current joint position q_i of the robot manipulator iteratively until the termination criterion, the dynamic tilting stability value computed by the MHS method, is fulfilled.

A spatial discretization is carried out in order to simplify the approximation of the gradient method. The revolute joints q_1 to q_4 of the robot manipulator are the only parameters used for the discretization because they produce the greatest influence on the tilting stability of the entire system. The tilting stability value remains almost unchanged if the joint variables q_5, q_6 and, eventually, q_7 varies, since they have about the same coordinate origin.

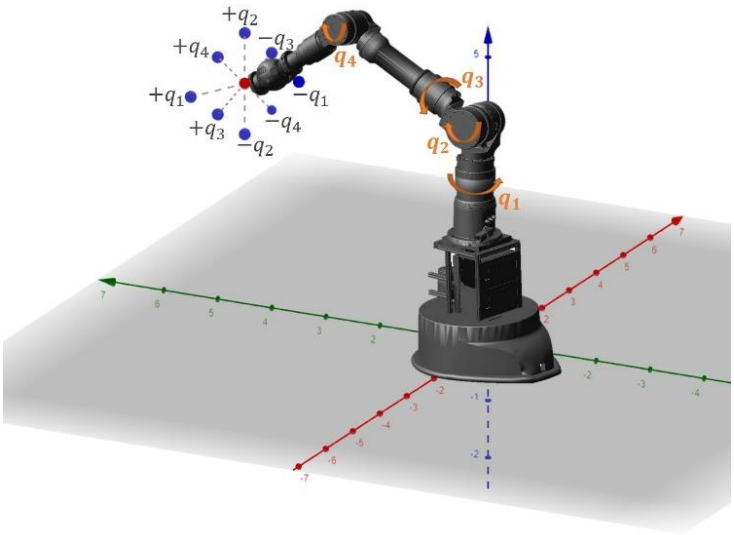


Figure 102 *Discretization of workspace for the implementation of the gradient function required for the repositioning of the robot manipulator's joints (following* [175]*).*

The red point illustrated in Figure 102 represents the current position of the robot manipulator´s TCP. Based on this configuration, when the tip over detection algorithm identifies an instability, i.e. the stability value drops lower than the predefined critical value, the tip over avoidance algorithm calculates a new tilting stability value, $\alpha_{\Delta\pm q}$, for the joint configuration that corresponds to each of the blue points. If the maximum value determined for this new tilting stability value, $\alpha_{\Delta\pm q}$, increases above the defined threshold value, $\alpha_{\mathrm{cm-critical}}$, then the new joint position is sent to the motion control and the robot manipulator repositions its joint, thus, improving the system stability state. Otherwise, if the calculated $\alpha_{\Delta\pm q}$ does not fully overshoot the threshold value $\alpha_{\mathrm{cm-critical}}$, the algorithm iteratively calculates a new tilting stability value $\alpha_{\Delta\pm q}$ until it fulfills the stability criterion. Using this technique, it can be ensured that the robot manipulator is repositioned as much as necessary. This algorithm is illustrated in Figure 103.

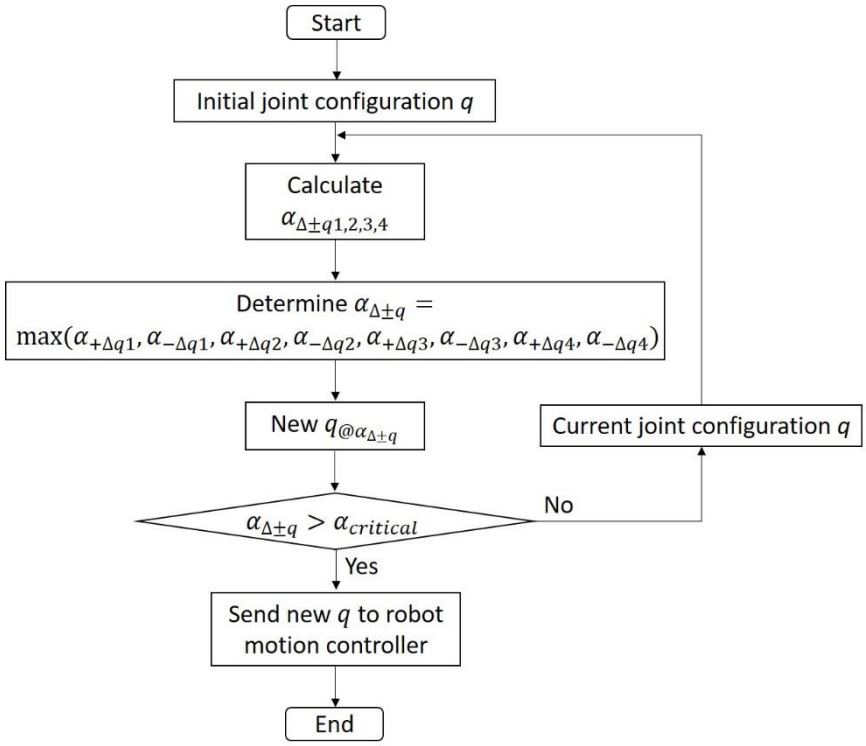


Figure 103 *Algorithm for calculating the new joint position for robot manipulator (following* [175]*).*

The developed algorithm of the stabilization strategy for mobile manipulators is also implemented in the ROS model to cover the "tilting detection" and "tilting avoidance" functionalities, as illustrated in Figure 104. It determines a new TCP position for the robot manipulator that implies higher tilting stability and returns it to the main program. Then, the control unit relocates the corresponding joints within the collision-free optimized workspace, specifying the target orientation as the current orientation of the TCP.

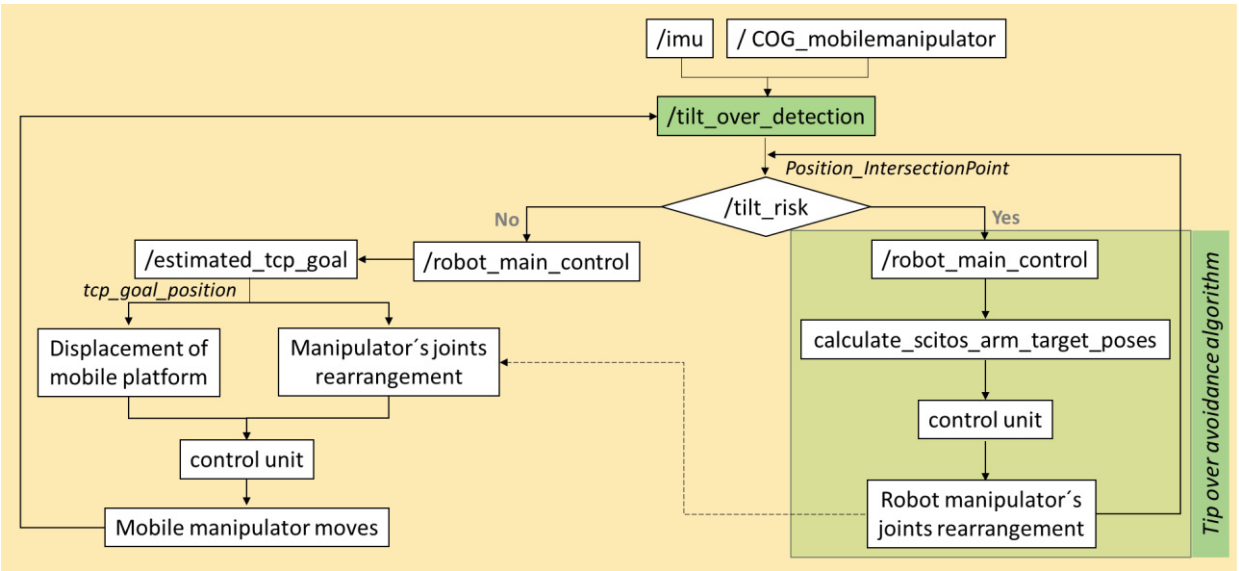


Figure 104 *Mobile manipulator control functionalities.*

Before the mobile manipulator starts moving, the */tilt_over_detection* node subscribes the topic of the IMU measurements, to get the data provided by the inertial sensors, and the topic COG, to get the position and orientation of the mobile manipulator's COG. Once the required

information is received, the same node */tilt_over_detection* estimates the vector *Position_IntersectionPoint* for all tilting edges and, consequently, publishes this information in the topic */tilt_risk*. If the *Position_IntersectionPoint* delivers a positive value, the calculated force points outside the overturning shape and the mobile manipulator is prone to tip over. Following this, the node */robot_main_control* subscribes the information contained in the topic */tilt_risk* and controls the robot manipulator to move to a safety position within the predefined optimized workspace, employing the gradient approach. After the mobile manipulator is considered as stable, the topic */estimated_tcp_goal* sets and publishes how to reach the desired *tcp_goal_position*, either only performing a rearrangement of the robot manipulator joints or including the displacement of mobile platform to another point in the space. After the path planning is completed, the target position and orientation for each robotic subsystem are posted iteratively to the control unit and the mobile manipulator starts moving to the goal position. Meanwhile, the */tilt_over_detection* checks the current stability value α_{i-cm} continuously: as soon as α_{i-cm} drops below the predefined critical value, the node */robot_main_control* starts the repositioning, since the whole system is unstable. Once the system exhibits a stable state, the */estimated_tcp_goal* adopts temporary the new position (after repositioning) in the path planning as target position for the robot manipulator until the mobile platform reaches its target position, so as to not affect the stability state achieved.

5.4 Implementation of stabilization strategy employing a testing system

Since this second approach demands the customization of the control system of the mobile manipulators and due to the fact that the standard industrial robot manipulator used as testing system in approach A (see Section 4.5) does not allow the alteration of its control system, an additional testing system was required for the implementation of the stabilization strategy.

The new adopted testing system is operated by a *higher* robot manipulator and by an even more compact mobile platform that, together, offer high agility and maneuverability due to its compact design, however, the addressed stability problem has been observed during some experimental attempts: large forces and moments generated by certain configurations and/or motions of the robot manipulator tended to tip the mobile manipulator over.

The testing system is comprised by the mobile platform Scitos G5 by MetraLabs and the 7-DOF lightweight robot manipulator LWA 4D by Schunk, illustrated in Figure 105. Both subsystems are equipped with open-source middlewares, which enable an easy and fast implementation of algorithms.

Figure 105 *Scitos G5 and LWA 4D as testing system for the developed approch B.*

The mobile platform uses differential drives to get moving. Especially for such mobile platforms, the direct contact between the ground and its drive wheels is essential; otherwise, if

the drive wheels raise off the ground, the theoretical current location of the TCP in the real world space will be incorrect because it will be calculated using the wrong parameters for its position and orientation.

The mobile platform's battery powers the robot manipulator through a DC-DC converter. This makes the mobile manipulator independent of wired connections. It must be noted that the mobile platform is controllable with the middleware for robotic applications MIRA [192], whereas the 7-DOF robot manipulator is controlled by the robot operating system ROS [193]. For the system integration of the two different middlewares, the ROS - MIRA bridge of the STRANDS project [194] was implemented. This made the whole robot system controllable via ROS, required for the execution of the stabilization strategy.

Although the presented approach was implemented through simulations, the goal is to validate the stabilization strategy in the real system once the results from the simulation sets are shown as satisfactory. For the assembly of both subsystems, the following mechanical, electrical and information technology integration were required:

- Mechanical integration. The robot manipulator was mounted centered on the upper plate of the mobile platform to ensure an equivalent handling workspace in all directions. This way, the robot manipulator is able to handle the same payload under the same circumstances in all directions of its world coordinate system, located center-aligned of the mobile platform.
- Electrical integration. Both, the mobile platform and the robot manipulator, operate on 24 VDC. The robot manipulator required 5 A for continuous operation or rather a maximum of 15 A for peak load operation [195]. Theoretically, the power consumption of both subsystems could be supplied by the built-in battery of the mobile platform [196]. The main challenge by adopting this solution was the variations on the battery power supply within the rated voltage depending on its state of charge (29 VDC while charging, 28 VDC at full charge and 22 VDC when almost empty). The servomotors of the robot manipulator were designed for being operated at 24 ±5% VDC (25.2 VDC – 22.8 VDC). If the robot manipulator was powered with a higher VDC, its components would be damaged. To meet this requirement, a DC-DC converter had been integrated into the electric circuit to power the robot manipulator with constant 24 VDC and to provide the required 6.2 A nominal current (5 A for its motors and 1.2 A for its control unit).
- Information technology integration. As mentioned above, due to the fact that the mobile platform was controlled by the middleware MIRA [197] and the 7-DOF robot manipulator was controlled by the middleware ROS, a software/firmware integration was needed in addition to the mechanical and the electrical integration. The mobile platform manufacturer does not offer any possibility to achieve a communication between ROS-based robot controllers with the mobile platform. Hence, both robotics systems should work based on MIRA or ROS, and one of the systems had to be adapted to work based on the middleware available by the other system. Considering the easiness ROS involves, an interface between MIRA and ROS enabled them to collaborate in a ROS environment: the mobile platform motion planning was calculated by MIRA but its execution was performed by ROS, acting as a single system for the entire mobile manipulator. For the integration, the Spation-Temporal Representation and Activities for Cognitive Control in Long-Term Scenarios (STRANDS [50]) was implemented as interface between MIRA and ROS. The STRANDS project contains the packages *scitos_drivers*, *scitos_common* and *sick300*. The actuation of the robot manipulator joints are received from the computational unit located

at the mobile platform to the servomotors via CANopen. Consequently, the integrated mobile manipulator can be controlled entirely through ROS.

5.4.1 Modeling a close-to-reality mobile manipulator testing system

The model of the testing system mobile manipulator was composed of the description of the robot manipulator and the description of the mobile platform. The robot manipulator was prepared according to the already available software package developed by Fraunhofer IPA [176,198,199], as a part of various cooperative research projects with the robot manufacturer. The subfolder package *schunk_modular_robotics/schunk_description/meshes* included the data related to the links and joints geometry of the robot manipulator in form of STL (stereo lithography CAD for 3D systems) mesh files.

The robot definition available from the IPA project [176,198,199] only included the standard configuration of the robot manipulator (LWA 4D). The robot manipulator employed as testing system in this work was a special edition of the LWA 4D, which contained a 230-mm extension piece between the third and fourth joints. Thus, for the purpose of analysis, the robot definition was slightly adjusted in the *schunk_lwa4d_moveit_config*, adding the mentioned extension piece together with its physical properties as mass, center of mass, inertial parameters, etc.

Besides, the subfolder *schunk_description/urdf* described the masses of all robot components, their material properties, how they were arranged to each other, where their coordinate systems were located, the dependencies on a moving part fixed coordinate system, the spatial boundaries, and so forth. Furthermore, security zones (collision zones) and the Denavit-Hartenberg parameters, for the kinematic chain of the robot manipulator, were also defined in the URDF-file.

The URDF-file for the mobile platform was built up from scratch based on STL. Since a detailed modeling of all its components was not target-oriented, its structure was therefore analyzed to identify the bodies with the greatest influence on the dynamic behavior of the mobile platform. However, the real geometry must be reproduced as accurately as possible. The mass and COG of each main body must match the reality.

The two wheels for the differential drives of the mobile platform together with its support wheels were added to the robot definition, permitting all the DOF needed for a real motion. Additionally, in order to help the detection of tilting instabilities, the same IMU implemented for approach A (see Section 4.5.4) and the laser scanners were also represented virtually using an additional node: Their raw data were employed as input variables in Gazebo for the current pose of the mobile manipulator in space [200]. Thanks to a coordinate transformation of the gravitational vector, changes in the direction of gravity, e.g., by driving on a ramp, could also be detected.

For the integration of both subsystems into one single model, the robot description of the mobile platform was defined as *parent* for the mobile manipulator, being the part on which the robot manipulator (defined as *child*) was mounted. This integration is shown in Figure 106.

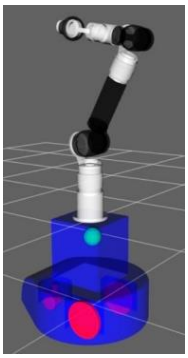


Figure 106 *The virtual model of the mobile manipulator* [95].

In order to obtain the same traveling behavior of the real mobile platform in the simulations (controlled by the computational unit), plugins for the data transfer between ROS and Gazebo for the differential drives were added and configured as in the real robot. These control packages involved the *scitos_drivers*, the *scitos_common* and the *sicks300*, provided in Annex A.5. Consequently, the entire mobile manipulator was able to be completely controlled through ROS.

The robot control program "RobotMainControl" (RMC) allowed the mobile manipulator to interact with its environment. RMC expected as input parameter the target position and orientation for the TCP in world coordinate system. Then, an additional node estimated the target position and orientation for each individual subsystem, the mobile platform and the robot manipulator, so that the expected TCP position and orientation is reached. Given a target position for the TCP, ROS nodes calculate the path planning, communicate their results to the motion driver node and, then, the whole mobile manipulator moves and reaches the target position.

In order to ensure that all required nodes started automatically and quickly one by one in the correct sequence, the launch files *kipp_dynamisch_bringup.launch* and *bringup_robot_main_control_dyn.launch* were implemented to manage the tasks needed for the automated ROS package initiation.

5.4.2 Tip-over detection algorithm

The effect of the linear acceleration of the mobile platform over the entire mobile manipulator can be be analyzed based on the illustrated example in Figure 107.

Figure 107 *Robot manipulator position for the estimation of the effects of linear accelerations.*

In this example, the mobile manipulator's COG is defined as

$$\begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} = \begin{bmatrix} 0\,m \\ 0\,m \\ 0.38\,m \end{bmatrix}$$

and the IMU only detects the value for the gravity vector

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} 0\,m/s^2 \\ 0\,m/s^2 \\ -9.81\,m/s^2 \end{bmatrix}^T$$

The distance between the system COG (S) and one of the two front wheels was d_f=0.075 m. Then, if the mobile platform was assumed to experience an acceleration of 1 m/s^2

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} 1\,m/s^2 \\ 0\,m/s^2 \\ -9.81\,m/s^2 \end{bmatrix}^T$$

a displacement of the projected COG in the direction of the applied acceleration derived. As a result of this, the distance from the COG to the tilt edge reduced to d_f=0.036 m.

The torques acting on each joint of the robot manipulator obtained from the ROS topic *rnea_return*[21] were validated for plausibility on the real mobile manipulator using the following two scenarios:

[21] Refer to [95] for source code.

1. In order to generate the greatest possible variation of torque by the robot manipulator's arm, the COG of the 5th, 6th and 7th robot manipulator links were aligned with the 4th joint rotatory axis so that no additional moments affected the 4th joint. Then, the 4th joint was accelerated from this position to the sides. As a result, during this acceleration the COG was shifted and, thus, an additional moment was generated due to the increasing lever arm. The moment acting on the 4th joint was analytically calculated for each instant with

$$M_4 = I_4 \cdot \omega_4. \tag{119}$$

Table 44 shows the results obtained from the analytical calculations and from the RNEA algorithm.

Table 44 *Torque generated at 4th joint* [95].

Analytical	Results from RNEA node employing the real robot manipulator
0.49 N·m	0.498 N·m

2. The robot manipulator's arm was placed vertically. But now, instead of accelerating its 4th joint to the sides, the mobile platform was linearly accelerated while the robot manipulator stands still.

$$\vec{a} = \begin{pmatrix} 0.37 \\ -0.03 \\ -0.13 \end{pmatrix} \mathrm{m/s^2} \quad , \quad \vec{S} = \begin{pmatrix} -0.0021 \\ -0.0017 \\ 0.6307 \end{pmatrix} \mathrm{m}$$

where $\vec{a}$ describes the linear acceleration of the mobile manipulator and $\vec{S}$ defines the position of its COG with respect to connection point P illustrated in Figure 96.
The inertial forces generated by the linear acceleration were calculated using d'Alembert's principle. Table 45 presents the forces and torques at the connection point P.

Table 45 *Forces and torques generated at point P* [95].

Force/Torque components	Analytical	RNEA using real robot manipulator
F_x	8.5 N	8.5 N
F_y	-3.2 N	-3.2 N
F_z	196.0 N	196.1 N
M_x	1.7 N·m	1.7 N·m
M_y	5.8 N·m	5.8 N·m
M_z	0.0 N·m	0.0 N·m

Thereupon, the proposed tip-over detection algorithm for the testing system was validated by computing the dynamic tilting stability value on both, simulations and real mobile manipulator. Table 46 shows that the calculated tilting stability from the simulations tends to be higher than those calculated from the real mobile manipulator for the same three predefined positions (with deviations smaller than 5%).

Table 46 *Comparison of the tilting stability value for the simulation model and the real robot manipulator in home, transport and one critical position* [175].

Robot manipulator position	Dynamic tilting stability value			
	Regarding tilting edge	Real system	Simulation environment	Discrepancy
Home position	α_{1-cm}	685	698	+1.9%
	α_{2-cm}	675	703	+4.2%
	α_{3-cm}	449	466	+3.8%
Transport position	α_{1-cm}	744	773	+3.9%
	α_{2-cm}	751	779	+3.7%
	α_{3-cm}	396	388	−2.0%
Critical position	α_{1-cm}	960	989	+3.0%
	α_{2-cm}	958	993	+3.6%
	α_{3-cm}	≈0	≈0	≈0%

The main found discrepancy was the value α_{i-cm} at which the mobile manipulator starts to tip over: during the simulations, the mobile manipulator started to tip over at α_{i-cm}=0, whereas the real mobile manipulator did at α_{i-cm}=150. The small variation in the braking process of the real mobile platform regarding the ideal simulation profile could be the reason for the higher stability state of the real mobile manipulator. Another possible cause could be the model simplification used for the robot description employed in the simulations.

Although the causes for the discrepancies might be the same for simulations of all kind of mobile manipulators, the value α_{i-cm} has to be determined for each particular real mobile manipulator.

5.4.3 Tip-over avoidance algorithm

The theoretical local workspace of the testing system robot manipulator of 0.963 m is originally described by the manufacturer as the radius of the sphere formed with its center at the second joint of the robot manipulator. During preliminary examinations, it was identified that this theoretical workspace could not be fully deployed as local workspace for the mobile manipulator due to the risk of tip-over.

Due to this fact, a redefinition of its theoretical local workspace was inferred by finding an optimum volume as large as possible so that the robot manipulator had maximum room to move freely and safely without risk of tip-over.

The diagrams in Figure 108 show the static tilting moment M_{static} as function of a wide range of values for the angles β and ϒ (refer to Section 5.3.2.2.1) of the testing system mobile manipulator. The red lines indicate the critical range in which the system presented instabilities (M_{static}>0).

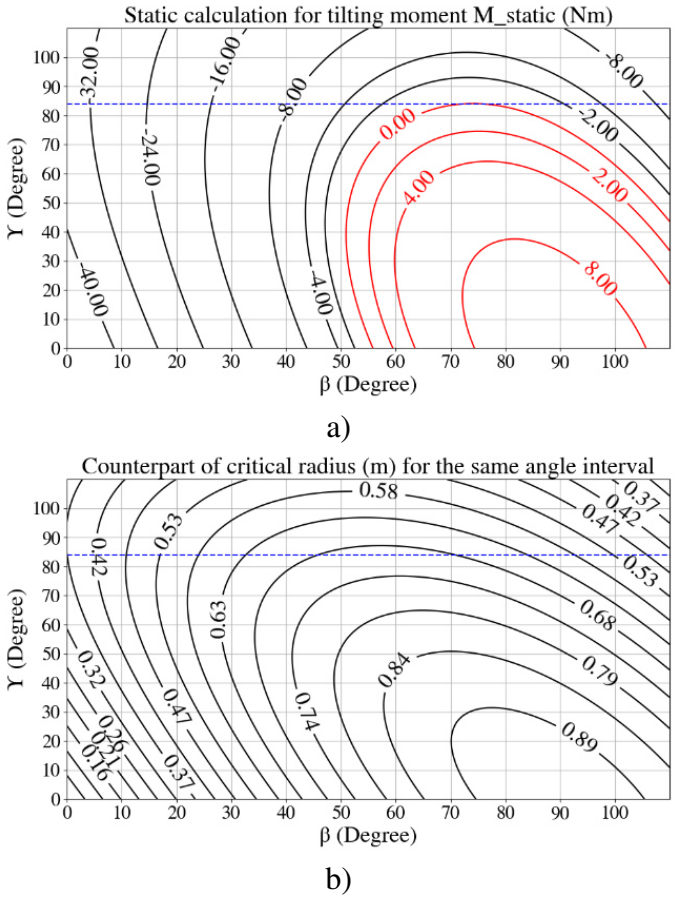


Figure 108 *Estimation of a) tilting moment (top) and b) ellipsoid critical radius (bottom) under static conditions for different angle configurations β and Y* [175].

The minimum value of Υ for which the calculated tilting moment did not exceed 0 N·m over the entire interval of β is pointed out with the blue auxiliary line at Υ=84° in Figure 108-a. This value was transferred to the diagram in Figure 108-b, which helped to estimate the critical radius employed for the workspace optimization. Hence, the maximum minor axis (r_{opt}) for the semi-ellipsoid could be deduced from Figure 108-b to be about 0.7 m.

The optimized workspace was spatially discretized with q=0.1 rad, equivalent to a 5.7° joint angle, as displayed with the dotted volume in Figure 109. Only within this optimized workspace, the robot manipulator was allowed to extend its upper joints as far as possible without causing any instability, as long as the mobile platform was not in motion. Additionally, the robot manipulator did not have to leave this safe optimized working space when its joints were moving to the target coordinate.

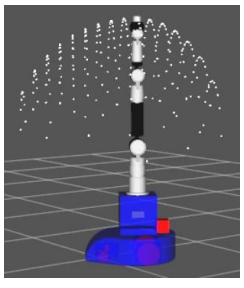


Figure 109 *Optimized workspace for mobile manipulator in RViz* [175].

In addition to the workspace optimization, the second tip-over avoidance countermeasure developed for such scenarios, in which the mobile platform is in motion, is implemented for the testing system. The normal and emergency braking processes of the mobile platform have the same profile as the presented in Section 5.5.3 (see Figure 55).

The diagrams in Figure 110 show the tilting moment generated on the mobile manipulator under different accelerations of the mobile platform: the black level lines display all states in which the stability moment was bigger than the tilting moment (stable states). The red level lines reveal the area at which the tilting moment had a higher value than the stability moment, i.e. unstable states.

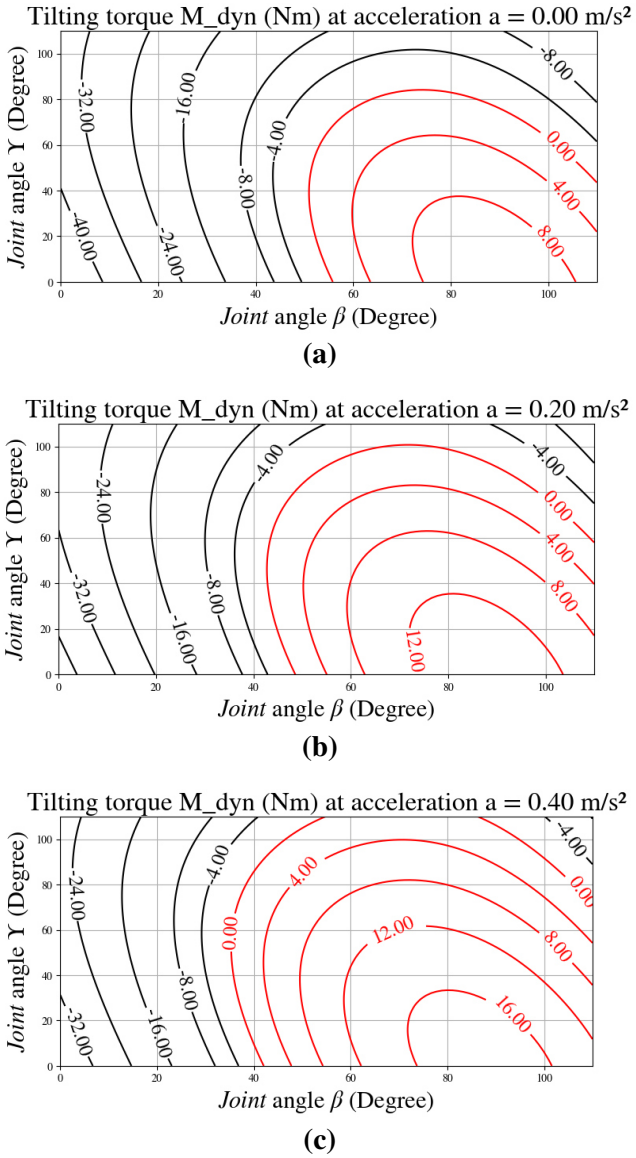
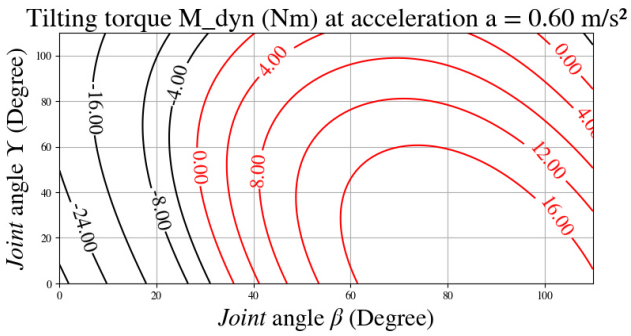
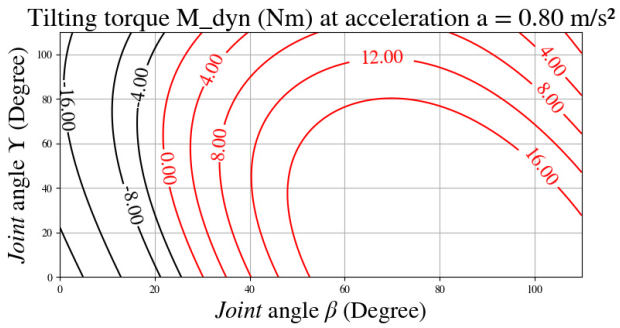


Figure 110 *Analysis of the stability moment as a function of the manipulator´s joint positions and mobile platform acceleration, employing as linear acceleration:* ***(a)*** *0 m/s², (**b**) 0.2 m/s², (**c**) 0.4 m/s², (**d**) 0.6 m/s², and (**e**) 0.8 m/s². The black lines represent stable states and the red lines represent unstable states of the mobile manipulator* [175].



(d)



(e)

Figure 110 (cont.) *Analysis of the stability moment as a function of the manipulator´s joint positions and mobile platform acceleration, employing as linear acceleration:* ***(a)*** *0 m/s², (**b**) 0.2 m/s², (**c**) 0.4 m/s², (**d**) 0.6 m/s², and (**e**) 0.8 m/s². The black lines represent stable states and the red lines represent unstable states of the mobile manipulator* [175].

The area that symbolized the stable states is reduced as a result of the increasing acceleration of the mobile platform, since the inertial forces are directly proportional to this acceleration. The graph below (Figure 111) demonstrates that the stability value $\alpha_{\mathrm{cm-critical}}$ is more dependent on the acceleration of the mobile platform than on the joint angle Υ of the robot manipulator.

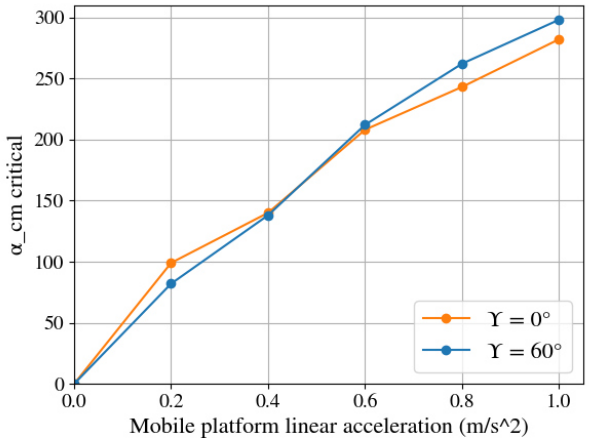


Figure 111 *Critical tilting instability for different values of Υ as a function of the mobile platform traveling acceleration* [175].

Considering 0.8 m/s² as the maximal acceleration of the mobile platform, a repositioning of the robot manipulator has to be triggered if its current stability value drops below the critical

value $\alpha_{\mathrm{cm-critical}}$=250. Additionally, the optimized workspace determined in the previous paragraphs was also adopted for scenarios under dynamic loads. Only then, can it be ensured that the mobile manipulator does not tip over at any time during travelling or braking.

5.4.4 Validation of stabilization strategy employing testing system

The proposed stabilization strategy is considered as successfully validated if:

1. Regarding the optimized workspace, the mobile manipulator is stable and capable of determining whether the mobile platform has to be displaced or not in order to reach a target point without any risk of instability.
2. The tilting stability value of the mobile manipulator is improved with repositioning procedures of the robot manipulator's arm, keeping the initial orientation of its TCP unchanged during the joint rearrangement.

Furthermore, the stabilization strategy should not set the mobile platform in motion until the overall system is sufficiently stable, characterized by the tilting stability value α_{i-cm}.

For the repositioning procedures, the increment used in the calculations of the gradient method (see Section 5.3.2.2.2) was added to and subtracted from the corresponding joint position q_i of the robot manipulator iteratively with a sample rate of 100 ms.

The developed scripts employed for the implementation of the stabilization strategy on the mobile manipulator testing system are available in Annex A.5.

5.4.4.1 Evaluation of the stabilization strategy by means of simulations

In order to assess the developed algorithms (including the tip-over detection) by means of simulations, the following tests scenarios were first performed without stabilization and, then, repeated under the same conditions applying the stabilization strategy.

For each of the sets, the mobile manipulator was set in different configurations before moving towards the target point. The mobile manipulator started at the initial position ($P_{Origin_{MobilePlatform}}$ and $P_{Origin_{RobotManipulator}}$). Then, it received the target points for each of the subsystems, $P_{Target_{MobilePlatform}}$ and $P_{Target_{RobotManipulator}}$, which were not reachable within the optimized workspace. An example of this action can be seen in Figure 112.

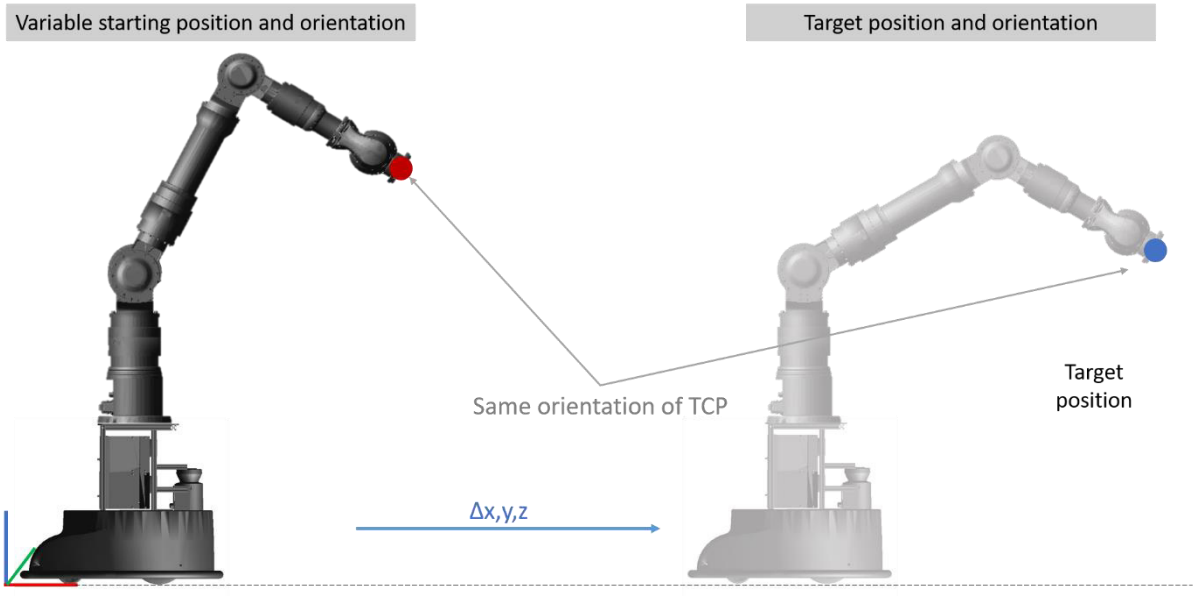


Figure 112 *Example of starting position and orientation of mobile manipulator moving to the predefined target position.*

The mobile manipulator behavior with and without the stabilization strategy are summarized in Table 47. It can be seen from these data that two different critical tilting stability values were considered during the operation: the first value, $\alpha_{cm-critical}$=250, was employed when the mobile manipulator was affected by dynamic conditions (which corresponds to the critical stability value at the maximal acceleration of the mobile platform, 0.8 m/s^2); the second value, $\alpha_{cm-critical}$=150, was used for situations where the mobile platform did not exhibit any accelerations (static tilting stability value). In other words, as long as the mobile platform is in motion, the threshold for a repositioning procedure was represented by $\alpha_{cm-critical}$=250. On the other hand, as soon as the mobile platform reached its target position (and, consequently, the robot manipulator would move to approach its target position), from that moment on, the static tilting stability value, $\alpha_{cm-critical}$=150, defined the threshold for eventual repositioning procedures. If the computed $\alpha_{i\text{-}cm}$ lies below the value for each particular condition, the mobile manipulator tends to tip over its tilting shape.

Table 47 *Comparative scenarios in order to verify the effectiveness of the active stabilization by means of simulations (following* [175]*).*

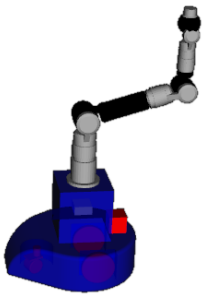
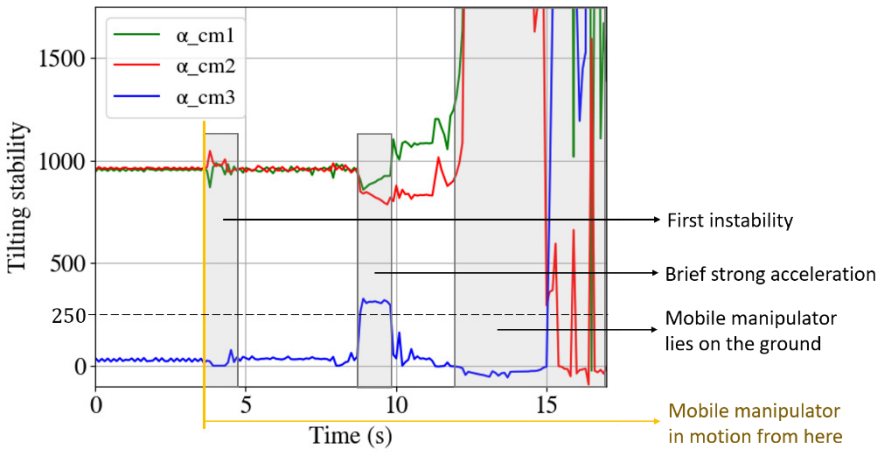
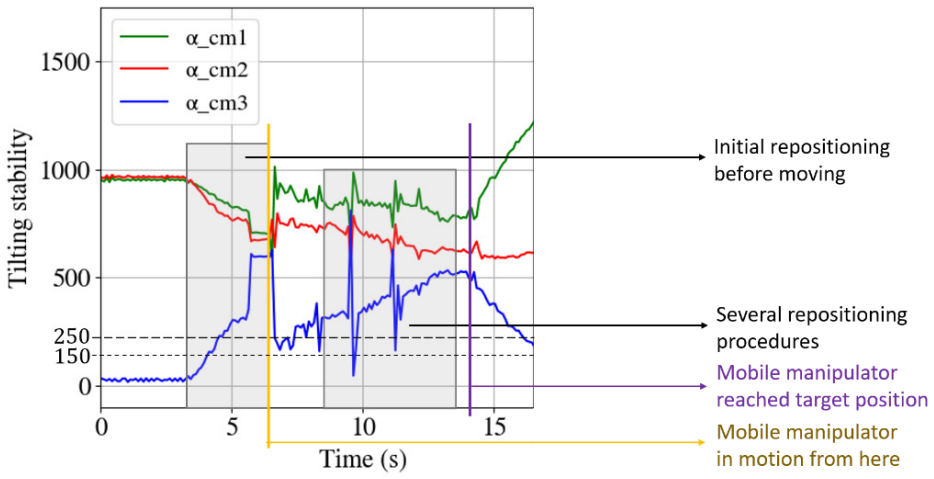
Set 1	
Initial dynamic tilting stability	$\alpha_{3-\mathrm{cm}} = 23$ 
Without stabilization	 The first instability occurred at the time the mobile manipulator started to move (at 4 s): the mobile platform tilted slightly over the front wheels. After a brief strong acceleration at 9 s, which was evident from the sudden increase in $\alpha_{3-\mathrm{cm}}$, a deceleration occured. As a consequence, the mobile manipulator completely tipped over from 12 s onwards.
With stabilization	 A repositioning was performed before the mobile platform started to move, thus the critical dynamic tilting stability increased to about $\alpha_{3-\mathrm{cm}}$=280. As long as the mobile platform was in motion, several repositionings were performed at 7 s, 8 s, 10 s and 12 s, because the dynamic tilting stability went below the threshold $\alpha_{\mathrm{cm-critical}}$=250. At approx. 14 s, the mobile platform reaches $P_{Target_{MobilePlatform}}$ and, hereafter, the robot manipulator moves to $P_{Target_{RobotManipulator}}$.

Table 47 (cont.) *Comparative scenarios in order to verify the effectiveness of the active stabilization by means of simulations (following* [175]*).*

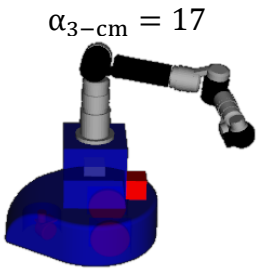
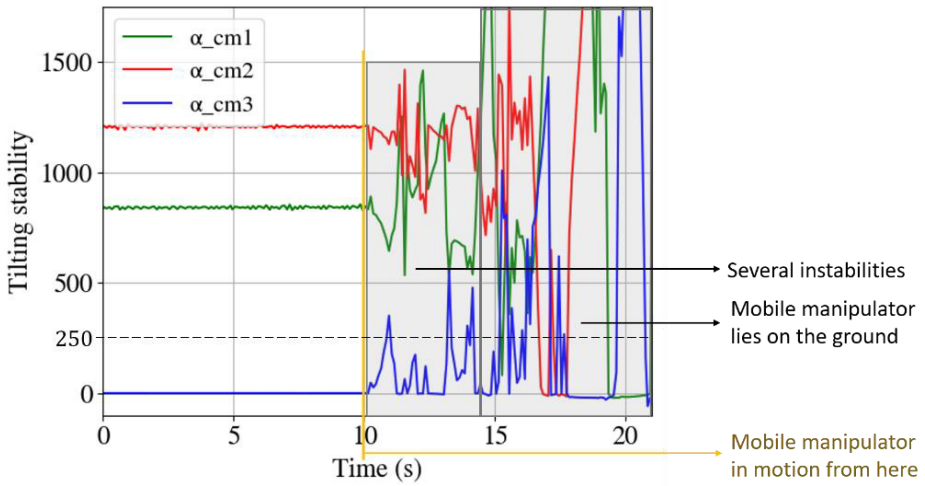
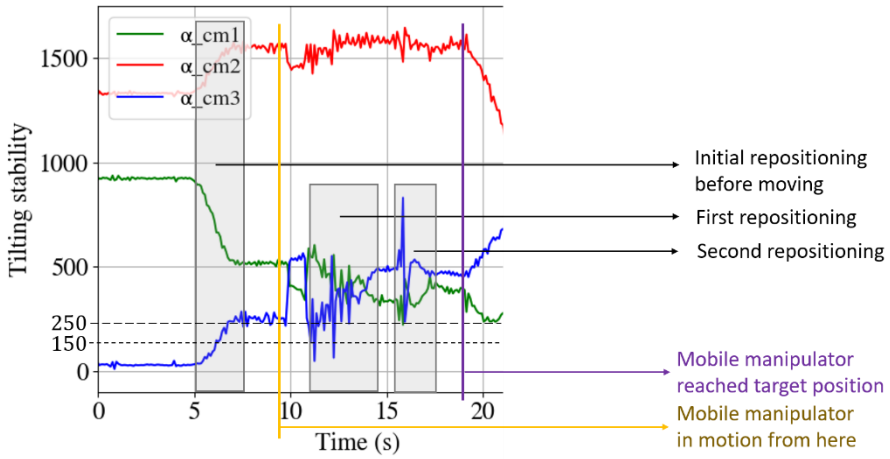
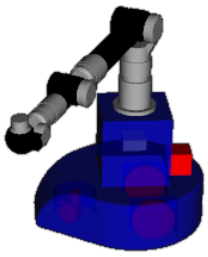
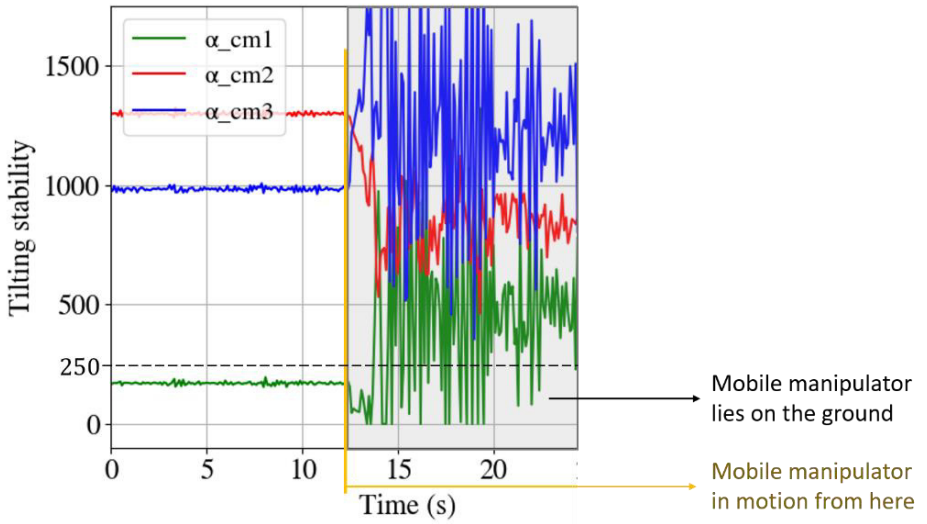
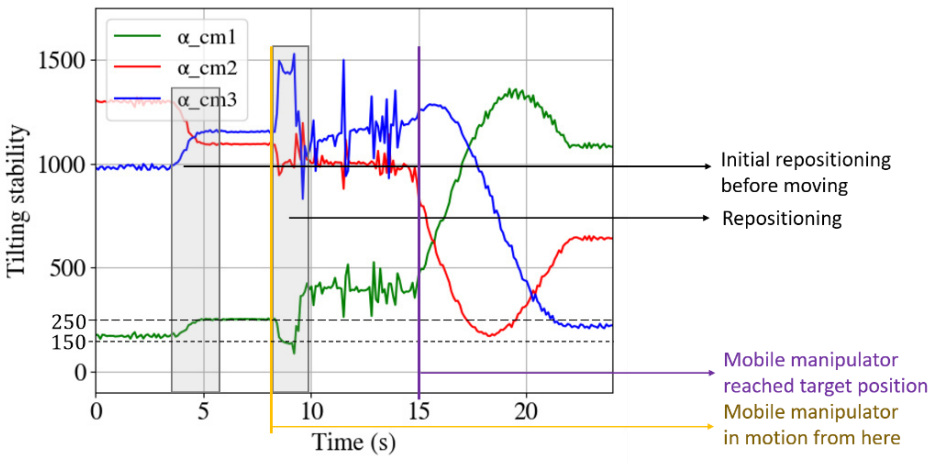
Set 2	
Initial dynamic tilting stability	
Without stabilization	 The acceleration process caused that the mobile manipulator tilted over the front edge several times between 10 s and 14.5 s. From about 14.5 s on, the mobile manipulator tipped completely over the front edge.
With stabilization	 First, before the mobile platform moved, a reposition process took place. The value of the critical tilting stability $\alpha_{3-\mathrm{cm}}$ increased to just over 250. The mobile platform then started to move. Further repositioning iterations took place at about 10 s, 11s and 12 s increasing α_{3-cm}. On the other hand, the $\alpha_{1-\mathrm{cm}}$ fell below the limit value at 16 s. This resulted in a new repositioning for the robot manipulator. At approx. 19 s, the mobile platform reaches $P_{Target_{MobilePlatform}}$ and, hereafter, the robot manipulator moves to $P_{Target_{RobotManipulator}}$.

Table 47 (cont.) *Comparative scenarios in order to verify the effectiveness of the active stabilization by means of simulations (following* [175]*).*

Set 3	
Initial dynamic tilting stability	The critical axis was one of the two lateral tilting axes, at about $\alpha_{1-\mathrm{cm}}$=187, representing the critical tilting stability. 
Without stabilization	 At about 13 s, the entire system tipped over the lateral tilting edge. The bumpers of the mobile platform dragged across the floor, causing a strong noise from 13 s onward.
With stabilization	 A repositioning occurred at about 4 s. The value for $\alpha_{1-\mathrm{cm}}$ increased thus to about 250. The acceleration process at about 7 s caused a destabilizing effect against the lateral tilt edge. A new reposition was initiated, increasing the value for $\alpha_{1-\mathrm{cm}}$ to about 400. At approx. 15 s, the mobile platform reaches $P_{Target_{MobilePlatform}}$ and, hereafter, the robot manipulator moves to $P_{Target_{RobotManipulator}}$.

For all simulations with the stabilization strategy, the repositioning of the robot manipulator took place several times during the trajectory of the mobile platform, increasing the stability of the entire system. As a result, the mobile manipulator reached the target points, both the $P_{Target_{MobilePlatform}}$ and the $P_{Target_{RobotManipulator}}$, without any instability risk. During sets 1 and 2, even a complete overturning was prevented.

5.4.4.2 Evaluation of the stabilization strategy using the real mobile manipulator

Similar to the simulation environment, the joint information of the real mobile manipulator was transmitted by a separate publisher which, in turn, received the information from the individual subsystems, arranged them in a certain sequence and outputted them at defined time intervals.

For the evaluation of the stabilization strategy on the real mobile manipulator, a recursive filter was implemented to reduce the noise of the signals received from the IMU.

$$y_n = (1 - \eta_F) y_{n-1} + \eta_F x_n \qquad (120)$$

where y_n is the output variable, x_n the input variable and y_{n-1} the output value from the previous iteration. The noise on the output variable was adjusted via coefficient η_F ($0 \leq \eta_F \leq 1$), where η_F=1 produced an unfiltered signal and, the closer the coefficient went towards η=0, the less influence the output variable had concerning its original value. For the present purposes, a coefficient η_F=0.25 for acceleration curves was sufficient. In consequence, the accelerations measured on the real mobile manipulator was significantly lower than those in the simulations. The filter was integrated as a separated ROS node.

In the following test scenarios, a real tip-over of the mobile manipulator was not induced in order to avoid damaging to the equipment. Analogous to the simulative tests in Section 5.4.4.1, the sets differed from each other on the initial position and orientation of the robot manipulator and the mobile platform, $P_{Origin_{RobotManipulator}}$ and $P_{Origin_{MobilePlatform}}$.

Table 48 outlines the effect of the stabilization strategy on the real testing system mobile manipulator.

Table 48 *Comparative scenarios in order to verify the effectiveness of the stabilization strategy using the real mobile manipulator (following* [175]*).*

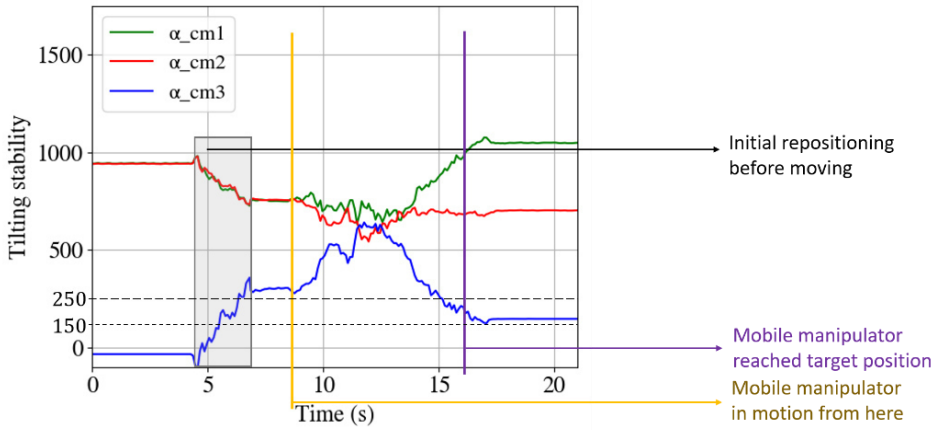
Set 1	
Initial position and orientation of robot manipulator	$\alpha_{3-\mathrm{cm}} = -34$ 
Dynamic tilting stability value during repositioning	 Between 5 s and 7 s, the robot manipulator was rearranged to achieve $\alpha_{3-\mathrm{cm}}$=280 before the mobile platform started to move. During the trajectory towards the target point, no additional repositioning was required. At approx. 16 s, the mobile platform reaches $P_{Target_{MobilePlatform}}$ and, hereafter, the robot manipulator moves to $P_{Target_{RobotManipulator}}$.
End position and orientation of robot manipulator	

Table 48 (cont.) *Comparative scenarios in order to verify the effectiveness of the stabilization strategy using the real mobile manipulator (following* [175]*).*

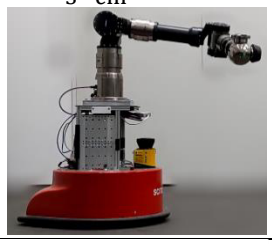
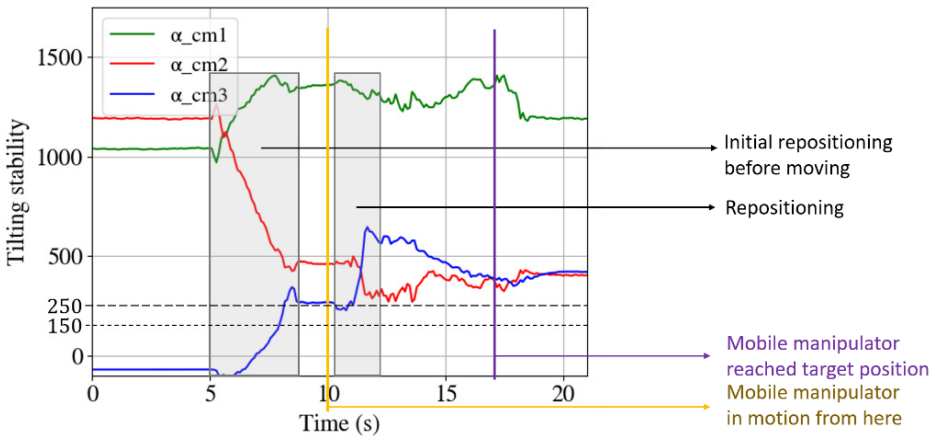
Set 2	
Initial position and orientation of robot manipulator	$\alpha_{3-cm} = -80$ 
Dynamic tilting stability value during repositioning	 Before the mobile platform started moving at 10 s, the robot manipulator was repositioned by arranging its joints q_1 and q_2, thus achieving α_{3-cm}=260. From 11 s to 17 s, the mobile platform kept in motion. At around 11 s, the tilting stability value decreased below $\alpha_{cm-critical}$=250 due to the acceleration of the mobile platform and, as a result, the robot manipulator was repositioned again, keeping this new position until the mobile platform reached its target point $P_{Target_{MobilePlatform}}$ at 17 s. Hereafter, the robot manipulator moves to $P_{Target_{RobotManipulator}}$.
End position and orientation of robot manipulator	

Table 48 (cont.) *Comparative scenarios in order to verify the effectiveness of the stabilization strategy using the real mobile manipulator (following* [175]*).*

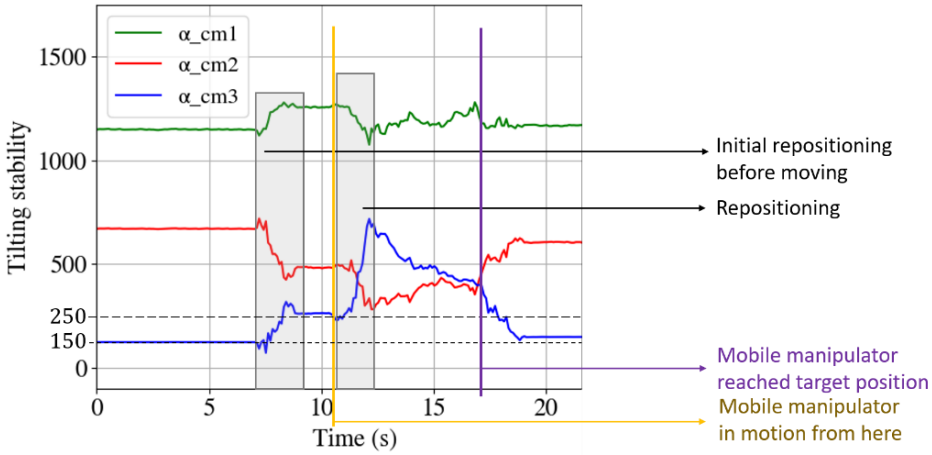
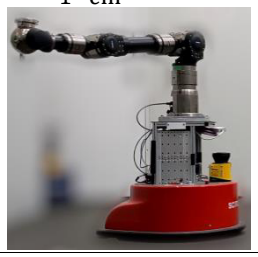
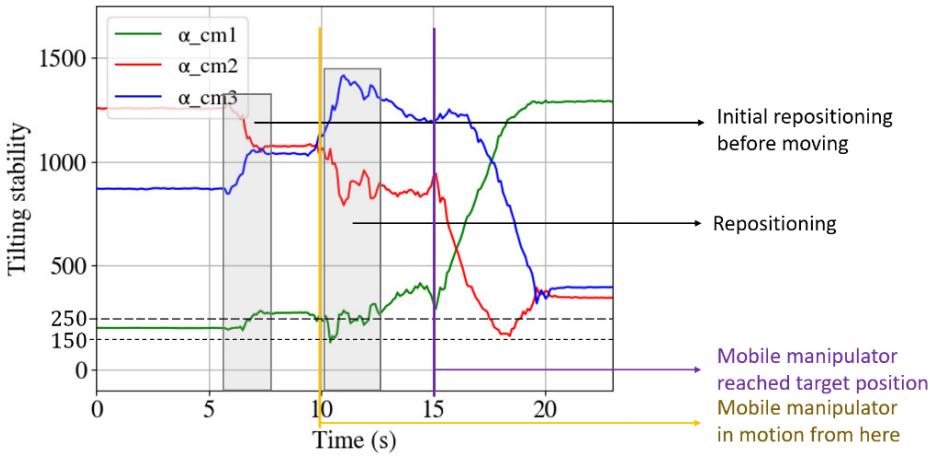
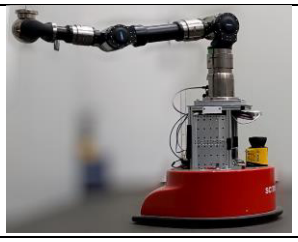
Set 3	
Initial position and orientation of robot manipulator	$\alpha_{3-cm} = 150$ 
Dynamic tilting stability value during repositioning	 Joint q_1 repositioned the robot manipulator before the mobile platform started to move, at 7 s, achieving $\alpha_{cm-critical}$=250. During the acceleration process, the threshold was slightly undercut again at 11 s, so the robot manipulator was repositioned one more time. At 17 s, the mobile manipulator achieved its target point $P_{Target_{MobilePlatform}}$ and, hereafter, the robot manipulator moves to $P_{Target_{RobotManipulator}}$.
End position and orientation of robot manipulator	

Table 48 (cont.) *Comparative scenarios in order to verify the effectiveness of the stabilization strategy using the real mobile manipulator (following* [175]*).*

Set 4	
Initial position and orientation of robot manipulator	$\alpha_{1-cm} = 208$ 
Dynamic tilting stability value during repositioning	 The tilting stability value was improved by a first repositioning at 6 s, rearranging only joints q_1 and q_4. Other repositioning iterations were required due to the mobile platform trajectory path, between 10 s and 12 s. At 15 s, the mobile manipulator reached its target point $P_{Target_{MobilePlatform}}$ and, hereafter, the robot manipulator moves to $P_{Target_{RobotManipulator}}$.
End position and orientation of robot manipulator	

For all the sets, the real mobile manipulator reached the target points for both, the $P_{Target_{MobilePlatform}}$ and the $P_{Target_{RobotManipulator}}$, without any instability risk and keeping its original TCP orientation. Therefore, the proposed stabilization strategy fulfilled the expectations.

6 Concluding assessment

Despite the fact that all implemented strategies show a considerable improvement in the stability state of the compact mobile manipulator's testing systems, they do not possess the same evaluation criteria with regard to their technical feasibility in industrial environments. Table 49 supports this evaluation by providing a summarized comparison of the stabilization strategies presented in this work.

According to their associated pros and contras, the last two stabilization strategies (employing the gyro effect from approach A and repositioning the robot manipulator from approach B) offer a notable stabilizing effect and, furthermore, both preserve the original orientation of the Tool Center Point (TCP) during the execution of the countermeasures. Nevertheless, it can be clearly seen that the last approach, B, provides an additional advantage derived from producing its stabilizing effect (torque compensation) for as long as required by the tilting detection algorithm, just by repositioning the robot manipulator further if necessary. In contrast, the gyroscope produces a torque compensation only during the time the precession motion is carried out.

Table 49 Comparison of the four stabilization strategies.

			Testing/ validation system	Methodology	Outcome	Pros	Cons
Approach A: External stabilization strategies	Employing linear actuators	Inclining/ Tilting	Robot Manipulator: Mitsubishi RV-3AL Mobile platform: MetraLabs Scitos X3	Development and verification using co-simulations between the validated MBS models of the mobile manipulator (MSC.Adams/ View) and their corresponding control algorithms (Matlab/ Simulink).	Mobile manipulator is prevented from tip over	• Bigger compensation of instability forces in comparison with the other linear drives strategy (compensation of angular momentum).	• Orientation of TCP is not maintained during countermeasure, threatening the work piece being handled. • Manipulator must be inclined backwards before the braking process of the mobile manipulator begins, e.g., based on the readings of laser scanners. • Compensation of destabilizing torques only during short periods.
		Compensa-tion of angular momentum			Mobile manipulator is prevented from tip over	• On-time reaction. • Easy to control.	• Orientation of TCP is not maintained during countermeasure, threatening the work piece being handled. • Poor effect against instabilities. • Compensation of destabilizing torques only during short periods.
	Employing gyroscopic effect				Mobile manipulator is prevented from tip over	• On-time reaction. • Quick generation of great compensation torques. • Easy to control. • Orientation of TCP is maintained during countermeasure.	• Large dimensioning of components. • Requires high safety standards due to its large flywheel rotation speed. • Compensation of destabilizing torques only during short periods.
Approach B: Active stabilization strategy	Workspace optimization together with repositioning of robot manipulator		Robot Manipulator: Schunk LWA 4D Mobile platform: MetraLabs Scitos G5	Development of tip-over detection and avoidance algorithms in ROS environments and validation using real testing system.	Mobile manipulator is prevented from tip over	• On-time reaction. • Orientation of TCP is maintained during countermeasure. • Stabilizing countermeasures can be applied as many times and for as long as needed.	• Theoretical workspace is reduced. • Mobile manipulator has to be operated by ROS.

7 Summary and outlook

7.1 Summary

Small-footprint mobile manipulators offer not only high flexibility due to their compact design, but also agility and maneuverability. Nevertheless, this kind of systems tends to tip over due to its ability to travel at large accelerations and to suddenly brake, as well as due to its high center of gravity. The present work explored two different ways to counteract the stability problem of small footprint mobile manipulators.

The first part of this thesis described three stabilization strategies that employ external actuators to generate the required compensation moments. The first proposed mechanism was comprised by linear actuators arranged in delta configuration, integrated between the mobile platform and the robot manipulator to produce a tilting effect and an angular momentum effect. Although it is well established that hexapods are commonly used for motion compensation, its implementation for the stabilization of mobile manipulators is limited due to cost and size constraints. Therefore, this study provides new insights into the proposed simplified mechanism, which only makes use of the g-tilt effect of the hexapod's operating principle.

For the "inclining/tilting" stabilization strategy, the robot manipulator is tilted in the opposite direction of travel of the mobile platform to shift the robot manipulator's COG backwards; this must be done before the braking process begins.

In contrast, the stabilization strategy "conservation of angular momentum" impels the robot manipulator in the direction of travel of the mobile platform during the braking process, thus generating an angular momentum forwards that compensates the decelerations produced by the mobile platform. To accomplish this strategy, the robot manipulator has to be gradually tilted backwards just before the braking process occurs.

The second mechanism was conceptualized based on the gyro effect principle, where the direction of the pivot axis of a rotating mass is rapidly changed to generate the compensation moment employed to stabilize the mobile manipulator. Despite being a well-known mechanism, the use of gyro effect to improve the stability of mobile manipulators had not been previously investigated.

A simulative procedure, based on multibody simulation (MBS) models, was proposed for the sizing and examination of the stabilization strategies. At the first stage, experimental modal analyses (EMA) is required to identify the natural frequencies and mode shapes of both real subsystems, the mobile platform and the robot manipulator. Particular attention has to be paid to the excitation and stimuli spots since they had to be chosen so that all possible directions for system oscillations are excited and measured, avoiding orthogonality. The EMA of the robot manipulator has to be performed in diverse positions and conditions (with energized and non-energized motors) in order to find out if different link configurations affect the dynamic system behavior.

The results obtained in this first stage are employed to parameterize and validate the MBS model of each subsystem in such a way as the modeled MBS system matches the dynamics of the real robot assemblies. In order to keep the simulation time and complexity within acceptable limits, both subsystems can be implemented as rigid bodies. The main challenge in the MBS modeling is to estimate the stiffness and damping values for all joint elements in the system. Each dynamical simulation generates magnitudes of natural frequencies depending on those

stiffness and damping values. Together with their corresponding mode shapes, they are compared with the empirical values obtained via EMA until they match suitably.

A limitation of the introduced MBS modeling technique is the iterative adjustment process to obtain the desired real modal parameters of the robot manipulator, since its greater degree of freedom (DOF) implies more unknown parameters for the stiffness and damping coefficients of each of its joints. Besides this, the robot manipulator is the subsystem most affected by abrupt braking maneuvers because of the height at which its COG is located. For this purpose, this adjustment procedure was improved by an automated parametrization algorithm that considers as reference for the setting not only the natural frequencies, but also the mode shapes by means of MAC. The algorithm was able to mathematically determine the approximate stiffness and damping values of each joint element of the system until the dynamic behavior of the real system matches the simulation model, thereby avoiding the time-consuming manual iterative process.

The actual implementation of the stabilization strategies took place in mechatronic co-simulations, where forces and torques acting in the mobile manipulator as well as angle displacements and velocities issued the output signals for the feedback block of the closed-loop control. They served as real-time reference values which, in turn, allow to compute the new set point for the actuation system to react against instabilities.

This approach was validated in a testing system consisting of a six-axis robot manipulator mounted on an autonomous mobile platform, both with no open access to their control system. A MBS model of the testing system mobile manipulator, including the articulated robot manipulator, the mobile platform and the designed actuation mechanism for the stabilization strategy, was accomplished following the proposed MBS modeling technique. The outcomes of the automated parametrization algorithm verified a good reproduction of the dynamical behavior of the testing system robot manipulator. The experimental work presented here provides one of the first investigations into the complete modeling process of robot manipulators and mobile platforms in order to carry out accurate mechatronic co-simulations: from identifying their modal parameters by means of EMAs, to their modeling and parametrization via MBS systems.

The mechatronic co-simulations demonstrated that all the presented strategies using external actuators improved the stability of the mobile manipulator and, thus, reduced its risk of tip-over, especially when large accelerations and decelerations affect the system. However, the “inclining/tilting” technique implied the prediction of the braking profile, making its implementation as a closed-control loop difficult. On the other hand, the “conservation of angular momentum-based principle” could be integrated in a closed-control loop, but its impact to the stability state was only minor. For both strategies, it was possible to achieve an improvement in the stability of the overall system for a short period of time. However, the stability of the mobile manipulator cannot be guaranteed if the mobile platform is subjected to higher loads or accelerations/decelerations.

Finally, the gyro stabilizer exhibited major potential, generating enough compensation torques against instabilities.

Although the current study was implemented on particular robots, it offers a helpful insight into the methodology for further complex systems, especially with regards to mobile platforms and robot manipulators, whose dynamic behavior is still little known.

The second part of this thesis addressed a further stabilization strategy which counteracts the instabilities with the own robot manipulator links. This stability strategy was developed so that it operates under robot manipulator movements, mobile platform movements, COGs

position vector and gravity vector changes. The stabilization strategy comprised the implementation of a workspace optimization and an active stabilization by means of repositioning the robot manipulator links.

As a first step, the tip-over detection algorithm was prepared. Despite the fact that the method Force-Angle (FA) stability measure was able to detect a tip-over risk, the dynamic stability value was estimated by the Height Stability method (MHS), since FA inspects only external forces measured by inertial measurement units (IMU). The advantage of the MHS method, over the FA approach, is that it also considers internal forces of the robot manipulator, i.e. inertial forces and torques produced from joint accelerations. Additionally, it gives a constant feedback of the stability value, while the FA approach only gives the stability status as a binary value. Especially under scenarios where the mobile platform was moving, the MHS method proved to be the most suitable for the tip-over detection algorithm. In addition, the value of the dynamic tilting stability, α, over time provides information about the degree of instability during the mobile manipulator motions. The bigger the value of α, the more stable the mobile manipulator is against a tip-over regarding the corresponding tilting edge for which the value was calculated. Therefore, the smaller the value of α, the greater the risk to tip over. The implemented MHS method considered not only the direction of the overall acceleration but also its effect on the whole system, since the system might appear to be stable, despite the fact that resulting inertial forces cause the system to tip over. The tip-over edge also acted as very important indicator for the implementation of suitable tip-over preventing actions, since a countermeasure applied to the wrong tilting edge facilitates the system tip-over.

Hereafter, the theoretical workspace of the robot manipulator was optimized based on the tilting stability value α, in order to constrain the maximum volume in which the robot manipulator is able to operate without instability risks when the mobile platform does not move.

Additionally, a tip-over avoidance algorithm was conceived. It triggered the repositioning of the robot manipulator´s links if the stability value resulting from the tip-over detection algorithm lay below a predefined stability threshold. The repositioning algorithm determined a safe configuration of the robot manipulator based on a gradient method, calculating the mobile manipulator´s stability value depending on different joint arrangements about the current TCP position. The algorithm was designed so that the robot manipulator TCP is kept as close as possible from its original orientation.

The strategy covered both, a predictive and real time countermeasure: a preplanning algorithm brought the mobile manipulator into a stable configuration before the mobile platform started moving; additionally, any instabilities that occurred during travel were compensated by repositioning the robot manipulator without stopping the mobile platform tasks.

Both, the workspace optimization and the repositioning of the robot manipulator's arm, contributed to improve the stabilization of the mobile manipulator. The entire active stabilization strategy consists of the following features:

- After receiving a target coordinate, the mobile manipulator decides independently whether the target point is within the predefined reachable and stable workspace or outside it. In case the target point is outside the optimized volume, the mobile platform moves to approach it.
- In case instabilities are detected by MHS before the mobile platform moves, the robot manipulator repositions its links to a non-critical, stable configuration.
- If instabilities are caused by accelerations or decelerations of the mobile platform, which means that the stability value falls below the predefined threshold value, the robot

manipulator repositions itself without aborting any tasks. During this repositioning process, the initial orientation of the TCP is maintained.

- The robot manipulator approaches the target point as soon as the mobile platform reaches the target position, so that by extending the robot manipulator´s links, the TCP is situated within the stable optimized workspace.

This strategy improved the approaches presented in literature, since only the robot manipulator was employed to achieve the stabilization, without considering any modification of the mobile platform's motion path. A further distinction is that the dynamic tilting stability value was utilized not only during the tip-over detection, but also for the tip-over avoidance, being the reference for the gradient potential function used for the calculation of the new joint configuration needed for the repositioning of the robot manipulator's links. And, perhaps the most significant added value with respect to previous works is that the countermeasures were carried out in real time and without manipulating the motion control of the robots, thus reducing the complexity for the implementation and increasing its flexibility as a universal solution.

The approach was implemented in a second mobile manipulator testing system. Real inspections showed that the mobile manipulator tended to tip over at certain configurations. The algorithms were implemented using ROS environment tools, and validated using the real system. Their effectiveness was proved observing the simulative and real mobile manipulator achieving its target position without suffering any instability. The successful tests using the real mobile manipulator and the corresponding results indicated that the stabilization strategy was able to avoid many tip-overs of the mobile manipulator under different circumstances.

7.2 Outlook

A relevant aspect observed during the implementation of the stabilization strategies of Approach A in the testing system were the appearance of certain discrepancies between the MBS- and the EMA-results, which were considered as acceptable for the purpose of this work. These deviations have been caused, on the one hand, by the model simplification to decrease the complexity of the robotic systems (e.g., bodies were modeled as rigid elements to avoid large computational effort, the friction coefficients were not experimentally determined, etc.) and, on the other hand, by parametric uncertainties such as lack of material homogeneity, unknown tolerances and clearances in the geometry, etc. Non-ideal real conditions in MBS models lead to difficulties in the adjustment of their modal parameters. Therefore, the developed parametrization algorithm reinforced the modeling of a more close-to-reality system. Despite the fact that satisfying results were obtained for the robot manipulator testing system, the algorithm would be improved if other parameters that cause great impact on the modal behavior of a certain system were easily identified. Furthermore, an extension of the optimization algorithm for mass, density and those parameters identified by additional sensitivity analyses would also complement the automated parametrization.

The stabilization strategies employing external actuators demonstrated the compensation of presented instabilities in the testing system. However, with respect to the viability, they could be insufficient if the duration of the braking process of the mobile platform increases.

From the two stabilization strategies using linear drives as external actuators, the "inclination/tilting" of the robot manipulator before the braking occurs offered significant improvement of the mobile manipulator stability, but its principle of action cannot be included into a closed-loop control without more ado. A solution for this limitation could be the

implementation of AI-techniques in order to predict the braking process profile in a timely manner, each time the laser scanner of the mobile platform sends a warning signal.

Moreover, the “conservation of angular momentum”, in which the robot manipulator is angular impelled in the direction of the braking process, was fast enough to react in a proper manner without any prior prediction, but its effects with respect to the stability compensation torque were only minor. Based on the same principle, the gyroscopic effect showed good potential for the compensation of instability torques, being, from all external stabilization strategies, the most promising one. The design optimization of the gyroscope regarding a flatter and bigger diameter might even enhance its performance.

The second approach, B, consisting of a tip-over detection and tip-over avoidance algorithm for those mobile manipulators with an open-access robot operating system exhibited better results. The small deviations in the calculated tilting stability value were compensated by repetitive repositioning processes. Although each joint constraints were taken into account for the computation of the new stable configuration of the robot manipulator, the repositioning was not executed if this new joint configuration was situated within the predefined collision space. This problem could be counteracted by using a new function, in which the collision constraints were provided as no-go areas for the calculations, which in turn seeks for suitable, collision-free joint positions.

Future research is recommended to explore the impact of employing other testing models of mobile manipulators and their different types of path planning and motion control algorithms. In any case, the outcomes point to the need for testing in a wide variety of environments to reinforce the developed algorithms.

8 List of References

[1] Zukunftsinstitut. Die Megatrends 2020. https://www.zukunftsinstitut.de/dossier/megatrends/ (accessed March 14, 2021).
[2] Zukunftsinstitut. Die 5 wichtigsten Megatrends für Unternehmern in den 2020ern 2020. https://www.zukunftsinstitut.de/artikel/die-5-wichtigsten-megatrends-fuer-unternehmern-in-den-2020ern/ (accessed March 14, 2021).
[3] Weinzierl S. Klarer Trend zu Losgröße 1. Produktion - Tech Und Wirtschaft Für Die Dtsch Ind 2017. https://www.produktion.de/wirtschaft/klarer-trend-zu-losgroesse-1-323.html (accessed March 14, 2021).
[4] ZukunfsInstitut. Glossar Individualisierung: Trendbegriffe zur Individualisierung 2020. https://www.zukunftsinstitut.de/artikel/megatrend-glossar/individualisierung-glossar/ (accessed March 14, 2021).
[5] Obermaier R. Handbuch Industrie 4.0 und Digitale Transformation: Betriebswirtschaftliche, technische und rechtliche Herausforderungen. Springer Fachmedien Wiesbaden; 2019.
[6] Mertens P, Potthof I. Wirtschaftsinformatik - von den Moden zum Trend. Bereich Wirtschaftsinformatik I; 1994.
[7] Inc. A. Fanuc Ltd. - Company Profile, Information, Business Description, History, Background Information on Fanuc Ltd. 2021. https://www.referenceforbusiness.com/history2/43/Fanuc-Ltd.html (accessed March 14, 2021).
[8] Florian Krebs, Stefan Nuschele. Kostenreduktion in der Qualitätssicherung durch Roboter-basierte zerstörungsfreie Prüfung. Augsburg, Germany: 2012.
[9] Matthias Reinisch. Auswirkungen verschiedener Varianten des Fabriklayouts auf die Materialflussplanung unter Beachtung der Lean Prinzipien. Technische Universität Graz, 2011.
[10] Poll D. Woraus die Smart Factory besteht und was aktuell dazu kommt. Produktion - Tech Und Wirtschaft Für Die Dtsch Ind 2020. https://www.produktion.de/technik/woraus-die-smart-factory-besteht-und-was-aktuell-dazu-kommt-118.html (accessed March 14, 2021).
[11] Zenner MJ. Autonomous Mobile Robots Push Robot Boundaries. Robot Ind Assoc 2019. https://www.automate.org/industry-insights/autonomous-mobile-robots-push-robot-boundaries (accessed March 14, 2021).
[12] Sinsel A. Das Internet der Dinge in der Produktion: Smart Manufacturing für Anwender und Lösungsanbieter. Springer Berlin Heidelberg; 2019.
[13] Fechter M. ARENA2036. Fraunhofer-Institut Für Produktionstechnik Und Autom IPA 2020. https://www.ipa.fraunhofer.de/de/zusammenarbeit/industry-on-campus/arena2036.html (accessed March 11, 2021).
[14] Unknown. ARENA2036: Der Forschungsbereich “ForschFab.” Bundesministerium Für Bild Und Forsch n.d. https://www.forschungscampus.bmbf.de/forschungscampi/arena2036/arena2036_forschungsfabrik (accessed March 14, 2021).
[15] Unknown. Die ARENA2036. ARENA2036 eV n.d. https://www.arena2036.de/de/ (accessed March 11, 2021).
[16] Poll D. Intralogistik: Die coolsten autonomen Transportsysteme -. Produktion - Tech Und Wirtschaft Für Die Dtsch Ind 2019. https://www.produktion.de/technik/intralogistik-die-coolsten-autonomen-transportsysteme-110.html (accessed March 11, 2021).

[17] Ullrich G, Kachur PA. Automated Guided Vehicle Systems: A Primer with Practical Applications. Springer Berlin Heidelberg; 2014.
[18] Hvilshj M, Bgh S, Madsen O, Kristiansen M. The mobile robot “Little Helper”: Concepts, ideas and working principles. ETFA 2009 - 2009 IEEE Conf. Emerg. Technol. Fact. Autom., 2009, p. 1–4. https://doi.org/10.1109/ETFA.2009.5347251.
[19] Tr A, Dogra A, Singla E. Workspace Reconstruction for Designing Modular Reconfigurable Manipulators, 2020. https://doi.org/10.1007/978-981-15-5689-0_24.
[20] Rey DA, Papadopoulos EG. On-line automatic tipover prevention for mobile manipulators. IEEE Int. Conf. Intell. Robot. Syst., 1997. https://doi.org/10.1109/iros.1997.656414.
[21] Tahboub KA. Robust control of mobile manipulators. J Robot Syst 1996. https://doi.org/10.1002/(SICI)1097-4563(199611)13:11<699::AID-ROB2>3.0.CO;2-Q.
[22] Ding X, Liu Y, Hou J, Ma Q. Online Dynamic Tip-Over Avoidance for a Wheeled Mobile Manipulator with an Improved Tip-Over Moment Stability Criterion. IEEE Access 2019. https://doi.org/10.1109/ACCESS.2019.2915115.
[23] Böge A, Böge W. Technische Mechanik: Statik – Reibung – Dynamik – Festigkeitslehre – Fluidmechanik. vol. 33. Springer Vieweg; 2019. https://doi.org/10.1007/978-3-658-25724-8.
[24] DØssing O. Strukturen prüfen Teil 1: Mechanische Beweglichkeits-Messungen. 1989.
[25] Weck M. Werkzeugmaschinen 5: Messtechnische Untersuchung und Beurteilung, dynamische Stabilität. Springer Berlin Heidelberg; 2006.
[26] Pastor M, Binda M, Harčarik T. Modal Assurance Criterion. Procedia Eng 2012;48:543–8. https://doi.org/https://doi.org/10.1016/j.proeng.2012.09.551.
[27] Irretier H. Experimentelle Modalanalyse in der Rotordynamik. VDI-Schwingungstagung, Kassel, Germany: 2000.
[28] Rosenow SE. Identifikation des dynamischen Verhaltens schiffbaulicher Strukturen. 2007.
[29] Zeller P, Andreas E, Fastl H, Kerber S, Hobelsberger J, Jebasinski R, et al. Handbuch Fahrzeugakustik: Grundlagen, Auslegung, Berechnung, Versuch. Springer Fachmedien Wiesbaden; 2018.
[30] Brandt A, Vaarning C. A Comparison of Non-Parametric Techniques for FRF Estimation Using Pure Random Excitation. Conf Proc Soc Exp Mech Ser 2012;5:523–34. https://doi.org/10.1007/978-1-4614-2425-3_49.
[31] DØssing O. Strukturen prüfen Teil 2: Modalanalyse und Simulation. 1989.
[32] Siemens Simcenter. What is a Frequency Response Function (FRF)? 2020. https://community.sw.siemens.com/s/article/what-is-a-frequency-response-function-frf (accessed May 3, 2022).
[33] Kipfmüller M. Aufwandsoptimierte Simulation von Werkzeugmaschinen. Shaker Verlag, 2010. https://doi.org/10.5445/IR/1000014668.
[34] Christl J, Kunz S, Bayrasy P, Kalmykov I, Kleinert J. FEA-MBS-Coupling-Approach for Vehicle Dynamics. NAFEMS Eur. Conf., Turin, Italy: 2015.
[35] Gustafsson L, Sternad M, Gustafsson E, Gustafsson L, Sternad M, Gustafsson E. The Full Potential of Continuous System Simulation Modelling. Open J Model Simul 2017;5:253–99. https://doi.org/10.4236/OJMSI.2017.54019.
[36] Woernle C. Mehrkörpersysteme: Eine Einführung in die Kinematik und Dynamik von Systemen starrer Körper. Springer Berlin Heidelberg; 2016.
[37] Dresig H, Rockhausen L, Holzweißig F. Maschinendynamik. Springer Berlin Heidelberg; 2013.
[38] Kreuzer E, Lugtenburg JB, Meißner HG, Truckenbrodt A. Industrieroboter: Technik, Berechnung und anwendungsorientierte Auslegung. Springer Berlin Heidelberg; 2012.
[39] Zirn O, Weikert S. Modellbildung und Simulation hochdynamischer Fertigungssysteme:

Eine praxisnahe Einführung. Springer Berlin Heidelberg; 2006.
[40] Rill G, Schaeffer T. Grundlagen und Methodik der Mehrkörpersimulation: Vertieft in Matlab-Beispielen, Übungen und Anwendungen. Springer Fachmedien Wiesbaden; 2017.
[41] Glöckler M. Simulation mechatronischer Systeme: Grundlagen und Beispiele für MATLAB®und Simulink®. Springer Fachmedien Wiesbaden; 2018.
[42] Adams n.d. https://www.mscsoftware.com/de/product/adams (accessed December 27, 2021).
[43] Adams - CAE Simulation und Solutions n.d. https://www.cae-sim-sol.com//software/msc-software/adams (accessed January 22, 2022).
[44] MATLAB - MathWorks - MATLAB & Simulink n.d. https://de.mathworks.com/products/matlab.html..html (accessed December 27, 2021).
[45] MATLAB® - The MathWorks - PDF Catalogs | Technical Documentation | Brochure n.d. https://pdf.directindustry.com/pdf/mathworks/matlab/12865-370414.html (accessed January 22, 2022).
[46] Pietruszka WD, Glöckler M. MATLAB®und Simulink®in der Ingenieurpraxis: Modellbildung, Berechnung und Simulation. Springer Fachmedien Wiesbaden; 2021.
[47] Control System Toolbox - MATLAB n.d. https://de.mathworks.com/products/control.html (accessed January 22, 2022).
[48] Simulink - The MathWorks - PDF Catalogs | Technical Documentation | Brochure n.d. https://pdf.directindustry.com/pdf/mathworks/simulink/12865-370436.html (accessed January 22, 2022).
[49] ROS: Home n.d. https://www.ros.org/ (accessed January 22, 2022).
[50] STRANDS · GitHub n.d. https://github.com/strands-project (accessed October 17, 2021).
[51] Newman WS. A Systematic Approach to Learning Robot Programming with ROS. CRC Press; 2017.
[52] Quigley M, Gerkey B, Smart WD. Programming Robots with ROS. O'Reilly; 2015.
[53] Gazebo n.d. http://gazebosim.org/ (accessed January 22, 2022).
[54] Chitta S, Hershberger D, Pooley A, Coleman D, Gorner M, Suarez F, et al. MoveIt Tutorials — moveit_tutorials Kinetic documentation 2018. http://docs.ros.org/en/kinetic/api/moveit_tutorials/html/ (accessed October 16, 2021).
[55] Featherstone R. Rigid Body Dynamics Algorithms. 1st ed. Springer US; 2008. https://doi.org/10.1007/978-1-4899-7560-7_1.
[56] Featherstone R. Robot Dynamics Algorithms. Springer US; 1987. https://doi.org/10.1007/978-0-387-74315-8.
[57] Fraunhofer Institute for Factory Operation and Automation. Mobile robots support airplane manufacturers - Research News - Topic 4 2014. https://www.fraunhofer.de/en/press/research-news/2014/april/mobile-robots.html (accessed January 17, 2022).
[58] KUKA AG. Mobility n.d. https://www.kuka.com/products/mobility/ (accessed January 17, 2022).
[59] Acar C, Murakami T. Underactuated two-wheeled mobile manipulator control using nonlinear backstepping method. 2008 34th Annu. Conf. IEEE Ind. Electron., 2008, p. 1680–5. https://doi.org/10.1109/IECON.2008.4758206.
[60] German Design Award. KUKA KMR iiwa - Special Mention Industry 2017. https://www.german-design-award.com/en/the-winners/gallery/detail/8600-kuka-kmr-iiwa.html (accessed May 3, 2022).
[61] Dimalog. Mobile Cobots n.d. https://www.dimalog.com/mobile-cobots/ (accessed May 3, 2022).
[62] Gráf R, Dillmann R. Aktive Beschleunigungskompensation mittels einer Stewart-

Plattform auf einem mobilen Roboter, 1997, p. 189–98. https://doi.org/10.1007/978-3-642-60904-6_17.
[63] Gráf R, Dillmann R. Active acceleration compensation using a Stewart-platform on a mobile robot, 1997, p. 59–64. https://doi.org/10.1109/EURBOT.1997.633569.
[64] Gráf R, Dillmann R. Die Stewart-Plattform als dynamisches Lastaufnahmesystem eines mobilen Roboters, 1999, p. 150–9. https://doi.org/10.1007/978-3-642-59708-4_15.
[65] Campos Bonilla AA, Quintero J, Saltaren R, Ferre M, Aracil R. Robotic Strategies to Assist Pilots in Landing and Takeoff of Helicopters on Ships and Offshore, 2010. https://doi.org/10.5772/10308.
[66] Gráf R, Dillmann R. Acceleration compensation using a Stewart platform on a mobile robot, 1999, p. 17–24. https://doi.org/10.1109/EURBOT.1999.827617.
[67] Dang AXH, Ebert-Uphoff I. Active acceleration compensation for transport vehicles carrying delicate objects. IEEE Trans Robot 2004;20:830–9. https://doi.org/10.1109/TRO.2004.832791.
[68] Xiaoli B, Jeremy D, James D, Turner J, Junkins J. Dynamics, Control and Simulation of a Mobile Robotic System for 6-DOF Motion Emulation. Lect Notes Eng Comput Sci 2007;2167.
[69] Danko TW, Chaney KP, Oh PY. A parallel manipulator for mobile manipulating UAVs. 2015 IEEE Int. Conf. Technol. Pract. Robot Appl., 2015, p. 1–6. https://doi.org/10.1109/TePRA.2015.7219682.
[70] Khaled M, Mohammed A, Ibraheem MS, Ali R. Balancing a Two Wheeled Robot. 2009. https://doi.org/10.13140/RG.2.2.25634.63683.
[71] Chan RPM, Stol KA, Halkyard CR. Review of modelling and control of two-wheeled robots. Annu Rev Control 2013;37:89–103. https://doi.org/https://doi.org/10.1016/j.arcontrol.2013.03.004.
[72] Wang J-J. Simulation studies of inverted pendulum based on PID controllers. Simul Model Pract Theory 2011;19:440–9. https://doi.org/https://doi.org/10.1016/j.simpat.2010.08.003.
[73] Bode H. MATLAB-SIMULINK: Analyse und Simulation dynamischer Systeme. Vieweg+Teubner Verlag; 2006.
[74] Bertram T, Svaricek F. Zur Fuzzy-Regelung eines aufrechtstehenden Pendels/On Fuzzy-control of an inverted pendulum. - Autom 1992;40:308–10. https://doi.org/doi:10.1515/auto-1992-0808.
[75] Wey T, Spielmann M. Analytische und Fuzzy-Regelungskonzepte am Beispiel eines aufrechtstehenden Pendels. - Autom 1999;47. https://doi.org/10.1524/auto.1999.47.1.20.
[76] Anderson CW. Learning to control an inverted pendulum using neural networks. IEEE Control Syst Mag 1989;9:31–7. https://doi.org/10.1109/37.24809.
[77] Miao S, Cao Q. Modeling of self-tilt-up motion for a two-wheeled inverted pendulum. Ind Robot An Int J 2011;38:76–85. https://doi.org/10.1108/01439911111097878.
[78] Alarfaj M, Kantor G. Centrifugal force compensation of a two-wheeled balancing robot. 2010 11th Int. Conf. Control Autom. Robot. Vis., 2010, p. 2333–8. https://doi.org/10.1109/ICARCV.2010.5707337.
[79] Kim S, Seo J, Kwon S. Development of a two-wheeled mobile tilting amp; balancing (MTB) robot. 2011 11th Int. Conf. Control. Autom. Syst., 2011, p. 1–6.
[80] Kwon S, Kim S, Yu J. Tilting-Type Balancing Mobile Robot Platform for Enhancing Lateral Stability. IEEE/ASME Trans Mechatronics 2015;20:1470–81. https://doi.org/10.1109/TMECH.2014.2364204.
[81] Zhao Y, Woo C, Lee J. Balancing control of mobile manipulator with sliding mode controller. 2015 15th Int. Conf. Control. Autom. Syst., 2015, p. 802–5. https://doi.org/10.1109/ICCAS.2015.7364730.

[82] Acar C, Murakami T. Multi-task control for dynamically balanced two-wheeled mobile manipulator through task-priority. 2011 IEEE Int. Symp. Ind. Electron., 2011, p. 2195–200. https://doi.org/10.1109/ISIE.2011.5984501.
[83] Leboutet Q, Dean-León E, Cheng G. Tactile-based compliance with hierarchical force propagation for omnidirectional mobile manipulators. 2016 IEEE-RAS 16th Int. Conf. Humanoid Robot., 2016, p. 926–31. https://doi.org/10.1109/HUMANOIDS.2016.7803383.
[84] Cha Y-S, Kim K, Lee J-Y, Lee J, Choi M, Jeong M-H, et al. MAHRU-M: A mobile humanoid robot platform based on a dual-network control system and coordinated task execution. Rob Auton Syst 2011;59:354–66. https://doi.org/https://doi.org/10.1016/j.robot.2011.01.003.
[85] Milighetti G, Petereit J, Kuntze H-B. Mobile Experimental Platform for the Development of Environmentally Interactive Control Algorithms towards the Implementation on a Walking Humanoid. ISR 2010 (41st Int. Symp. Robot. Robot. 2010 (6th Ger. Conf. Robot., 2010, p. 1–7.
[86] Keshtkar S, Moreno JA, Kojima H, Uchiyama K, Nohmi M, Takaya K. Spherical gyroscopic moment stabilizer for attitude control of microsatellites. Acta Astronaut 2018;143:9–15. https://doi.org/https://doi.org/10.1016/j.actaastro.2017.10.033.
[87] ARNOLD RN, MAUNDER L. CHAPTER 9 - GYROSCOPIC VIBRATION ABSORBERS AND STABILIZERS. In: ARNOLD RN, MAUNDER L, editors. Gyrodynamics its Eng. Appl., Academic Press; 1961, p. 177–227. https://doi.org/https://doi.org/10.1016/B978-0-12-063852-9.50012-8.
[88] Novoselov VS. Motion of stabilized gyroscopic systems on a moving base. J Appl Math Mech 1959;23:1375–81. https://doi.org/https://doi.org/10.1016/0021-8928(59)90142-X.
[89] Matrosov VM. On the stability of gyroscopic stabilizers. J Appl Math Mech 1960;24:1214–24. https://doi.org/https://doi.org/10.1016/0021-8928(60)90102-7.
[90] Kuz'mina LK. On stability of systems of gyroscopic stabilization in the presence of perturbations. J Appl Math Mech 1980;44:119–22. https://doi.org/https://doi.org/10.1016/0021-8928(80)90184-7.
[91] Ghasempoor A, Sepehri N. Measure of machine stability for moving base manipulators. Proc. - IEEE Int. Conf. Robot. Autom., 1995. https://doi.org/10.1109/robot.1995.525596.
[92] Li Y. Dynamic stability analysis and control for the mobile manipulator. Can. Conf. Electr. Comput. Eng., 2002. https://doi.org/10.1109/ccece.2002.1015287.
[93] Papadopoulos EG, Rey DA. New measure of tipover stability margin for mobile manipulators. Proc. - IEEE Int. Conf. Robot. Autom., 1996. https://doi.org/10.1109/robot.1996.509185.
[94] Papadopoulos E, Rey DA. Force-angle measure of tipover stability margin for mobile manipulators. Veh Syst Dyn 2000. https://doi.org/10.1076/0042-3114(200001)33:1;1-5;FT029.
[95] Bergmann T. Dynamische Kipperkennung eines mobilen Serviceroboters. Hochschule Karlsruhe – Technik und Wirtschaft, 2019.
[96] Vukobratović M, Borovac B. Zero-Moment Point - Thirty Five Years of its Life. Int J Humanoid Robot 2004. https://doi.org/10.1142/s0219843604000083.
[97] Sugano S, Huang Q, Kato I. Stability criteria in controlling mobile robotic systems. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS '93), 1993. https://doi.org/10.1109/iros.1993.583186.
[98] Huang Q, Sugano S, Kato I. Stability control for a mobile manipulator using a potential method. IEEE/RSJ/GI Int. Conf. Intell. Robot. Syst., 1994. https://doi.org/10.1109/iros.1994.407542.
[99] Moosavian SAA, Alipour K. On the dynamic tip-over stability of wheeled mobile

manipulators. Int J Robot Autom 2007. https://doi.org/10.2316/journal.206.2007.4.206-3036.
[100] Moosavian SAA, Alipour K. Stability evaluation of mobile robotic systems using moment-height measure. 2006 IEEE Conf. Robot. Autom. Mechatronics, 2006. https://doi.org/10.1109/RAMECH.2006.252730.
[101] Moosavian SAA, Alipour K. Moment-Height tip-over measure for stability analysis of mobile robotic systems. IEEE Int. Conf. Intell. Robot. Syst., 2006. https://doi.org/10.1109/IROS.2006.282270.
[102] Roan PR, Burmeister A, Rahimi A, Holz K, Hooper D. Real-world validation of three tipover algorithms for mobile robots. Proc. - IEEE Int. Conf. Robot. Autom., 2010. https://doi.org/10.1109/ROBOT.2010.5509506.
[103] Huang Q, Sugano S. Manipulator motion planning for stabilizing a mobile-manipulator. IEEE Int. Conf. Intell. Robot. Syst., 1995. https://doi.org/10.1109/iros.1995.525926.
[104] Alipour K, Hasanpour A, Daemy P. Comparing two online tip-over avoidance algorithms for mobile manipulators. 2014 2nd RSI/ISM Int. Conf. Robot. Mechatronics, ICRoM 2014, 2014. https://doi.org/10.1109/ICRoM.2014.6990919.
[105] He L. Tip-over avoidance algorithm for modular mobile manipulator. Proc. 2012 1st Int. Conf. Innov. Eng. Syst. ICIES 2012, 2012. https://doi.org/10.1109/ICIES.2012.6530855.
[106] Hatano M, Obara H. Stability evaluation for mobile manipulators using criteria based on reaction. SICE 2003 Annu Conf (IEEE Cat No03TH8734) 2003.
[107] Furuno S, Yamamoto M, Mohri A. Trajectory planning of mobile manipulator with stability considerations. Proc. - IEEE Int. Conf. Robot. Autom., 2003. https://doi.org/10.1109/robot.2003.1242116.
[108] Kim J, Chung WK, Youm Y, Lee BH. Real-time ZMP compensation method using null motion for mobile manipulators. Proc - IEEE Int Conf Robot Autom 2002. https://doi.org/10.1109/ROBOT.2002.1014829.
[109] Li Y, Liu Y. Fuzzy logic self-motion planning and robust adaptive control for tip-over avoidance of redundant mobile modular manipulators. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM, 2005. https://doi.org/10.1109/aim.2005.1511187.
[110] Li Y, Liu Y. Real-time tip-over prevention and path following control for redundant nonholonomic mobile modular manipulators via fuzzy and neural-fuzzy approaches. J Dyn Syst Meas Control Trans ASME 2006. https://doi.org/10.1115/1.2229253.
[111] Brettel M, Fischer F, Bendig D, Weber A, Wolff B. Enablers for Self-optimizing Production Systems in the Context of Industrie 4.0. Procedia CIRP 2016;41:93–8. https://doi.org/10.1016/j.procir.2015.12.065.
[112] Seemann S. Entwicklung und Simulation einer Strategie zur Stabilisierung eines Knickarmroboters auf einer mobilen Plattform. Fakultät Elektro- und Informationstechnik Masterstudiengang Elektro- und Informationstechnik, 2018.
[113] Kipfmueller M, Toledo Fuentes A, Seemann S, Prieto J. Simulation of stabilization strategies for industrial robots on mobile platforms. XXV Semin. Anu. Automática, Electrónica Ind. e Instrumentación Libr. Actas. Semin. Anu. automática, electrónica Ind. e instrumentación (SAAEI 2018), 2018, p. 221–6.
[114] LEIFIphysik. Drehbewegungen n.d. https://www.leifiphysik.de/mechanik/drehbewegungen (accessed February 20, 2022).
[115] Helmer P, Heemann P. Entwicklung und Auslegung eines Gyrostabilisers (Mid-term Project). Karlsruhe: 2018.
[116] Seakeeper | Eliminate Boat Roll n.d. https://www.seakeeper.com/ (accessed January 21, 2022).
[117] Giallanza A, Elms T. Interactive roll stabilization comparative analysis for large yacht: gyroscope versus active fins. Int J Interact Des Manuf 2020;14:143–51.

https://doi.org/10.1007/s12008-019-00618-y.
[118] Webhofer M. Modellierung eines Mehrkörpersystems zur Simulation der Querpendelbewegung von Einseilumlaufbahnen bei der Stationseinfahrt. Technical University of Munich (TUM), 2000.
[119] Rahnejat H, Rothberg S. Multi-body Dynamics: Monitoring and Simulation Techniques III. Wiley; 2004.
[120] Bauchau OA. Flexible Multibody Dynamics. Springer Netherlands; 2012.
[121] What is mechatronic system simulation? n.d. https://community.sw.siemens.com/s/article/what-is-mechatronic-system-simulation (accessed October 15, 2022).
[122] Co-simulation with Abaqus and Dymola n.d. https://www.3ds.com/products-services/simulia/training/course-descriptions/co-simulation-with-abaqus-and-dymola/ (accessed March 28, 2022).
[123] Krauße M. Aufwandsoptimierte Simulation von Produktionsanlagen durch Vergrößerung der Geltungsbereiche von Teilmodellen. Karlsruhe Institute of Technology, 2014. https://doi.org/978-3-8440-2799-0.
[124] Blundell M, Harty D. The Multibody Systems Approach to Vehicle Dynamics. Elsevier Science; 2014.
[125] Vöth S. Dynamik schwingungsfähiger Systeme: Von der Modellbildung bis zur Betriebsfestigkeitsrechnung mit MATLAB/SIMULINK®. Vieweg+Teubner Verlag; 2007.
[126] Schumacher A. Optimierung mechanischer Strukturen: Grundlagen und industrielle Anwendungen. Springer Berlin Heidelberg; 2013.
[127] National Technology and Engineering Solutions of Sandia L. Documentation | Dakota n.d. https://dakota.sandia.gov/documentation.html (accessed December 23, 2021).
[128] Dakota Reference Manual: nl2sol n.d. https://dakota.sandia.gov/sites/default/files/docs/6.0/html-ref/method-nl2sol.html (accessed October 16, 2022).
[129] Adams, B. M.; Ebeida, M. S.; Eldred, M. S.; Jakeman, J. D.; Swiler, L. P.; Stephens, A.; Vigil, D. M.; Wildey TM. Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 Theory Manual. Albuquerque, New Mexico: 2014.
[130] Dakota Reference Manual: package_scolib n.d. https://dakota.sandia.gov/sites/default/files/docs/6.0/html-ref/topic-package_scolib.html (accessed October 16, 2022).
[131] Portje V. Methode zur computerunterstützten Parametrierung von Mehrkörpermodellen. Hochschule Karlsruhe - Fakultät für Maschinenbau und Mechatronik, 2018.
[132] Bhave S. Mechanical Vibrations: Theory and Practice. Dorling Kindersley; 2010.
[133] Mohr M, Portje V. Parametrierung eines Mehrkörpersystems mit Open Source Software Dakota. 2016.
[134] Schenk A. Schwingungsidentiikation von Schienenfahrzeugen: Ziele, Methodik und Nutzen. VDI-Schwingungstagung Exp. und Rechn. Modalanalyse sowie Identifikation dynamischer Syst., Kassel, Germany: 2000.
[135] Mohr M, Portje V. Methode zur Parametrierung von Mehrkörpersystemen mittels Open-source Optimierungssoftware Dakota. 2017.
[136] Füllekrug U. Computation of real normal modes from complex eigenvectors. Mech Syst Signal Process 2008;22:57–65.
[137] Schilders WH, van der Vorst HA, Rommes J. Model Order Reduction: Theory, Research Aspects and Applications. Springer Berlin Heidelberg; 2008.
[138] Shampine LF, Thompson S. Stiff systems. Scholarpedia 2007;2:2855. https://doi.org/10.4249/SCHOLARPEDIA.2855.

[139] MSC Software Corporation. Adams Solver User's Guide. MSC Software Corporation; 2021.
[140] Richter T, Wick T. Einführung in die Numerische Mathematik: Begriffe, Konzepte und zahlreiche Anwendungsbeispiele. Springer Berlin Heidelberg; 2017.
[141] Bärwolff G. Numerik für Ingenieure, Physiker und Informatiker. Springer Berlin Heidelberg; 2020.
[142] Negrut D, Dyer A. ADAMS/Solver Primer. 2004.
[143] Hexagon. Adams/View Overview 2021. https://simcompanion.hexagon.com/customers/s/article/adams-view-help---adams-2014-doc10647 (accessed February 16, 2021).
[144] Schröder D. Elektrische Antriebe - Regelung von Antriebssystemen. Springer Berlin Heidelberg; 2015.
[145] Keviczky L, Bars R, Hetthéssy J, Bányász C. Control Engineering. Springer Nature Singapore; 2018.
[146] Gattringer H. Starr-elastische Robotersysteme: Theorie und Anwendungen. Springer Berlin Heidelberg; 2011.
[147] Sick AG. Betriebsanleitung S300 Sicherheits-Laserscanner 2013:156. https://www.sick.com/media/pdf/5/95/595/IM0017595.PDF (accessed December 15, 2021).
[148] Toledo Fuentes A, Kipfmüller M, Burghart C, José Prieto MÁ, Bertram T, Bryg M, et al. Stable operation of arm type robots on mobile platforms. 14th CIRP Conf. Intell. Comput. Manuf. Eng. CIRP ICME '20, Italy, 2020, p. 104–10.
[149] Möllmann S. Integration of an Inertial Measurement Unit into the autonomous vehicle platform CampusBot (in German). 2014.
[150] Neugebauer, Reimund; Kolouch, M.; Richter, M.; Schulten M. Fehlerquellen bei einer Modalanalyse: Untersuchung von Einflussfaktoren während der praktischen Durchführung Sources of errors at conducting an experimental modal analysis. Wt Werkstattstech Online 99 2009:889–94.
[151] Fuentes A, Kipfmueller M, Prieto M. 6 DOF articulated-arm robot and mobile platform: Dynamic modelling as Multibody System and its validation via Experimental Modal Analysis. IOP Conf Ser Mater Sci Eng 2017;257:12008. https://doi.org/10.1088/1757-899X/257/1/012008.
[152] Gross D, Hauger W, Schröder J, Wall WA, Govindjee S. Engineering Mechanics 3: Dynamics. Springer Berlin Heidelberg; 2014.
[153] Stewart DE. Dynamics with Inequalities: Impacts and Hard Constraints. Society for Industrial and Applied Mathematics; 2011.
[154] Hexagon Manufacturing Intelligence Inc. About Adams/Solver. Hexagon Manufacturing Intelligence Inc.; 2011.
[155] Devore JL, Farnum NR, Doi JA. Applied Statistics for Engineers and Scientists. Cengage Learning; 2013.
[156] Kipfmueller M, Munzinger C. Efficient Simulation of Parallel Kinematic Machine Tools. Vol. 7 33rd Mech. Robot. Conf. Parts A B, San Diego: ASMEDC; 2009, p. 523–30. https://doi.org/10.1115/DETC2009-86769.
[157] Vinayak H, Singh R. MULTI-BODY DYNAMICS AND MODAL ANALYSIS OF COMPLIANT GEAR BODIES. J Sound Vib 1998;210:171–214.
[158] Bremer H. Elastic Multibody Dynamics: A Direct Ritz Approach. Springer Netherlands; 2008.
[159] Ewins DJ. Modal Testing: Theory, Practice and Application. Wiley; 2009.
[160] Lutz H, Wendt W. Taschenbuch der Regelungstechnik: mit MATLAB und Simulink. Verlag Europa-Lehrmittel Nourney, Vollmer; 2014.
[161] Maedler North America. Linear Drives (lifting devices) SFL 12 V - 24 V n.d.

http://smarthost.maedler.de/datenblaetter/K42_800_EN.pdf?_ga=2.164682360.651806919.1663001258-1001102722.1663001258 (accessed September 12, 2022).
[162] Unknown. Technical Data n.d. http://smarthost.maedler.de/datenblaetter/SFL_Kennlinien.pdf?_ga=2.164682360.651806919.1663001258-1001102722.1663001258 (accessed September 12, 2022).
[163] Drela M, Astro A&. Second-Order DC Electric Motor Model 2006.
[164] Product 47520114 - Linear drive SFL with DC-motor operating voltage 12V-24V nominal lifting force 400N with Hall-IC n.d. https://www.maedler.de/Article/47520114 (accessed April 6, 2022).
[165] Berufsbildende Schulen Wolfsburg. Regelungstechnik: Vergleich und Dimensionierung 2020. https://www.xplore-dna.net/mod/page/view.php?id=95 (accessed March 28, 2021).
[166] Nelles O. Regelungstechnik (Script). Univ Siegen n.d. https://www.mb.uni-siegen.de/mrt/lehre/rt/rt_skript.pdf (accessed May 21, 2021).
[167] Kuhn U. Eine praxisnahe Einstellregel für PID-Regler: die T-Summen-Regel. Autom. Prax., 1995.
[168] Zacher S. Hinweise zur Identifikation einer Regelstrecke mit MATLAB nach Versuchsdaten. Autom Nr 3 2011:10.
[169] Maxon Group. Catalog page DCX35L n.d. https://www.maxongroup.de/medias/sys_master/root/8881024892958/EN-21-106.pdf (accessed April 6, 2022).
[170] Gerdt Seefrid GmbH. Motor 627.031 DCGM 77 T72 n.d. https://products.zilvertron.com/files/seefrid/Datasheet/627031.pdf (accessed April 6, 2022).
[171] Hochschule Karlsruhe. Systemtheorie Online: IT1-Glied n.d. https://www.eit.hs-karlsruhe.de/mesysto/teil-a-zeitkontinuierliche-signale-und-systeme/uebertragungsglieder-der-regelungstechnik/zusammengesetzte-uebertragungsglieder/it1-glied.html (accessed April 28, 2022).
[172] Tietze U, Schenk C. Halbleiter-Schaltungstechnik. Springer Berlin Heidelberg; 2013.
[173] Kessler R. Weglose Waage: Simulation Einstellregeln von Tietze-Schenk nd von Ziegler-Nichols (Sensorsystemtechnik). Karlsruhe: n.d.
[174] Spreckelsen NT. Co-Simulation eines Gyroskops als Stabilisierungsmechanismus für ein mobiles Robotersystems. 2021.
[175] Kempf F. Development of an algorithm for active stabilization of a mobile manipulator. Hochschule Karlsruhe – Technik und Wirtschaft, 2021.
[176] Fraunhofer IPA. GitHub - ipa320/schunk_modular_robotics 2019. https://github.com/ipa320/schunk_modular_robotics (accessed October 16, 2021).
[177] Grote KH, Feldhusen J. Dubbel: Taschenbuch für den Maschinenbau. Springer Berlin Heidelberg; 2011.
[178] Fok C-L, Johnson G, Sentis L, Mok A, Yamokoski JD. ControlIt! | A Software Framework for Whole-Body Operational Space Control. Int J Humanoid Robot 2015;13:1550040. https://doi.org/10.1142/S0219843615500401.
[179] GitHub - hakuturu583/robot_kinetics_pkgs: ROS packages about robot kinetics(get center of gravity etc..) n.d. https://github.com/hakuturu583/robot_kinetics_pkgs (accessed October 17, 2021).
[180] KDL wiki | The Orocos Project n.d. https://www.orocos.org/kdl.html (accessed October 17, 2021).
[181] Smits R. orocos_kdl: KDL::ChainIdSolver_RNE Class Reference n.d. http://docs.ros.org/en/kinetic/api/orocos_kdl/html/classKDL_1_1ChainIdSolver__RNE.html (accessed October 17, 2021).
[182] Murray R, Li Z, Sastry S. A mathematical introduction to robotic manipulation Cited by

me analytic_grasp_synt... grasp_quality_metrics. vol. 29. CRC Press; 1994.
[183] Siciliano B, Khatib O. Springer Handbook of Robotics. 2nd ed. Switzerland: Springer International Publishing; 2016. https://doi.org/10.1007/978-3-319-32552-1_1.
[184] Zhao J, Feng Z, Chu F, Ma N. Workspace of the End Effector of a Robot Mechanism. Adv Theory Constraint Motion Anal Robot Mech 2014:159–200. https://doi.org/10.1016/B978-0-12-420162-0.00005-9.
[185] Maier H. Grundlagen der Robotik. Vde Verlag GmbH; 2019.
[186] Pobil AP, Serna MA. Spatial Representation and Motion Planning. Springer Berlin Heidelberg; 1995.
[187] PickNik Robotics. MoveIt Setup Assistant — moveit_tutorials Kinetic documentation n.d. http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/setup_assistant/setup_assistant_tutorial.html (accessed October 17, 2021).
[188] PickNik Robotics. Move Group C++ Interface — moveit_tutorials Kinetic documentation n.d. http://docs.ros.org/en/kinetic/api/moveit_tutorials/html/doc/move_group_interface/move_group_interface_tutorial.html (accessed October 17, 2021).
[189] Company SR. Planners Benchmarking Documentation Shadow Robot Company 2020.
[190] Moll M, Sucan IA, Kavraki LE. Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization. IEEE Robot Autom Mag 2015. https://doi.org/10.1109/MRA.2015.2448276.
[191] Khalil W, Dombre E. Chapter 1 - Terminology and general definitions. In: Khalil W, Dombre E, editors. Model. Identif. Control Robot., Oxford: Butterworth-Heinemann; 2002, p. 1–12. https://doi.org/https://doi.org/10.1016/B978-190399666-9/50001-4.
[192] Einhorn E, Langner T, Stricker R, Martin C, Gross H-M. MIRA - middleware for robotic applications. 2012 IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2012, p. 2591–8. https://doi.org/10.1109/IROS.2012.6385959.
[193] Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, et al. ROS: an open-source Robot Operating System. ICRA Work. Open Source Softw., vol. 3, 2009.
[194] GitHub - strands-project/scitos_drivers: Scitos G5 drivers that interface ROS to MIRA n.d. https://github.com/strands-project/scitos_drivers (accessed January 21, 2022).
[195] Schunk GmbH & Co. KG. Modulare und Mobile Greifsysteme n.d.
[196] MetraLabs GmbH. Betriebsanleitung (Original) 2014.
[197] MIRA: Concepts n.d. https://www.mira-project.org/MIRA-doc/ConceptsPage.html (accessed October 16, 2021).
[198] schunk_robots - ROS Wiki. Open Robot n.d. http://wiki.ros.org/schunk_robots (accessed October 23, 2021).
[199] GitHub - ipa320/schunk_robots n.d. https://github.com/ipa320/schunk_robots (accessed October 23, 2021).
[200] Bryg M. Kopplung einer mobilen Roboterplattform mit einem Industrieroboterarm Bachelorarbeit (B . Eng .). Hochschule Karlsruhe Technik und Wirtschaft, 2018.
[201] Mobile Roboter - KUKA AG n.d. https://www.kuka.com/de-de/produkte-leistungen/mobilität/mobile-roboter (accessed May 5, 2022).
[202] RB-KAIROS Mobile Robot with Omnidirectional Wheels - Robotic Gizmos n.d. https://www.roboticgizmos.com/rb-kairos-mobile-robot/ (accessed May 5, 2022).
[203] Industrial mobile robots for material flow and intralogistics by NEOBOTIX n.d. https://www.expo21xx.com/industrial-robots/18488_st3_service-robots/default.htm (accessed May 5, 2022).
[204] MuR205 - Mobile Universal Robot with Whole Body Control and Redundancy Resolution - YouTube n.d. https://www.youtube.com/watch?v=v1bzNrgP8kg (accessed May 5, 2022).

[205] Global Robotic Standards from A3 Robotics. Robot Ind Assoc 2020. https://www.automate.org/a3-content/global-robotic-standards (accessed March 11, 2021).
[206] Technologisches Zentrum - ROBOTEC n.d. https://robotec.sk/de/technologisches-zentrum-2/ (accessed August 1, 2022).
[207] Freight100 OEM Base - Fetch Robotics n.d. https://fetchrobotics.com/freight100-oem-base/ (accessed August 1, 2022).
[208] SCITOS G5. - MetraLabs n.d. https://www.metralabs.com/en/mobile-robot-scitos-g5/ (accessed August 1, 2022).
[209] Gasch R, Knothe K, Liebich R. Strukturdynamik: Diskrete Systeme und Kontinua. Springer Berlin Heidelberg; 2012.
[210] How to calculate damping from a FRF? n.d. https://community.sw.siemens.com/s/article/how-to-calculate-damping-from-a-frf (accessed November 7, 2022).
[211] Mohr M. Mehrkörpermodell einer mobilen Roboterplattform und zugehörige Validierung mittels Modalanalyse. Karlsruhe University of Applied Sciences, 2015.
[212] Mechanical Dynamics I. ADAMS: Getting Started Using ADAMS/View : Version 10. Mechanical Dynamics, Incorporated; 1999.
[213] Basic ADAMS Full Simulation Training Guide. Mechanical Dynamics, Incorporated; 2001.
[214] Building Models in ADAMS/View. Ann Arbor, Michigan: Mechanical Dynamics, Incorporated; 2000.
[215] Graf R, Vierling R, Dillmann R. A flexible controller for a Stewart platform. In: Jain LC, Jain RK, editors. Knowledge-Based Intell. Electron. Syst. 2nd Int. Conf. {KES} 1998, Adelaide, South Aust. 21-23 April 1998, Proceedings, Part {II}, IEEE; 1998, p. 52–9. https://doi.org/10.1109/KES.1998.725892.

9 List of Figures

10 List of Tables

11 Annex

A.1 DAKOTA files that conform the parametrization algorithm employed for the optimization of the MBS of the robot manipulator [131]. The constants indicated as `0.0` need to be adjusted with appropriate values depending on the system to be analyzed.

Settings for DAKOTA gradient based algorithm:
dakota_opt_settings.in [131]

```
environment
tabular_graphics_data
tabular_graphics_file = 'result_dakota.dat'
results_output

method
nl2sol
convergence_tolerance = 1e-3
output debug

model
single

variables
continuous_design  = 24
initial_point       0.0    0.0    0.0    0.0    0.0
lower_bounds        0.0    0.0    0.0    0.0    0.0
upper_bounds        0.0    0.0    0.0    0.0    0.0
descriptors        '<c1>' '<c2>' '<c3>' '<c4>' '<c5>'

interface
system
analysis_driver = 'run.bat'
parameters_file = 'params.in'
results_file    = 'results.out'
aprepro

responses
calibration_terms        = 8
calibration_data_file = 'calibration.dat'
freeform
weights               = 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
numerical_gradients
method_source Dakota
interval_type forward
fd_gradient_step_size = .01
no_hessians
```

Settings for DAKOTA evolutionary algorithm:
rosen_opt_ea.in [131]

```
environment
graphics
tabular_data
tabular_data_file = ’rosen_opt_ea.dat’

method
```

```
coliny_ea
max_iterations           = 100
max_function_evaluations = 2000
seed                     = 11011011
population_size          = 50
fitness_type             merit_function
mutation_type            offset_normal
mutation_rate            1.0
crossover_type           two_point
crossover_rate           0.0
replacement_type chc     = 10

model
single

variables
continuous_design = 2
lower_bounds       0.0  0.0
upper_bounds       0.0  0.0
descriptors      '<c1>' '<c2>'

interface
analysis_drivers  = ’rosenbrock’
direct

responses
objective_functions = 1
```

Section of script that corresponds to a MBS model generated by MSC.Adams/View and tagged with the variables to be parametrized:
template.adm [131]

```
!---------------------------------- FORCES ------------------------------------
!
!                   adams_view_name='BUSHING_1_Base_Shoulder'
BUSHING/1
, I = 151
, J = 152
, C = 0.001, 0.001, 0.001
, K = 2.42037E+06, 2.41481E+06, 2.46091E+06
, CT = 5.729577951, 5.729577951, 5.729577951
, KT = 4.583662361E+09, 4.583662361E+08, 4.583662361E+08
!
!                   adams_view_name='BUSHING_2_Shoulder_UpperArmLeft'
BUSHING/2
, I = 153
, J = 154
, C = 0.001, 0.001, 0.001
, K = <c1>, <c2>, <c3>
, CT = 5.729577951, 5.729577951, 5.729577951
, KT = <c4>, <c5>, <c6>
!
!                  adams_view_name='BUSHING_3_Shoulder_UpperArmRight'
BUSHING/3
, I = 155
, J = 156
, C = 0.001, 0.001, 0.001
, K = <c7>, <c8>, <c9>
, CT = 5.729577951, 5.729577951, 5.729577951
, KT = <c10>, <c11>, <c12>
!
!                 adams_view_name='BUSHING_4_UpperArmRight_Elbow'
BUSHING/4
```

```
, I = 157
, J = 158
, C = 0.001, 0.001, 0.001
, K = <c13>, <c14>, <c15>
, CT = 5.729577951, 5.729577951, 5.729577951
, KT = <c16>, <c17>, <c18>
!
!                  adams_view_name='BUSHING_5_UpperArmLeft_Elbow''
BUSHING/5
, I = 159
, J = 160
, C = 0.001, 0.001, 0.001
, K = <c19>, <c20>, <c21>
, CT = 5.729577951, 5.729577951, 5.729577951
, KT = <c22>, <c23>, <c24>
!
```

Settings for parametrization criteria:
settings.py [131]

```
import numpy as np

np.savez('settings',
damping         = "off",           # "on" / "off"
penalfunction   = "off",           # "on" / "off"
measpoint_ema   = [0.0,0.0,0.0],   # PART/1, PART/3, PART/5 -> [1,3,5...]
measpoint_adams = [0.0,0.0,0.0],   # PART/1, PART/3, PART/5 -> [1,3,5...]
accelerometerDOF= ["v1","v2","v3"] # for 3D "v1"=x;"v2"=y;"v3"=z
)
```

Penal function:
penalfunction.py [131]

```
from parameter import *
import numpy as np
import re

A = [ [ 1 ,-1 , 0],
      [ 0 ,-1 , 1],
      [-1 , 0 , 1], ] # linear constraints

Exponential_gain = 10
Linear_gain      = 1

def penalfunction():
matrix_A = np.array(A)
parameter = []
parameter_dict = getParameter()
for key, initial in parameter_dict.items(): parameter.append(parameter_dict[key])
parameter = np.array(parameter)
delta = np.dot(matrix_A,parameter)
for i in range(len(delta)):
if delta[i] < 0: delta[i] = 0
return Linear_gain*(np.linalg.norm(delta))**Exponential_gain
```

DAKOTA pre-processing:
preprocessor.py [131]

```
import parameter as par
import re
import os

dir  = os.path.dirname(__file__)
filetemplate = os.path.join(dir, '../ADAMS/template.adm')
fileberechnung = os.path.join(dir, '../ADAMS/calculation.adm')

fin  = open(filetemplate,"r")
fout = open(filecalculation,"w")

parameter = par.getParameter()
for line in fin:
if line.find("<") == -1:
fout.write(line)
else:
fout.write(par.swapTags(line,parameter))
```

Searching tagged parameters in MBS model:
parameter.py [131]

```
import re
import math
import numpy as np
import os

dir = os.path.dirname(__file__)
fileparams = os.path.join(dir, '../params.in')

regex_taggedpar = "[-+]?\d+[\.]?\d*[eE]?[-+]?\d*"

def getParameter():
fpar = open(fileparams,"r")
parameter_dict = dict()
for line in fpar:
if not line.find("{ <c") == -1:
a = re.findall(regex_taggedpar, line)[0]
b = re.findall(regex_taggedpar, line)[1]
parameter_dict["<c"+str(a)+">"] = float(b)
if not line.find("{ <d") == -1:
a = re.findall(regex_taggedpar, line)[0]
b = re.findall(regex_taggedpar, line)[1]
parameter_dict["<d"+str(a)+">"] = float(b)
if not line.find("{ <E>") == -1:
a = re.findall(regex_taggedpar, line)[0]
parameter_dict["<E>"] = round(float(a),2)
else:
if not line.find("{ <E") == -1:
a = re.findall(taggedpar, line)[0]
b = re.findall(taggedpar, line)[1]
parameter_dict["<E"+str(a)+">"] = float(b)
fpar.seek(0)
return parameter_dict

def getfinParameter(file):
initline = 1
parameter_dict = dict()
f1_oldvalue = float('inf')
```

```
for line in file:
if initline == 1:
tags = re.findall("<.\d*.>",line)
isea = re.findall("(f1).?",line)
initline = 0
if len(isea) == 0:
ea = 0
isea.insert(0,0)
if isea[0] == "f1":
ea = 1
continue
if ea == 0:
parameterline = re.findall(taggedpar,line)
if ea == 1:
new = re.findall(taggedpar,line)
if float(neu[-1]) < f1_wert_alt:
parameterline = neu
f1_oldvalue = float(new[-1])
for i in range(len(tags)):
parameter_dict[tags[i]] = float(parameterline [1+i])
file.seek(0)
return parameter_dict

def swapTags(line,parameter):
for key, initial in parameter.items():
if not line.find(key) == -1:
if not line.find("KT") == -1:
initial = initial*180/math.pi
line = line.replace(key, str(initial))
return line
```

DAKOTA pre-processing:
preprocessor.py [131]

```
from result_extract import *
from mac import *
from parameter import *
from penalfunction import *
import numpy as np
import os

dir  = os.path.dirname(__file__)
fileresadams = os.path.join(dir, '../ADAMS/result_adams.txt')
fileresout = os.path.join(dir, '../results.out')
fin  = open(fileresadams,"r")
fout = open(fileresout,"w")

frequency = getFrequency(fin)
for i in range(len(frequency)):
fout.write(str(frequency[i])+"\n")

if np.load('../Python/settings.npz')["damping"] == "on":
dampingratio = getDamping(fin)
for i in range(len(dampingratio)):
fout.write(str(dampingratio[i])+"\n")

moden_sim = getMode_xyz(fin)
moden_exp = np.load('../OROS/modaldata.npz')["mode"]

mac_diag = np.diag(MAC(moden_exp,moden_sim))
for i in range(len(mac_diag)):
```

```
fout.write(str(mac_diag[i])+"\n")

if np.load('../Python/settings.npz')["penalfunction"] == "on":
fileIteration = os.path.join(dir, '../Iteration.out')
fite = open(fileIteration,"a")
penalterm =  penalfunction ()
fout.write(str(penalterm))
fite.write(str(penalterm)+"\n")
```

Extract parameters from MBS model:
result_extract.py [131]

```
import parameter as par
import re
import numpy as np

regex_taggedpar = "[-+]?\d+[\.]?\d*[eE]?[-+]?\d*"

def getFrequency(file):
freq = []
for line in file:
if not line.find("Undamped natural freq.") == -1:
freq.append(float(re.findall(regex_taggedpar,line)[0]))
file.seek(0)
if np.load('../Python/settings.npz')["damping"] == "on":
damp = getDamping(file)
for i in range(len(freq)):
freq[i] = freq[i]*np.sqrt(1-(damp[i]/100)**2)
return freq

def getDamping(file):
damp = []
for line in file:
if not line.find("Damping ratio") == -1:
damp.append(float(re.findall(regex_taggedpar,line)[0])*100)
file.seek(0)
return damp

def getMode(file):
mode = dict()
mode_red = []
mode_matrix = []
oldposition = 0
for line in file:
if not line.find("PART/") == -1:
position = int(re.findall(regex_taggedpar,line)[0])
if position < oldposition:
for j in np.load('../Python/settings.npz')["measpoint_adams"]:
mode_red.append(mode[str(j)])
mode_matrix.append(mode_red[:])
mode_red.clear()
mode[str(position)]= re.findall(regex_taggedpar,line)
oldposition = 0
else:
mode[str(position)] = re.findall(regex_taggedpar,line)
oldposition = position
for j in np.load('../Python/settings.npz')["measpoint_adams"]:
mode_red.append(mode[str(j)])
mode_matrix.append(mode_red[:])
file.seek(0)
return mode_matrix
```

```
def scaling(mode):
for i in range(len(mode)):
max = np.amax(mode[i])
min = np.amin(mode[i])
if abs(min) > max:
max = abs(min)
for j in range(len(mode[i])):
if max == 0:
max = 1
mode[i][j] = mode[i][j]/max
return mode

def getMode_xyz(file):
spatialdir = np.load('settings.npz')["Spatial directions"]
print(len(spatialdir))
if len(spatialdir) == 2:
return print("Error: only one spatial direction!")
mode_matrix = getMode(file)
row = []
mode_xyz = []
mode_1D = []
out = 0
def P2R(radian, angles):
return radian * np.exp(1j*angles*np.pi/180)
for mode in mode_matrix:
for line in mode:
x = P2R(float(line[1]), float(line[2]))
y = P2R(float(line[3]), float(line[4]))
z = P2R(float(line[5]), float(line[6]))
if len(spatialdir) == 3:
row.extend([x,y,z])
else:
if spatialdir == "v1":
row.append(x)
if spatialdir == "v2":
row.append(y)
if spatialdir == "v3":
row.append(z)
if len(spatialdir) == 3:
mode_xyz.append(row[:])
row.clear()
else:
mode_1D.append(row[:])
row.clear()
if len(spatialdir) == 3:
out = scaling(np.array(mode_xyz))
else:
out = scaling(np.array(mode_1D))
file.seek(0)
return out
```

Compute MAC:
mac.py [131]

```
import numpy as np
import matplotlib.pyplot as plt

def mac(phi1,phi2):
phi1_c = np.conjugate(phi1)
phi1_c = np.conjugate(phi2)
mac_ij = abs(np.dot(phi1_c,phi2))**2/(np.dot(phi1_c,phi1)*np.dot(phi2_c,phi2))
```

```
mac_ij = mac_ij.real
if not mac_ij.imag == 0 :
print("Error: MAC exhibits imaginary part")
return mac_ij

def MAC(exp,sim):
mac_ij = np.zeros((len(exp),len(sim)))
for i in range(len(exp)):
for j in range(len(sim)):
a = np.squeeze(np.asarray(exp[i]))
b = np.squeeze(np.asarray(sim[j]))
mac_ij[i][j] = mac(a,b)
return mac_ij
```

Supply references:
uff.py [131]

```
import numpy as np
import matplotlib.pyplot as plt
import pyuff
# https://github.com/openmodal/pyuff/blob/master/pyuff%20Showcase.ipynb
import glob
import imp

uffdata = []
for filename in glob.glob('../OROS/*.unv'):
uffdata.append(pyuff.UFF(filename).read_sets())

spatialdir = np.load('settings.npz')["Spatial direction"]

def getFreq(data):
freq = data["eig"].imag/(2*np.pi)
return freq

def getDamp(data):
a = abs(data["eig"].real/data["eig"].imag)
damp = a/np.sqrt(a**2+1)
return damp

def getEigenvalue(data):
eigen = data["eig"]
return eigen

def getMode(data):
if len(spatialdir) == 1:
sp = [data[spatialdir [0]]]
if len(spatialdir) == 2:
sp = [data[spatialdir [0]], data[spatialdir [1]]]
if len(raumrichtung) == 3:
sp = [data[spatialdir [0]], data[spatialdir [1]], data[spatialdir [2]]]
node_nums = data["node_nums"]
node_nums = list(map(int,node_nums))
mode = dict()
for i in range(len(node_nums)):
if len(spatialdir) == 1:
mode[str(node_nums[i])] = sp[0][i]
if len(spatialdir) == 2:
mode[str(node_nums[i])] = [sp[0][i], sp[1][i]]
if len(spatialdir) == 3:
mode[str(node_nums[i])] = [sp[0][i], sp[1][i], sp[2][i]]
return mode
```

```
def getAllModaldata (data):
dict = {"freq":getFreq(data),"damp":getDamp(data),"eigen":getEigenvalues(data),
        "mode":getMode(data)}
return dict

def getMatrixModaldata(data):
Modaldata = []
for i in range(len(data)):
if isinstance(data[i], dict) == True:
Modaldata.append(getAllModaldata(data[i]))
else:
for j in range(len(data[i])):
Modaldata.append(getModaldata_i(data[i][j]))
Modaldata.sort(key=lambda x:x["freq"])
return Modaldata

def scaling(mode):
for i in range(len(mode)):
max = np.amax(mode[i])
min = np.amin(mode[i])
if abs(min) > max:
max = abs(min)
for j in range(len(mode[i])):
if max == 0:
max = 1
mode[i][j] = mode[i][j]/max
return mode

def ModalParameter(data):
Modaldata = getMatrixModaldata(data)
freq = []
damp = []
eigenvalue = []
mode = []
for i in range(len(Modaldata)):
freq.append(Modaldata[i]["freq"])
damp.append(100*Modaldata[i]["damp"])
eigenvalue.append(Modaldata[i]["eigen"])
datamode = []
for j in np.load('settings.npz')["measpoint_ema"]:
if len(spatialdir) == 1:
datamode.append(Modaldata[i]["mode"][str(j)])
else:
for element in Modaldata[i]["mode"][str(j)]:
datamode.append(element)
mode.append(datamode[:])
mode = normiere(mode)
return [freq,damp,eigenwert,mode]
```

Mode transformation:
modetrans.py [131]

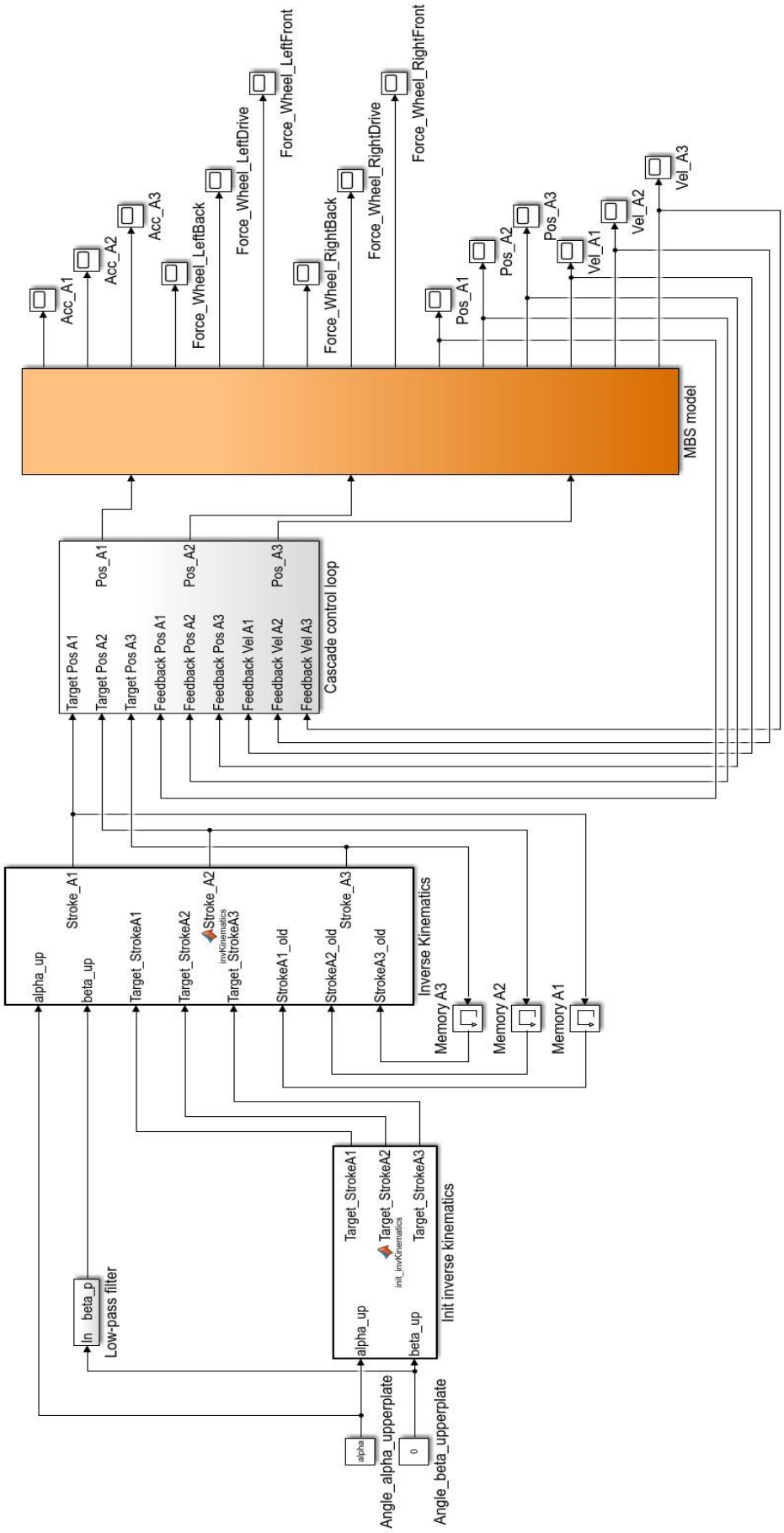
```
import numpy as np
import uff
import scipy.io

def eigen2n(eig):
eig2n = []
for i in range(len(eig)):
eig2n.append(eig[i])
```

```
eig2n.append(np.conjugate(eig[i]))
return eig2n

def mode2n(mod):
modcon = []
for i in mod:
modi = []
for j in i:
j = np.conjugate(j)
modi.append(j)
modcon.append(modi[:])
mod2n = []
for i,j in zip(mod,modcon):
mod2n.append(i)
mod2n.append(j)
return mod2n

def sort(f,x):
sort = []
j = 0
for i in f:
sort.append([i,x[j]])
j+=1
sort.sort(key=lambda x:x[0]) # nach der groesse der Frequenzen sortieren
f = []
x = []
for i in sort:
f.append(i[0])
x.append(i[1])
x = np.transpose(x)
return [f,x]

def transundampmodalparameter2damped(eigenvalue,X):
eigen = eigen2n(eigenvalue)
X = mode2n(X)
n_moden = int(len(X)/2)
n_elemente = int(len(X[0]))
X = np.transpose(X)
U, s, V = np.linalg.svd(np.real(X), full_matrices=True)
X_red = np.dot(np.transpose(U),X)
X_red = X_red[0:n_moden]

eigenX_red = np.multiply(X_red,eigen)
lhs = np.concatenate((X_red, eigenX_red), axis=0)
rhs = -1*np.multiply(X_red,np.array(eigen)**2)
MCMD = np.real(np.dot(rhs,np.linalg.inv(lhs)))
MC = np.hsplit(MCMD,2)[0]

eigen,X_red = np.linalg.eig(MC)

freq = np.sqrt(eigen)/(2*np.pi)
X_red = np.concatenate((X_red, np.zeros((n_elemente-n_moden,n_moden))), axis=0)

X = np.dot(U,X_red)
X = uff.normiere(np.transpose(X))
freq, X  = sortieren(freq,X)
X = np.transpose(X)
return [freq,X]
```

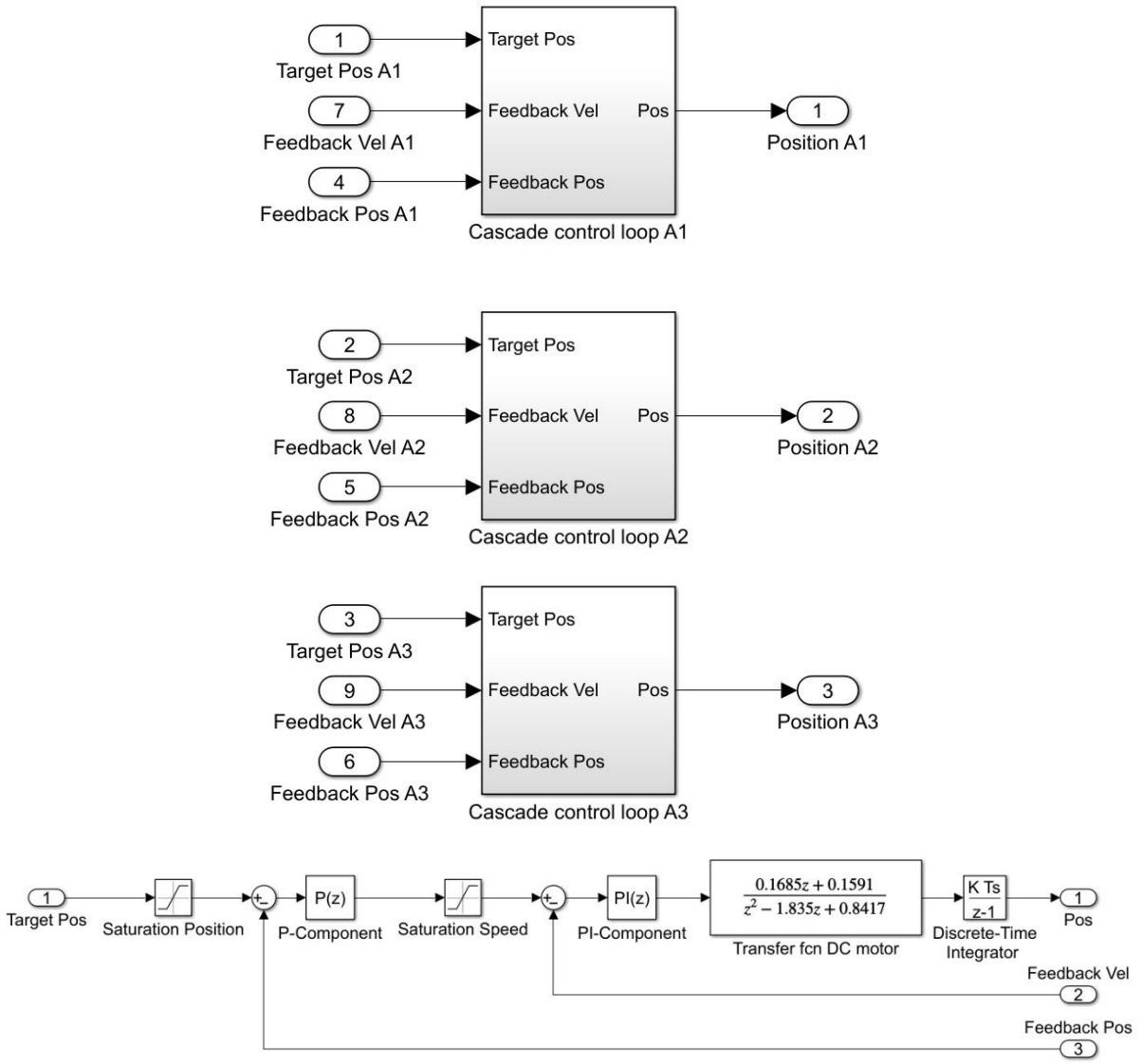
Create calibration.dat:
reference.py [131]

```
import uff
import modentrans as mt
import numpy as np
import os

dir = os.path.dirname(__file__)
filecalibration = os.path.join(dir, '../calibration.dat')
fcal = open(filecalibration,"w")

freq, damp, eigenvalue, mode = uff.ModalParameter(uff.uffdata)
if np.load('settings.npz')["damping"] == "off":
freq,mode = mt. transundampmodalparameter2damped(eigenvalue,mode)
np.savez('../OROS/modaldata', freq=freq, mode=mode)
for f in freq:
fcal.write(str(f)+"\n")
for m in mode:
fcal.write(str(1.0)+"\n")
if np.load('settings.npz')["damping"] == "an":
freq_undamp = []
for f,d in zip(freq,damp):
f = f/np.sqrt(1-(d/100)**2)
freq_undamp.append(f)
np.savez('../OROS/modaldata', freq=freq_o, damp=damp, mode=mode)
for f in freq:
fcal.write(str(f)+"\n")
for d in damp:
fcal.write(str(d)+"\n")
for m in mode:
fcal.write(str(1.0)+"\n")

if np.load('settings.npz')["penalfunction"] == "on":
fcal.write(str(0.0))

fileacf = os.path.join(dir, '../ADAMS/run_adams.acf')
fdat = open(fileacf,"w")
fdat.write("calculation.adm"+"\n")
fdat.write("result_adams"+"\n")
if np.load('settings.npz')["damping"] == "on":
fdat.write("LINEAR/EIGENSOL, COORDS=1,"+str(len(freq))+",TABLE_PRECISION = 3")
else:
fdat.write("LINEAR/EIGENSOL,NODAMPING,COORDS=1,"+str(len(freq))+",TABLE_PRECISION =3")
```

A.2 Block diagram for mechatronic co-simulations of the linear drive stabilization strategies.

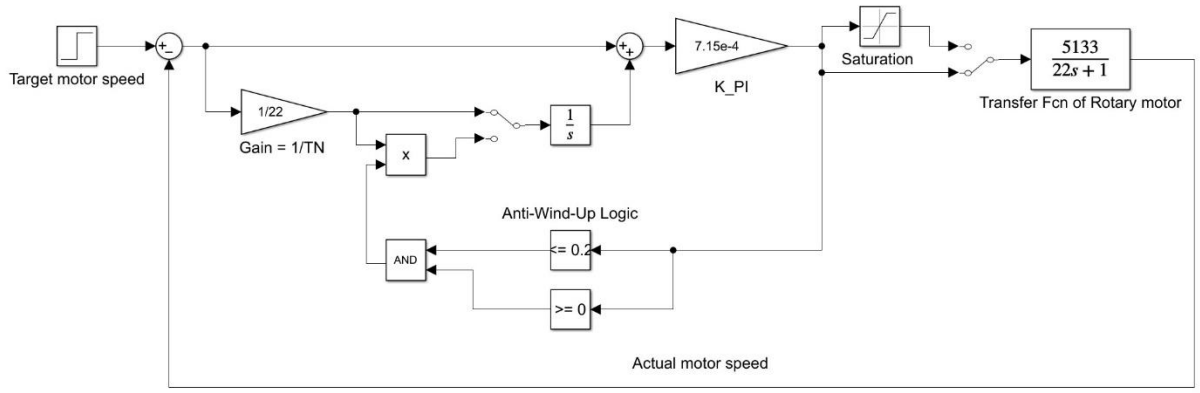


A.3 Cascade control loops of DC motors for the linear drives.

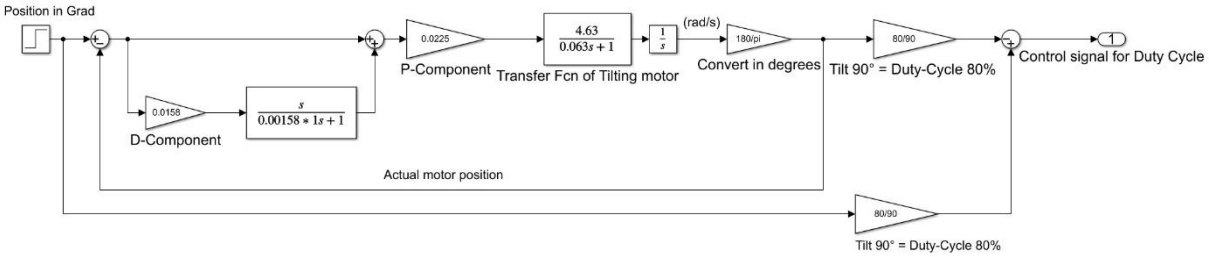


A.4 Control loops of the rotary and tilting motor for the stabilization strategy employing the gyro mechanism.

Closed-loop control of rotatory motor.



Closed-loop control of tilting motor.



A.5 Scripts implemented for the Approach B stabilization strategy.

kipp_dynamisch_bringup.launch [95]

```
<?xml version="1.0" ?>
<launch>

<node pkg="acceleration_publisher" type="acceleration_publisher"
name="acceleration_publisher" output="screen">
</node>

<node pkg="tipover_detection" type="tipover_detection" name="tipover_detection"
output="screen" respawn="true">
</node>

</launch>
```

acceleration_publisher.cpp [95]

```
#include "acceleration_publisher.h"
#include <kdl_parser/kdl_parser.hpp>//kdl_parser to create kdl_tree from URDF

//Constructor will get called whenever an instance of this class is created
// odd syntax: have to pass nodehandle pointer into constructor for constructor to build
subscribers, etc
acceleration_publisher::acceleration_publisher(ros::NodeHandle*
nodehandle):nh_(*nodehandle)
{ // constructor
ROS_INFO("constructor of class acceleration_publisher");
//initializeSubscribers(); // package up the messy work of creating subscribers; do this
overhead in constructor
initializeSubscribers();
initializePublishers();
// can also do tests/waits to make sure all required services, topics, etc are alive
}

//member helper function to set up subscribers;
//note odd syntax: &example::subscriberjointstateCallback is a pointer to a member function
of example
//"this" keyword is required, to refer to the current instance of class example

void acceleration_publisher::initializeSubscribers()
{
ROS_INFO("initializing Subscribers");
joint_state_subscriber = nh_.subscribe("joint_states", 1,
&acceleration_publisher::subscriberjointstateCallback,this);
}

//member helper function to set up publishers;
void acceleration_publisher::initializePublishers()
{
ROS_INFO("initializing Publishers");
joint_state_acceleration_publisher =
nh_.advertise<sensor_msgs::JointState>("joint_states_acceleration", 1);
}

void acceleration_publisher::subscriberjointstateCallback(const
sensor_msgs::JointState::ConstPtr& jointstate_msg) {
```

```
// this callback function wakes up every time a new message is published on "/joint_states"

for(int i=0; i<=10; i++) //iterate all joints
{
name[i] = jointstate_msg->name[i];//get joint names from topic "/joint_states""
position[i] = jointstate_msg->position[i];//get joint positions from topic "/joint_states""
velocity[i] = jointstate_msg->velocity[i];//get joint velocities from topic
"/joint_states""
}

//i=0 -> right_wheel_jointi=6 -> arm_3_joint
//i=1 -> left_wheel_joint i=7 -> arm_4_joint
//i=2 -> back_caster_jointi=8 -> arm_5_joint
//i=3 -> back_wheel_jointi=9 -> arm_6_joint
//i=4 -> arm_1_jointi=10 -> arm_7_joint
//i=5 -> arm_2_joint

//get current ROS Time
double currentTime = ros::Time::now().toSec();

//calculate time delta between last and current cycle of this callback function
double delta = currentTime - storedTime;

//calculate joint accelerations as derivative of joint velocities with f'=(y(t) - y(t-
delta_t))/delta_t (differential quotient)
for(int i=0; i<=10; i++)
{
acceleration[i] = (velocity[i]-velocity_remember[i])/delta;
}
//remember current velocity values for next loop, must be done AFTER acceleration is
calculated
for(int i=0; i<=10; i++)
{
velocity_remember[i] = velocity[i];
}
//remember current time for next loop, must be done AFTER acceleration is calculated
storedTime = currentTime;
}

int main(int argc, char** argv)
{
// ROS set-ups:
ros::init(argc, argv, "acceleration_publisher_node"); //node name
ros::NodeHandle nh; // create a node handle; need to pass this to the class constructor
ros::Rate loop_rate(10);//set a desired run time of a cycle in Hz
sensor_msgs::JointState output_message;//create output message of Type
sensor_msg::JointState

//segmentation fault (core dumped) without resize of output message
output_message.name.resize(11);
output_message.position.resize(11);
output_message.velocity.resize(11);
output_message.effort.resize(11);

ROS_INFO("main: instantiating an object of type acceleration_publisher");
acceleration_publisher acceleration_publisher(&nh);  //instantiate an
acceleration_publisher class object and pass in pointer to nodehandle for constructor to
use
while(ros::ok())
{
output_message.header.stamp = ros::Time::now();//timestamp of message
for(int i=0; i<=10; i++) //write the joint states in output message
```

```
{
output_message.name[i] = name[i];
output_message.position[i] = position[i];
output_message.velocity[i] = velocity[i];
output_message.effort[i] = acceleration[i];//publish the accelerations as "effort" in
output message of type JointState
//because there is no array for "acceleration" available in sensor_msgs::JointState
}

acceleration_publisher.joint_state_acceleration_publisher.publish(output_message);
//publish output_message on topic "/joint_states_acceleration""
::spinOnce();
loop_rate.sleep();//sleeps for any leftover time in a cycle to meet desired loop rate
(calculated from the last time sleep/reset/constructor was called)
}
return 0;
}
```

tipover_detection.cpp [95]

```
//headers in this package
#include "tipover_detection.h"

//headers for standard library
#include <iostream>
#include <fstream>

//headers for ROS
#include <ros/ros.h>
#include <ros/package.h>
#include <sensor_msgs/PointCloud.h>
#include <geometry_msgs/PointStamped.h>
#include <geometry_msgs/Point32.h>
#include <geometry_msgs/Inertia.h>
#include <geometry_msgs/Vector3.h>

//headers for urdf-parser
#include <kdl_parser/kdl_parser.hpp>

//headers for RNEA
#include <kdl/chainidsolver_recursive_newton_euler.hpp>
#include <kdl/chain.hpp>
#include <kdl/tree.hpp>
#include <kdl/segment.hpp>

//Constructor will get called whenever an instance of this class is created
//odd syntax: have to pass nodehandle pointer into constructor for constructor to build
subscribers, etc
tipover_detection::tipover_detection(ros::NodeHandle* nodehandle):nh_(*nodehandle)
{
// constructor for class tipover_detection
ROS_INFO("constructor of class tipover_detection");

initializeSubscribers();
initializePublishers();

//get robot_description parameter from parameter server
std::string robot_description_text;
nh_.param("robot_description", robot_description_text, std::string());
//alternative: nh_.getParam("/robot_description", robot_description_text);

//parse urdf by using kdl_parser
```

```
KDL::Tree robot_tree;
if(!kdl_parser::treeFromString(robot_description_text, robot_tree))
{
ROS_ERROR("failed to construct kdl tree: robot_tree");
}

//show number of tree segments and joints
ROS_INFO_STREAM("robot_tree segments: " << robot_tree.getNrOfSegments()<<" Joints:
"<<robot_tree.getNrOfJoints());

//get chain from robot_tree
if(!robot_tree.getChain("base_link","arm_ee_link",robot_chain))
//tree.getChain(chain_root,chain_tip,chain) base_link
{
ROS_ERROR("failed to construct kdl chain robot_chain from tree robot_tree");
}

//show number of chain segments and joints
ROS_INFO_STREAM("robot_chain segments: " << robot_chain.getNrOfSegments()<<"Joints:
"<<robot_chain.getNrOfJoints());

//inverse dynamics solver can only calculate forces/torques for joints
//solution: create "virtual" joints for roll/pitch/yaw (torque) and x/y/z (force) in a
separate URDF (these joints do not exist on the real robot, only needed for calculations)
//origin of these joints is where the robot arm (Schunk LWA 4D) is mounted on the base
(Scitos G5)
//create torque_tree from URDF file "torque_joints.urdf.xacro"
KDL::Tree torque_tree;
if(!kdl_parser::treeFromFile("/home/franzi/ros/ros_robo_ws_sim/src/kipp_dynamisch_ws_pgks/t
orque_joints.urdf.xacro",torque_tree))
{
ROS_ERROR("failed to construct kdl tree: torque_tree");
}
//show number of tree segments and joints
ROS_INFO_STREAM("torque_tree segments: "<<torque_tree.getNrOfSegments()<<"joints:
"<<torque_tree.getNrOfJoints());

//create chain from torque_tree
if(!torque_tree.getChain("torque_base","connector_link",torque_chain))
{
ROS_ERROR("failed to construct kdl chain torque_chain from tree torque_tree");
}
//show number of chain segments and joints
ROS_INFO_STREAM("torque_chain segments: " << torque_chain.getNrOfSegments()<<"joints:
"<<torque_chain.getNrOfJoints());

//add robot_chain at the end of torque_chain (robot arm should be "on top" of the torque
joints)
torque_chain.addChain(robot_chain);
//show number of segments and joints of combined chain
ROS_INFO_STREAM("combined chain segments: " << torque_chain.getNrOfSegments()<<"joints:
"<<torque_chain.getNrOfJoints());

//calculate mass of mobile base
std::string
base_links[6]={"base_link","scitos_right_wheel","scitos_left_wheel","scitos_wheel_back_cyli
nder","scitos_wheel_back_plate","scitos_wheel_back"};
base_mass = 0;
for(int i=0;i<6;i++)
{
base_mass=base_mass+robot_tree.getSegment(base_links[i])>second.segment.getInertia().getMas
s();
```

```
}

//resize arrays for position, velocity and acceleration to number of joints in chain (in
this case: 13)
q.resize(torque_chain.getNrOfJoints());
q_dot.resize(torque_chain.getNrOfJoints());
q_dotdot.resize(torque_chain.getNrOfJoints());

//set values to zero here in constructor as safety, values will get updated by callback
function of acceleration_subscriber
SetToZero(q);
SetToZero(q_dot);
SetToZero(q_dotdot);
}

//member function to set up subscribers
//note odd syntax: &tipover_detection::subscriberaccelerationCallback is a pointer to a
member function of tipover_detection
//"this" keyword is required, to refer to the current instance of class tipover_detection
void tipover_detection::initializeSubscribers()
{
ROS_INFO("initializing Subscribers");
acceleration_subscriber = nh_.subscribe("joint_states_acceleration",1,
&tipover_detection::subscriberaccelerationCallback,this);  //subscribes to topic
"/joint_states_acceleration""
imu_subscriber = nh_.subscribe("imu",1,&tipover_detection::subscriberimuCallback,this);
//subscribes to topic "/imu"
imu_without_gravity_subscriber =
nh_.subscribe("linear_acc_without_gravity",1,&tipover_detection::subscriberimuwithoutgravit
yCallback,this);   //subscribes to topic "/linear_acc_without_gravity"
cog_subscriber =
nh_.subscribe("cog/robot",1,&tipover_detection::subscribercogCallback,this);
}

//member function to set up publishers
void tipover_detection::initializePublishers()
{
ROS_INFO("initializing Publishers");
joint_torque_publisher = nh_.advertise<sensor_msgs::JointState>("joint_torques", 1, true);
//publishes topic "/joint_torques"
tipover_publisher = nh_.advertise<sensor_msgs::JointState>("tipover_data", 1, true);
}

void tipover_detection::subscriberaccelerationCallback(const
sensor_msgs::JointState::ConstPtr& acceleration_msg)
{
// callback function for acceleration_subscriber
// it wakes up every time a new message is published on "/joint_states_acceleration"

//position, velocity and acceleration values of torque_joint_roll, torque_joint_pitch and
torque_joint_yaw are always zero ("virtual" joints do not move)
q.data[0] = 0.0;
q_dot.data[0] = 0.0;
q_dotdot.data[0] = 0.0;
q.data[1] = 0.0;
q_dot.data[1] = 0.0;
q_dotdot.data[1] = 0.0;
q.data[2] = 0.0;
q_dot.data[2] = 0.0;
q_dotdot.data[2] = 0.0;
//position and velocity of force_joint_x, force_joint_y and force_joint_z is zero
("virtual" joints do not move)
```

```
//accelerations are set to base linear acceleration in subscriberimuwithoutgravityCallback
function to calculate forces
q.data[3] = 0.0;
q_dot.data[3] = 0.0;
//q_dotdot.data[3] = 0.0;
q.data[4] = 0.0;
q_dot.data[4] = 0.0;
//q_dotdot.data[4] = 0.0;
q.data[5] = 0.0;
q_dot.data[5] = 0.0;
//q_dotdot.data[5] = 0.0;

for (int i=6; i<=12; i++)  //robot arm joint 1..7 is "joint_states_acceleration" at
[4]..[10}
{
q.data[i] = acceleration_msg->position[i-2];      //write joint positions from topic
"/joint_states_acceleration"" to KDL::JntArray q
q_dot.data[i] = acceleration_msg->velocity[i-2];  //write joint velocities from topic
"/joint_states_acceleration"" to KDL::JntArray q_dot
q_dotdot.data[i] = acceleration_msg->effort[i-2]; //write joint accelerations from topic
"/joint_states_acceleration"" to KDL::JntArray q_dotdot
}
}

void tipover_detection::subscriberimuCallback(const sensor_msgs::Imu::ConstPtr &imu_msg)
{
//callback function for imu_subscriber
//wakes up every time a new message is published on "/imu"
//gravitational acceleration (minus any movement) in m/s²
gravity.data[0] =  10*imu_msg -> linear_acceleration.x;     //gravity vector x
gravity.data[1] =  10*imu_msg -> linear_acceleration.y;     //gravity vector y
gravity.data[2] =  10*imu_msg -> linear_acceleration.z;     //gravity vector z
}

void tipover_detection::subscriberimuwithoutgravityCallback(const
sensor_msgs::Imu::ConstPtr &linear_acc_without_gravity_msg)
{
//callback function for imu_without_gravity_subscriber
//wakes up every time a new message is published on "/linear_acc_without_gravity"
//linear acceleration data (acceleration minus gravity) in m/s²
linear_accel.data[0] = linear_acc_without_gravity_msg -> linear_acceleration.x;
linear_accel.data[1] = linear_acc_without_gravity_msg -> linear_acceleration.y;
linear_accel.data[2] = linear_acc_without_gravity_msg -> linear_acceleration.z;

//set force joint accelerations to moving base linear accelerations for force calculation
q_dotdot.data[3] = linear_acc_without_gravity_msg -> linear_acceleration.x;
q_dotdot.data[4] = linear_acc_without_gravity_msg -> linear_acceleration.y;
q_dotdot.data[5] = linear_acc_without_gravity_msg -> linear_acceleration.z;
}

void tipover_detection::subscribercogCallback(const geometry_msgs::PointStamped::ConstPtr
&cog_msg)
{
//callback function for cog_subscriber
//wakes up every time a new message is published on "/cog/robot"
cog.x = cog_msg->point.x;   //cog x in coordinate frame of "base_link"
cog.y = cog_msg->point.y;   //cog y in coordinate frame of "base_link"
cog.z = cog_msg->point.z;   //cog z in coordinate frame of "base_link"
}

geometry_msgs::Vector3 tipover_detection::crossProduct(geometry_msgs::Vector3 vector_a,
geometry_msgs::Vector3 vector_b)
```

```
{
//member function to calculate the cross product of two vectors
geometry_msgs::Vector3 vector_c;
vector_c.x = (vector_a.y * vector_b.z) - (vector_a.z * vector_b.y);
vector_c.y = (vector_a.z * vector_b.x) - (vector_a.x * vector_b.z);
vector_c.z = (vector_a.x * vector_b.y) - (vector_a.y * vector_b.x);
return vector_c;
}

double tipover_detection::dotProduct(geometry_msgs::Vector3 vector_a,
geometry_msgs::Vector3 vector_b)
{
//member function to calculate the dot product (= scalar product) of two vectors
double dotProduct = (vector_a.x * vector_b.x) + (vector_a.y * vector_b.y) + (vector_a.z *
vector_b.z);
return dotProduct;
}

geometry_msgs::Vector3 tipover_detection::addVector(geometry_msgs::Vector3 vector_a,
geometry_msgs::Vector3 vector_b)
{
//member function for addition of two vectors
geometry_msgs::Vector3 vector_c;
vector_c.x = vector_a.x + vector_b.x;
vector_c.y = vector_a.y + vector_b.y;
vector_c.z = vector_a.z + vector_b.z;
return vector_c;
}

double tipover_detection::alphacalc(double factor,double base)
{
//member function to calculate tip-over stability margin alpha
if(factor > 0)
{
return pow(base,1)*factor;
}
else
{
return pow(base,-1)*factor;
}
}

double tipover_detection::alpha_cm_calc(double factor,double base)
{
//member function to calculate tip-over stability margin alpha_cm (with incorporation of
c.m. height)
if(factor > 0)
{
return pow(base,-1)*factor;
}
else
{
return pow(base,1)*factor;
}
}

void tipover_detection::calculate_tipover(geometry_msgs::Vector3
right_wheel,geometry_msgs::Vector3 back_wheel,geometry_msgs::Vector3
left_wheel,geometry_msgs::Vector3 edge1,
geometry_msgs::Vector3 edge2,geometry_msgs::Vector3 edge3,geometry_msgs::Vector3
cog_right_wheel,geometry_msgs::Vector3 cog_back_wheel,geometry_msgs::Vector3
cog_left_wheel)
```

```
{
//member function to perform all calculations to determine tip-over stability
//calculate joint torques/forces with inverse dynamics solver (RNE)
//use the calculated forces/torques to determine (critical) tip-over margin

//build as many wrenches in vector f_ext as the number of segments in the chain (in this
case: 17 segments)
//wrenches represent external forces/torques acting on each chain element
std::vector<KDL::Wrench> f_ext; //external forces
KDL::Vector extforce(0.0,0.0,0.0); //set external forces to zero
KDL::Vector exttorque(0.0,0.0,0.0); //set external torques to zero

for(int i=0; i<=torque_chain.getNrOfSegments()-1; i++)
{
KDL::Wrench externalforce(extforce,exttorque);
//externalforce.Zero();
f_ext.push_back(externalforce);
}

rnea_return.resize(torque_chain.getNrOfJoints()); //resize array to number of joints in
chain (in this case: 13)

KDL::ChainIdSolver_RNE solver(torque_chain,gravity);  //create an element of class
ChainIdSolver_RNE and initialise it with chain and gravity vector

//q, q_dot, q_dotdot and rnea_return must be the same size as the number of joints in the
chain
//f_ext must be the same size as the number of segments in the chain
if(solver.CartToJnt(q,q_dot,q_dotdot,f_ext,rnea_return)!=0) //calculate joint torques
{
ROS_ERROR("calculation of joint torques and forces failed");       //error message if
calculation fails
}


//calculate the force acting on the cog of the mobile base (due to gravitational forces and
base motion)
//Force = (linear acceleration + gravitational acceleration) * base mass
geometry_msgs::Vector3 F_base;
F_base.x = (linear_accel[0]*(-1)+gravity[0])*base_mass;
F_base.y = (linear_accel[1]*(-1)+gravity[1])*base_mass;
F_base.z = (linear_accel[2]*(-1)+gravity[2])*base_mass;

//Wrench with ALL forces and torques exerted to the base body (in point F) due to
manipulator motion, gravitational forces, inertial force and external forces/torques
//this wrench reflects the whole effect of the manipulator arm on the mobile base
(including manipulator dynamics, end-effector loading
//and reaction forces due to interaction with the environment)
geometry_msgs::Vector3 F_r, M_r;
F_r.x = rnea_return.data[3]*(-1);  //force in direction of x-axis
F_r.y = rnea_return.data[4]*(-1);  //force in direction of y-axis
F_r.z = rnea_return.data[5]*(-1);  //force in direction of z-axis
M_r.x = rnea_return.data[0]*(-1);   //torque about x-axis (roll)
M_r.y = rnea_return.data[1]*(-1);   //torque about y-axis (pitch)
M_r.z = rnea_return.data[2]*(-1);   //torque about z-axis (yaw)

//moment of forces/torques in "F" about the corner points of the support polygon can be
calculated
// M = (r x F) + n
geometry_msgs::Vector3 M_f1, M_f2, M_f3;
M_f1 = addVector(crossProduct(right_wheel,F_r),M_r);
M_f2 = addVector(crossProduct(back_wheel,F_r),M_r);
```

```
M_f3 = addVector(crossProduct(left_wheel,F_r),M_r);

//moment about corner points of support polygon exerted by force acting on the cog of the
mobile base
geometry_msgs::Vector3 M_base1,M_base2,M_base3;
M_base1 = crossProduct(cog_right_wheel,F_base);
M_base2 = crossProduct(cog_back_wheel,F_base);
M_base3 = crossProduct(cog_left_wheel,F_base);

//calculate total moment about each corner point by adding M_f and M_base
geometry_msgs::Vector3 Mv1,Mv2,Mv3;
Mv1 = addVector(M_f1,M_base1);
Mv2 = addVector(M_f2,M_base2);
Mv3 = addVector(M_f3,M_base3);

//moments about corner points (vertices) can be projected about the different edges of the
support polygon
double M1,M2,M3;
M1 = dotProduct(Mv1,edge1);
M2 = dotProduct(Mv2,edge2);
M3 = dotProduct(Mv3,edge3);

//base moment of inertia about i-th edge of support boundary (i=1..3) in kg/m²
double I_v1 = 4.673861;
double I_v2 = 4.57659;
double I_v3 = 4.724126;

//dynamic stability margin (alpha) about each boundary edge ist computed
alpha1 = alphacalc(M1,I_v1);
alpha2 = alphacalc(M2,I_v2);
alpha3 = alphacalc(M3,I_v3);

//MHS measure is computed by considering the most critical case (smallest alpha)
//  - if the minimum of all alphas is positive (which means all alphas are positive), the
system is stable
//  - a negative alpha represents an instability about the corresponding edge (robot is
tipping over)
//  - alpha value of zero represents the critical dynamic stability
//std::min() returns the smaller of both values
alpha_critical = std::min(std::min(alpha1,alpha2),alpha3);   //alpha_critical is the
smallest of the three values

//MHS measure in the above form is not directly sensitive to the height of the center of
mass
//measurement can be improved by incorporating the c.m. height
//cog.z is expressed in coordinate frame of "base_link", needs to be converted to represent
height above ground level
double h_cm = 0.6088+(cog.z-0.452); //center of mass height (cog.z-0.452 converts from
"base_link" to coordinate frame in F, point F is 0.6088 cm above ground contact)

alpha1_cm = alpha_cm_calc(alpha1,h_cm);
alpha2_cm = alpha_cm_calc(alpha2,h_cm);
alpha3_cm = alpha_cm_calc(alpha3,h_cm);

alpha_critical_cm = std::min(std::min(alpha1_cm,alpha2_cm),alpha3_cm);
//alpha_critical_cm is the smallest of the three values
}

int main(int argc, char** argv)
{
//ROS SET-UPS:
ros::init(argc, argv, "tipover_detection_node"); //node name
```

```
ros::NodeHandle nh; // create a node handle; need to pass this to the class constructor
ros::Rate loop_rate(10);    //set a desired run time of a cycle in Hz

sensor_msgs::JointState torque_output;  //initialise an output message for joint
forces/torques
sensor_msgs::JointState tipover_output; //initialise an output message for tip-over
stability values

//resize output messages
tipover_output.name.resize(8);
tipover_output.effort.resize(8);
torque_output.name.resize(13);
torque_output.effort.resize(13);

//GEOMETRY SET-UP OF THE ROBOT
//position vectors of ground contact points in the "base coordinate frame", origin is point
A where manipulator exerts the moving base
geometry_msgs::Vector3 left_wheel, right_wheel, back_wheel;

//P1 (right wheel)
right_wheel.x = 0.075;
right_wheel.y = -0.155;
right_wheel.z = -0.6088;

//P2 (back wheel)
back_wheel.x = -0.327;
back_wheel.y = 0.0;
back_wheel.z = -0.6088;

//P3 (left wheel)
left_wheel.x = 0.075;
left_wheel.y = 0.155;
left_wheel.z = -0.6088;

//calculate the unit vectors of the edges between the ground contact points (wheels) which
form the support polygon (in this case support polygon is a triangle)
//vectors are defined so that they make a closed loop in clockwise direction
geometry_msgs::Vector3 edge1, edge2, edge3;
double norm1,norm2,norm3;
edge1.x = back_wheel.x-right_wheel.x;
edge1.y = back_wheel.y-right_wheel.y;
edge1.z = back_wheel.z-right_wheel.z;
norm1 = sqrt(pow(edge1.x,2)+pow(edge1.y,2)+pow(edge1.z,2));
edge1.x = edge1.x / norm1;
edge1.y = edge1.y / norm1;
edge1.z = edge1.z / norm1;

edge2.x = left_wheel.x-back_wheel.x;
edge2.y = left_wheel.y-back_wheel.y;
edge2.z = left_wheel.z-back_wheel.z;
norm2 = sqrt(pow(edge2.x,2)+pow(edge2.y,2)+pow(edge2.z,2));
edge2.x = edge2.x / norm2;
edge2.y = edge2.y / norm2;
edge2.z = edge2.z / norm2;

edge3.x = right_wheel.x-left_wheel.x;
edge3.y = right_wheel.y-left_wheel.y;
edge3.z = right_wheel.z-left_wheel.z;
norm3 = sqrt(pow(edge3.x,2)+pow(edge3.y,2)+pow(edge3.z,2));
edge3.x = edge3.x / norm3;
edge3.y = edge3.y / norm3;
edge3.z = edge3.z / norm3;
```

```
geometry_msgs::Vector3 cog_base,cog_right_wheel,cog_left_wheel,cog_back_wheel;
//position vector of mobile base cog, origin is point A (fixed values because mobile base
itself is rigid, therefore cog does not change)
cog_base.x = -0.00235;
cog_base.y = -0.00194;
cog_base.z = -0.39443;

//vector pointing FROM each wheel TO cog_base
cog_right_wheel.x = cog_base.x - right_wheel.x;
cog_right_wheel.y = cog_base.y - right_wheel.y;
cog_right_wheel.z = cog_base.z - right_wheel.z;

cog_back_wheel.x = cog_base.x - back_wheel.x;
cog_back_wheel.y = cog_base.y - back_wheel.y;
cog_back_wheel.z = cog_base.z - back_wheel.z;

cog_left_wheel.x = cog_base.x - left_wheel.x;
cog_left_wheel.y = cog_base.y - left_wheel.y;
cog_left_wheel.z = cog_base.z - left_wheel.z;

//invert position vectors of the ground contact points
//necessary because calculation of tip-over needs vector pointing FROM reference point of
torque (= corner points) TO the point of
//application of the force (= Point A), NOT the other way around
right_wheel.x = right_wheel.x*(-1);
right_wheel.y = right_wheel.y*(-1);
right_wheel.z = right_wheel.z*(-1);

back_wheel.x = back_wheel.x*(-1);
back_wheel.y = back_wheel.y*(-1);
back_wheel.z = back_wheel.z*(-1);

left_wheel.x = left_wheel.x*(-1);
left_wheel.y = left_wheel.y*(-1);
left_wheel.z = left_wheel.z*(-1);

ROS_INFO("main: instantiating an object of type tipover_detection");
tipover_detection tipover_detection(&nh);  //instantiate an tipover_detection class object
and pass in pointer to nodehandle for constructor to use

while(ros::ok())
{
//call member function to calculate tip-over stability
tipover_detection.calculate_tipover(right_wheel,back_wheel,left_wheel,edge1,edge2,edge3,cog
_right_wheel,cog_back_wheel,cog_left_wheel);

//write the tip-over values in output message
tipover_output.name[0] = "alpha1";
tipover_output.effort[0] = tipover_detection.alpha1;
tipover_output.name[1] = "alpha2";
tipover_output.effort[1] = tipover_detection.alpha2;
tipover_output.name[2] = "alpha3";
tipover_output.effort[2] = tipover_detection.alpha3;
tipover_output.name[3] = "alpha_critical";
tipover_output.effort[3] = tipover_detection.alpha_critical;
tipover_output.name[4] = "alpha1_cm";
tipover_output.effort[4] = tipover_detection.alpha1_cm;
tipover_output.name[5] = "alpha2_cm";
tipover_output.effort[5] = tipover_detection.alpha2_cm;
tipover_output.name[6] = "alpha3_cm";
tipover_output.effort[6] = tipover_detection.alpha3_cm;
```

```
tipover_output.name[7] = "alpha_critical_cm";
tipover_output.effort[7] = tipover_detection.alpha_critical_cm;

//write the calculated joint torques in output message
torque_output.name[0] = "torque_joint_roll";
torque_output.name[1] = "torque_joint_pitch";
torque_output.name[2] = "torque_joint_yaw";
torque_output.name[3] = "force_x";
torque_output.name[4] = "force_y";
torque_output.name[5] = "force_z";
torque_output.name[6] = "arm_1_joint";
torque_output.name[7] = "arm_2_joint";
torque_output.name[8] = "arm_3_joint";
torque_output.name[9] = "arm_4_joint";
torque_output.name[10] = "arm_5_joint";
torque_output.name[11] = "arm_6_joint";
torque_output.name[12] = "arm_7_joint";

for(int i=0;i<13;i++)
{
torque_output.effort[i] = tipover_detection.rnea_return.data[i];
}

//write timestamp of message in msg header
torque_output.header.stamp = ros::Time::now();
tipover_output.header.stamp =ros::Time::now();

//publish joint torques to topic "/joint_torques"
tipover_detection.joint_torque_publisher.publish(torque_output);
//publish tip-over values to topic "/tipover_data"
tipover_detection.tipover_publisher.publish(tipover_output);

ros::spinOnce();
loop_rate.sleep();   //sleeps for any leftover time in a cycle to meet desired loop rate
(calculated from the last time sleep/reset/constructor was called)
}
return 0;
}
```

bringup_robot_main_control_dyn.launch [175]

```
<?xml version="1.0" ?>
<launch>
<node  pkg="lowpass_filter" type="lowpass_filter" name="lowpass_filter" output="screen">
</node>
<node pkg="robot_main_control_dyn" type="robot_main_control_dyn"
name="robot_main_control_dyn" output="screen" respawn="true">
</node>
<node pkg="test_cmdvel_vali" type="test_points.py" name="help_points" output="screen"
respawn="true">
</node>
</launch>
```

lowpass_filter.cpp [175]

```
#include "ros/ros.h"
#include "geometry_msgs/Twist.h"
#include <sensor_msgs/Imu.h>

double T = 0.3; //sec
```

```
double dt = 0.1; //ms

double x_ = 0;
double y_ = 0;
double z_ = 0;
double epsilon = 0.0001;

ros::Publisher imu_filtered_pub;
sensor_msgs::Imu imu_filtered;

double lowPassFilter(double x, double y0, double dt, double T)
{
double res = y0 + (x - y0) * (dt/(dt+T));
if ((res*res) <= epsilon)
res = 0;
return res;
}

void imuCallback(const sensor_msgs::Imu::ConstPtr &imu_msg){
double x =  imu_msg -> linear_acceleration.x;
double y =  imu_msg -> linear_acceleration.y;
double z =  imu_msg -> linear_acceleration.z;

x_ = lowPassFilter(x, x_, dt, T);
y_ = lowPassFilter(y, y_, dt, T);
z_ = lowPassFilter(z, z_, dt, T);

imu_filtered.header.stamp = ros::Time::now();
imu_filtered.linear_acceleration.x = x_;
imu_filtered.linear_acceleration.y = y_;
imu_filtered.linear_acceleration.z = z_;

//imu_filtered_pub.publish(imu_filtered);
}
int main(int argc, char **argv)
{
// Publishes Odometry messages
ros::Publisher baseOdometryPublisher;

ros::init(argc, argv, "lowpass_filter");
ros::NodeHandle n;

//setup input/output communication
n.param("T", T, 0.3);
n.param("dt", dt, 0.1);
// Receives Sensor messages from Talker.
ros::Subscriber talker_sub = n.subscribe("linear_acc_without_gravity", 1000, &imuCallback);
imu_filtered_pub =n.advertise<sensor_msgs::Imu>("linear_acc_without_gravity_filter",1);
//coordination
ros::Rate rate(10);

while (n.ok()){
ros::spinOnce();
imu_filtered_pub.publish(imu_filtered);
rate.sleep();
}
return 0;
}
```

robot_main_control_dyn.cpp [175]

```
#include "robot_main_control_dyn.h"

// constructor
robot_main_control::robot_main_control(ros::NodeHandle* nodehandle):nh_(*nodehandle)
{
ROS_INFO("in class constructor of robot_main_control");
init();
run();
}

void robot_main_control::init()
{
//speed_scaling_factor controls the moving speed of the robot arm
speed_scaling_factor_arm = 1.0;
//speed_scaling_factor_base = 1.0;
//moveit_plannerID specifies the IK-solver
moveit_plannerID = "RRT"; //ESTkConfigDefault
overturn.data = false;
start.data = false;

//initialize tcp poses and current pose with zero
received_tcp_goal.pose.position.x = 0.0;
received_tcp_goal.pose.position.y = 0.0;
received_tcp_goal.pose.position.z = 0.0;
received_tcp_goal.pose.orientation.x = 0.0;
received_tcp_goal.pose.orientation.y = 0.0;
received_tcp_goal.pose.orientation.z = 0.0;
received_tcp_goal.pose.orientation.w = 0.0;

current_state.pose.position.x = 0.0;
current_state.pose.position.y = 0.0;
current_state.pose.position.z = 0.0;
current_state.pose.orientation.x = 0.0;
current_state.pose.orientation.y = 0.0;
current_state.pose.orientation.z = 0.0;
current_state.pose.orientation.w = 0.0;

using_tcp_goal.pose.position.x = 0.0;
using_tcp_goal.pose.position.y = 0.0;
using_tcp_goal.pose.position.z = 0.0;
using_tcp_goal.pose.orientation.x = 0.0;
using_tcp_goal.pose.orientation.y = 0.0;
using_tcp_goal.pose.orientation.z = 0.0;
using_tcp_goal.pose.orientation.w = 0.0;

target_arm_pose.pose.position.x = 0.0;
target_arm_pose.pose.position.y = 0.0;
target_arm_pose.pose.position.z = 0.0;
target_arm_pose.pose.orientation.x = 0.0;
target_arm_pose.pose.orientation.y = 0.0;
target_arm_pose.pose.orientation.z = 0.0;
target_arm_pose.pose.orientation.w = 0.0;

initializeSubscribers(); //creating subscribers

//RNEA
//get robot_description parameter from parameter server
std::string robot_description_text;
nh_.param("robot_description", robot_description_text, std::string());
//alternative: nh_.getParam("/robot_description", robot_description_text);
```

```
//parse urdf by using kdl_parser
KDL::Tree robot_tree;
if(!kdl_parser::treeFromString(robot_description_text, robot_tree))
{
ROS_ERROR("failed to construct kdl tree: robot_tree");
}

//show number of tree segments and joints
ROS_INFO_STREAM("robot_tree segments: " << robot_tree.getNrOfSegments()<<"Joints:"
<<robot_tree.getNrOfJoints());

//get chain from robot_tree
if(!robot_tree.getChain("base_link","arm_ee_link",robot_chain))
//tree.getChain(chain_root,chain_tip,chain) base_link
{
ROS_ERROR("failed to construct kdl chain robot_chain from tree robot_tree");
}

//show number of chain segments and joints
ROS_INFO_STREAM("robot_chain segments: " << robot_chain.getNrOfSegments()<<" Joints:"
<<robot_chain.getNrOfJoints());

//inverse dynamics solver can only calculate forces/torques for joints
//solution: create "virtual" joints for roll/pitch/yaw (torque) and x/y/z (force) in a
separate URDF (these joints do not exist on the real robot, only needed for calculations)
//origin of these joints is where the robot arm (Schunk LWA 4D) is mounted on the base
(Scitos G5)
//create torque_tree from URDF file "torque_joints.urdf.xacro"
KDL::Tree torque_tree;
if(!kdl_parser::treeFromFile("/home/franzi/ros/ros_robo_ws_sim/src/kipp_dynamisch_ws_pgks/
torque_joints.urdf.xacro",torque_tree))
{
ROS_ERROR("failed to construct kdl tree: torque_tree");
}
//show number of tree segments and joints
ROS_INFO_STREAM("torque_tree segments: "<<torque_tree.getNrOfSegments()<<" joints: "
<<torque_tree.getNrOfJoints());

//create chain from torque_tree
if(!torque_tree.getChain("torque_base","connector_link",torque_chain))
{
ROS_ERROR("failed to construct kdl chain torque_chain from tree torque_tree");
}
//show number of chain segments and joints
ROS_INFO_STREAM("torque_chain segments: " << torque_chain.getNrOfSegments()<<" joints: "
<<torque_chain.getNrOfJoints());

//add robot_chain at the end of torque_chain (robot arm should be "on top" of the torque
joints)
torque_chain.addChain(robot_chain);
//show number of segments and joints of combined chain
ROS_INFO_STREAM("combined chain segments: " << torque_chain.getNrOfSegments()<<" joints: "
<<torque_chain.getNrOfJoints());

//calculate mass of mobile base
std::string base_links[6] =
{"base_link","scitos_right_wheel","scitos_left_wheel","scitos_wheel_back_cylinder",
"scitos_wheel_back_plate","scitos_wheel_back"};
base_mass = 0;
for(int i=0;i<6;i++)
{
```

```
base_mass = base_mass + robot_tree.getSegment(base_links[i])>
second.segment.getInertia().getMass();
}

//resize arrays for position, velocity and acceleration to number of joints in chain (in
this case: 13)
q.resize(torque_chain.getNrOfJoints());
q_dot.resize(torque_chain.getNrOfJoints());
q_dotdot.resize(torque_chain.getNrOfJoints());

//set values to zero here in constructor as safety, values will get updated by callback
function of acceleration_subscriber
SetToZero(q);
SetToZero(q_dot);
SetToZero(q_dotdot);

//define alpha subscriber
alpha_crit.resize(4);
SetToZero(alpha_crit);

q_calc.resize(torque_chain.getNrOfJoints());

SetToZero(q_calc);

}
geometry_msgs::Pose robot_main_control::calculate_new_arm_position()
{

//GEOMETRY SET-UP OF THE ROBOT
//position vectors of ground contact points in the "base coordinate frame", origin is
point A where manipulator exerts the moving base
geometry_msgs::Vector3 left_wheel, right_wheel, back_wheel;

//P1 (right wheel)
right_wheel.x = 0.075;
right_wheel.y = -0.155;
right_wheel.z = -0.6088;

//P2 (back wheel)
back_wheel.x = -0.327;
back_wheel.y = 0.0;
back_wheel.z = -0.6088;

//P3 (left wheel)
left_wheel.x = 0.075;
left_wheel.y = 0.155;
left_wheel.z = -0.6088;

//calculate the unit vectors of the edges between the ground contact points (wheels) which
form the support polygon (in this case support polygon is a triangle)
//vectors are defined so that they make a closed loop in clockwise direction
geometry_msgs::Vector3 edge1, edge2, edge3;
double norm1,norm2,norm3;
edge1 = addVector(back_wheel, right_wheel,0);
norm1 = sqrt(pow(edge1.x,2)+pow(edge1.y,2)+pow(edge1.z,2));
edge1.x = edge1.x / norm1;
edge1.y = edge1.y / norm1;
edge1.z = edge1.z / norm1;

edge2 = addVector(left_wheel,back_wheel,0);
norm2 = sqrt(pow(edge2.x,2)+pow(edge2.y,2)+pow(edge2.z,2));
edge2.x = edge2.x / norm2;
```

```
edge2.y = edge2.y / norm2;
edge2.z = edge2.z / norm2;

edge3 = addVector(right_wheel,left_wheel,0);
norm3 = sqrt(pow(edge3.x,2)+pow(edge3.y,2)+pow(edge3.z,2));
edge3.x = edge3.x / norm3;
edge3.y = edge3.y / norm3;
edge3.z = edge3.z / norm3;

geometry_msgs::Vector3 cog_base,cog_right_wheel,cog_left_wheel,cog_back_wheel,cog;
//position vector of mobile base cog, origin is point A (fixed values because mobile base
itself is rigid, therefore cog does not change)
cog_base.x = -0.00235;
cog_base.y = -0.00194;
cog_base.z = -0.39443;

//vector pointing FROM each wheel TO cog_base
cog_right_wheel = addVector(cog_base,right_wheel,0);
cog_back_wheel = addVector(cog_base,back_wheel,0);
cog_left_wheel = addVector(cog_base,left_wheel,0);
//invert position vectors of the ground contact points
//necessary because calculation of tip-over needs vector pointing FROM reference point of
torque (= corner points) TO the point of
//application of the force (= Point A), NOT the other way around
right_wheel.x = right_wheel.x*(-1);
right_wheel.y = right_wheel.y*(-1);
right_wheel.z = right_wheel.z*(-1);

back_wheel.x = back_wheel.x*(-1);
back_wheel.y = back_wheel.y*(-1);
back_wheel.z = back_wheel.z*(-1);

left_wheel.x = left_wheel.x*(-1);
left_wheel.y = left_wheel.y*(-1);
left_wheel.z = left_wheel.z*(-1);

//calculates alpha, from actual situation, and if arm_link_1 and arm_link_2 are moved
about +- dq
//highest new alpha is used to calculate new point for arm.
// plus arm_1_joint = 6, arm_2_joint = 7, arm_3_joint = 8, arm_4_joint = 9 ....
q_calc = q;

double delta_q_max = 0;
KDL::Frame np_output;
KDL::Frame np_output2;
KDL::ChainFkSolverPos_recursive NewPoint(torque_chain);

std::array<int,4> needed_jointArray = {6,7,8,9};
std::array<double,2*needed_jointArray.size()+1> delta_q;

int count_itt =0;
bool error = false;
while (delta_q_max < alpha_G || error == false ) {
int a = 0;
for (int n=0;n<delta_q.size();n++) {
q_repo = q_calc;
if (n == 0){
}else {
if( n % 2 == 1){
if (a % 2 == 1){
if (q_calc.data[needed_jointArray[a]]+dq < 2.0){
q_repo.data[needed_jointArray[a]] = q_calc.data[needed_jointArray[a]]+dq;
```

```
}else {
q_repo.data[needed_jointArray[a]] = 2.0;
}
}else {
if (q_calc.data[needed_jointArray[a]]+dq < 3.0){
q_repo.data[needed_jointArray[a]] = q_calc.data[needed_jointArray[a]]+dq;
}else {
q_repo.data[needed_jointArray[a]] = 3.0;
}
}
}else {
if (a % 2 == 1){
if (q_calc.data[needed_jointArray[a]]-dq > -2.0){
q_repo.data[needed_jointArray[a]] = q_calc.data[needed_jointArray[a]]-dq;
}else {
q_repo.data[needed_jointArray[a]] = -2.0;
}
}else {
if (q_calc.data[needed_jointArray[a]]-dq > -3.0){
q_repo.data[needed_jointArray[a]] = q_calc.data[needed_jointArray[a]]-dq;
}else {
q_repo.data[needed_jointArray[a]] = -3.0;
}
a++;
}

}
}
calculate_tipover(right_wheel,back_wheel,left_wheel,edge1,edge2,edge3,cog_right_wheel,cog_
back_wheel,cog_left_wheel,q_repo);
//write calculated alpha critical in Array
delta_q[n]= this -> alpha_critical_cm;
if (int y = NewPoint.JntToCart (q_repo, np_output, torque_chain.getNrOfSegments())<0){
ROS_ERROR("failed to calcutale new point");
}
delta_h[n] = np_output.p.z()+0.452;
}
//get Maximum of calculated alpha critical
int x= 0;
double delta_h_max = delta_h[0];
delta_q_max = delta_q[0];
for (int n=0;n<delta_q.size();n++) {
if (delta_q[n] > delta_q_max){ // && delta_h[i]>delta_h_max){
x = n;
delta_q_max = delta_q[n];
delta_h_max = delta_h[n];
}
}
q_repo = q_calc;

if (x == 0){
error = true;
ROS_WARN_STREAM("No convergence or better value. Use last calculated value");
}else {
// set new konfiguration.
// target_joint is the joint which has to be changed. Get Position of Targetjoint in Array
needed_jointArray
int target_joint = (int) (x+1)/2 -1;
// Check if with Modulus if +dp or -dp
if (x%2 ==0) {
q_repo.data[needed_jointArray[target_joint]]=
q_calc.data[needed_jointArray[target_joint]]-dq;
```

```
}else {
q_repo.data[needed_jointArray[target_joint]]=
q_calc.data[needed_jointArray[target_joint]]+dq;
}

// set reposition konfiguration for new konfiguraion
q_calc = q_repo;
// iteration counter
}
count_itt++;
}

if (int y = NewPoint.JntToCart (q, np_output2, torque_chain.getNrOfSegments())<0)
{
ROS_ERROR("failed to calcutale new point");
}

if (int y = NewPoint.JntToCart (q_repo, np_output, torque_chain.getNrOfSegments())<0)
{
ROS_ERROR("failed to calcutale new point");
}
//geometry_msgs::Pose repo_state;
repo_state.pose.position.x = np_output.p.x();
repo_state.pose.position.y = np_output.p.y();
repo_state.pose.position.z = np_output.p.z()+0.452;
repo_state.pose.orientation.x = PoseEE.pose.orientation.x;
repo_state.pose.orientation.y = PoseEE.pose.orientation.y;
repo_state.pose.orientation.z = PoseEE.pose.orientation.z;
repo_state.pose.orientation.w = PoseEE.pose.orientation.w;

return  repo_state.pose;
}

void robot_main_control::run()
{
//preconfigure part for scitos and robotarm
//tell the action client that we want to spin a thread by default
MoveBaseClient ac("move_base", true);

//wait for the action server to come up
while(!ac.waitForServer(ros::Duration(5.0))){
ROS_INFO("Waiting for the move_base action server to come up");
}
move_base_msgs::MoveBaseGoal scitos_goal;

scitos_goal.target_pose.header.frame_id = "map";
scitos_goal.target_pose.header.stamp = ros::Time::now();
// set static variable for Planning_group
static const std::string PLANNING_GROUP = "arm";
moveit::planning_interface::MoveGroupInterface arm_plan_group(PLANNING_GROUP);

// We are going to use the :planning_scene_interface:`PlanningSceneInterface`
// class to add and remove collision objects in our "virtual world" scene
moveit::planning_interface::PlanningSceneInterface planning_scene_interface;

arm_plan_group.setMaxAccelerationScalingFactor(speed_scaling_factor_arm);
arm_plan_group.setMaxVelocityScalingFactor(speed_scaling_factor_arm);

//configure moveit Planner
arm_plan_group.setPlannerId(moveit_plannerID);
arm_plan_group.setPlanningTime(10);
arm_plan_group.setPoseReferenceFrame("base_footprint");// base_footprint
```

```
arm_plan_group.setStartStateToCurrentState();

ROS_INFO("Planning reference frame: %s", arm_plan_group.getPlanningFrame().c_str());
ROS_INFO("End effector reference frame: %s", arm_plan_group.getEndEffectorLink().c_str());

int reduce_counter=0;
ros::AsyncSpinner spinner(1);
spinner.start();
const moveit::core::JointModelGroup* joint_model_group =
arm_plan_group.getCurrentState()->getJointModelGroup(PLANNING_GROUP);

// Define a collision object ROS message.
moveit_msgs::CollisionObject collision_object;
collision_object.header.frame_id = arm_plan_group.getPlanningFrame();

// The id of the object is used to identify it.
collision_object.id = "box1";

// Define a box to add to the world.
shape_msgs::SolidPrimitive primitive;
primitive.type = primitive.BOX;
primitive.dimensions.resize(3);
primitive.dimensions[0] = 0.2;
primitive.dimensions[1] = 1.0;
primitive.dimensions[2] = 2;

// Define a pose for the box (specified relative to frame_id)
geometry_msgs::Pose box_pose;
box_pose.orientation.w = 1.0;
box_pose.position.x = X_KRIT+primitive.dimensions[0]*0.5;
box_pose.position.y = 0;
box_pose.position.z = primitive.dimensions[2]*0.5;

collision_object.primitives.push_back(primitive);
collision_object.primitive_poses.push_back(box_pose);
collision_object.operation = collision_object.ADD;

std::vector<moveit_msgs::CollisionObject> collision_objects;
collision_objects.push_back(collision_object);

ROS_INFO("Add the box in front of the Robot into the world");
planning_scene_interface.addCollisionObjects(collision_objects);

//main
while(ros::ok())
{
try{

listen_pose_of_scitos.waitForTransform("/map","/base_footprint",ros::Time(0),
ros::Duration(0.2));

listen_pose_of_scitos.lookupTransform("/map","/base_footprint",ros::Time(0),
transform_of_scitos);

listen_pose_of_ee.waitForTransform("/base_footprint","/arm_ee_link",ros::Time(0),
ros::Duration(0.2));

listen_pose_of_ee.lookupTransform("/base_footprint","/arm_ee_link",ros::Time(0),
transform_of_ee);
}
catch (tf::TransformException ex){
ROS_ERROR("%s",ex.what());
```

```
ros::Duration(1.0).sleep();
}

// Get current pose of scitos via tf /map /base_footprint
current_state.pose.position.x = transform_of_scitos.getOrigin().x();
current_state.pose.position.y = transform_of_scitos.getOrigin().y();
current_state.pose.position.z = transform_of_scitos.getOrigin().z();

current_state.pose.orientation.x = transform_of_scitos.getRotation().x();
current_state.pose.orientation.y = transform_of_scitos.getRotation().y();
current_state.pose.orientation.z = transform_of_scitos.getRotation().z();
current_state.pose.orientation.w = transform_of_scitos.getRotation().w();

ee_pose.pose.orientation.x = transform_of_ee.getRotation().x();
ee_pose.pose.orientation.y = transform_of_ee.getRotation().y();
ee_pose.pose.orientation.z = transform_of_ee.getRotation().z();
ee_pose.pose.orientation.w = transform_of_ee.getRotation().w();

ROS_INFO("Please publish a target pose to the topic: /estimated_tcp_goal and then start the
execution with publishing at the topic /start_moving_robot.");

while(start.data == false && ros::ok()){
} //waiting for publishing start true;
move_arm.data = false;
goal_pos.data = false;
repo.data = false;

using_tcp_goal = received_tcp_goal;
// prepare Point for coordinate Transformation from /map to /base_footprint
TCP_global.header.stamp = ros::Time(0);
TCP_global.header.frame_id ="map";
TCP_global.point.x = received_tcp_goal.pose.position.x;
TCP_global.point.y = received_tcp_goal.pose.position.y;
TCP_global.point.z = received_tcp_goal.pose.position.z;

// Transform the Point into Base frame
listen_pose_of_scitos.transformPoint("/base_footprint", TCP_global,TCP_base);

//call function
calculate_scitos_arm_target_poses();

repo.data = false;
//check if estimated TCP is within reachable workingspace
if (in_workspace.data == false){
if (alpha_crit.data[3]<= alpha_G)
{
ROS_WARN("Critical state. Initializing repositioning of arm.");
while (alpha_crit.data[3]<= alpha_G) {

// get initial Orientation
PoseEE = arm_plan_group.getCurrentPose("arm_ee_link");

arm_plan_group.setPoseTarget(calculate_new_arm_position());
moveit::planning_interface::MoveGroupInterface::Plan my_plan;

moveit_msgs::MoveItErrorCodes success = arm_plan_group.plan(my_plan);
if(success.SUCCESS == 1) {

ROS_INFO("Sending robot arm new Position");
arm_plan_group.execute(my_plan);
arm_plan_group.clearPathConstraints();
```

```
geometry_msgs::Vector3 ee_position;
ee_position.x = transform_of_ee.getOrigin().x();
ee_position.y = transform_of_ee.getOrigin().y();
ee_position.z = transform_of_ee.getOrigin().z();

geometry_msgs::Vector3 repo_position;
repo_position.x =  repo_state.pose.position.x;
repo_position.y =  repo_state.pose.position.y;
repo_position.z =  repo_state.pose.position.z;
int wait = 0;
//Wait until robot arm has finished moving
while (dTwoPoints(ee_position,repo_position)>0.05) {
listen_pose_of_ee.lookupTransform("/base_footprint","/arm_ee_link",ros::Time(0),
transform_of_ee);

ee_position.x = transform_of_ee.getOrigin().x();
ee_position.y = transform_of_ee.getOrigin().y();
ee_position.z = transform_of_ee.getOrigin().z();

sleep(1);
wait ++;
if (wait == 5){break;}
}
}
}
repo.data = true;

}else {
repo.data = true;
}
}else {
move_arm.data = true;
}
if (repo.data){
//start planing and executing robot scitos
scitos_goal.target_pose.header.frame_id = "map";
scitos_goal.target_pose.header.stamp = ros::Time::now();
scitos_goal.target_pose.pose = scitos_pose.pose;

ac.sendGoal(scitos_goal);
ROS_INFO("Sending scitos goal: (%f,%f,%f)",scitos_goal.target_pose.pose.position.x,
scitos_goal.target_pose.pose.position.y,scitos_goal.target_pose.pose.position.z);
ROS_INFO("Sending scitos goal");
repo.data = false;

// Calculate Distance between Goal an actual position
geometry_msgs::Vector3 scitos_position;
scitos_position.x = transform_of_scitos.getOrigin().x();
scitos_position.y = transform_of_scitos.getOrigin().y();
scitos_position.z = transform_of_scitos.getOrigin().z();

geometry_msgs::Vector3 scitos_goal_position;
scitos_goal_position.x = scitos_goal.target_pose.pose.position.x;
scitos_goal_position.y = scitos_goal.target_pose.pose.position.y;
scitos_goal_position.z = scitos_goal.target_pose.pose.position.z;

geometry_msgs::Vector3Stamped ex_global;
ex_global.header.stamp = ros::Time(0);
ex_global.header.frame_id ="map";
ex_global.vector.x = 1;
ex_global.vector.y = 0;
ex_global.vector.z = 0;
```

```
geometry_msgs::Vector3Stamped ex_base;
double angle_scitos_goal= 180;
ROS_INFO_STREAM("distance = "<< dTwoPoints(scitos_position,scitos_goal_position) <<
 "  Angle: "<< angle_scitos_goal);
// Check if Scitos is on goal pose.

while (dTwoPoints(scitos_position,scitos_goal_position)>0.1 || angle_scitos_goal >10 ) {
listen_pose_of_scitos.lookupTransform("/map","/base_footprint",ros::Time(0),
transform_of_scitos);
scitos_position.x = transform_of_scitos.getOrigin().x();
scitos_position.y = transform_of_scitos.getOrigin().y();
scitos_position.z = transform_of_scitos.getOrigin().z();

// calculate angle between goal orientation and current orientation from scitos
listen_pose_of_scitos.transformVector("/base_footprint", ex_global,ex_base);
// Scalar produkt from ex_base with (1|0) x-direction from scitos

angle_scitos_goal = acos((1*ex_base.vector.x + 0*ex_base.vector.y))*180/PI;
if (alpha_crit.data[3] <= alpha_G)
{
ROS_INFO("Thats too critical i have find an other position for my arm");
PoseEE = arm_plan_group.getCurrentPose("arm_ee_link");
ROS_ERROR_STREAM("Orientation is, x: "<< PoseEE.pose.orientation.x << " y:"
<< PoseEE.pose.orientation.y << " z: " << PoseEE.pose.orientation.z << " w: "
<< PoseEE.pose.orientation.w);

arm_plan_group.setPoseTarget(calculate_new_arm_position());

moveit::planning_interface::MoveGroupInterface::Plan my_plan;

moveit_msgs::MoveItErrorCodes success = arm_plan_group.plan(my_plan);
if(success.SUCCESS == 1) {
arm_plan_group.execute(my_plan); //wait until planning is ready
ROS_INFO("Sending robot arm new Position");
}
}
}
move_arm.data = true;
goal_pos.data = true;
}
if (move_arm.data){
//start Planning and execute the robot arm
arm_plan_group.setGoalTolerance(0.1);
arm_plan_group.setPoseTarget(target_arm_pose.pose);
moveit::planning_interface::MoveGroupInterface::Plan my_plan;
moveit_msgs::MoveItErrorCodes success = arm_plan_group.plan(my_plan);

if(success.SUCCESS == 1) {
arm_plan_group.execute(my_plan); //wait until planning is ready
ROS_INFO("Sending robot arm goal");
}
goal_pos.data = true;
}
int waiting = 0;
while(waiting <= 5 && goal_pos.data)
{
waiting++;
sleep(1);
}

OS_INFO("Ready for next target! Please publish first on tcp_goal and then on start.");
tart.data = false;
```

```
}
spinner.stop();
}
double robot_main_control::dTwoPoints(geometry_msgs::Vector3 p1, geometry_msgs::Vector3
p2){
double distance;

distance = sqrt(pow(p1.x-p2.x,2)+pow(p1.y-p2.y,2)+pow(p1.z-p2.z,2));

return distance;
}
void robot_main_control::initializeSubscribers()
{
ROS_INFO("Initializing Subscribers");
tcp_goal_subscriber = nh_.subscribe("estimated_tcp_goal", 1,
&robot_main_control::subscribergoalCallback,this);
overtrun_subscriber = nh_.subscribe("kippgefahr", 1,
&robot_main_control::subscriberoverturnCallback,this);
start_moving_subscriber = nh_.subscribe("/start_moving_robot", 1,
&robot_main_control::subscriber_start_moving_Callback,this);
tipover_value_subscriber = nh_.subscribe("/tipover_data", 1,
&robot_main_control::subscriber_tipover_Callback,this);

acceleration_subscriber = nh_.subscribe("joint_states_acceleration", 1,
&robot_main_control::subscriberaccelerationCallback,this);  //subscribes to topic
"/joint_states_acceleration""
imu_subscriber = nh_.subscribe("imu",1,&robot_main_control::subscriberimuCallback,this);
//subscribes to topic "/imu"

//Check Parameter "use_lowpassfilter" and initialize Subcriber
bool lowpassfilter;
ros::param::get("/use_lowpass",lowpassfilter);
if (lowpassfilter)
{
imu_without_gravity_subscriber =
nh_.subscribe("linear_acc_without_gravity_filter",1,&robot_main_control::subscriberimuwitho
utgravityCallback,this);   //subscribes to topic "/linear_acc_without_gravity"
ROS_INFO("Parameter for Lowpassfilter is set: %d, I'm using filtered data ",lowpassfilter);
}
else
{
imu_without_gravity_subscriber =
nh_.subscribe("linear_acc_without_gravity",1,&robot_main_control::subscriberimuwithoutgravi
tyCallback,this);   //subscribes to topic "/linear_acc_without_gravity"
ROS_INFO("Parameter for Lowpassfilter is set: %d, I'm using raw data ",lowpassfilter);
}
cog_subscriber =
nh_.subscribe("cog/robot",1,&robot_main_control::subscribercogCallback,this);
}

void robot_main_control::subscribergoalCallback(const geometry_msgs::Pose::ConstPtr
&pose_goal_msg)
{
received_tcp_goal.pose.position = pose_goal_msg->position;
received_tcp_goal.pose.orientation = pose_goal_msg->orientation;
}

void robot_main_control::subscriberoverturnCallback(const std_msgs::Bool::ConstPtr
&overturn_msg)
{
 overturn.data = overturn_msg->data;
}
```

```
void robot_main_control::subscriber_tipover_Callback (const
sensor_msgs::JointState::ConstPtr &alpha_krit)
{
// it wakes up every time a new message is published on "/tipover_data"
alpha_crit.data[0] = alpha_krit ->effort[4];
alpha_crit.data[1] = alpha_krit ->effort[5];
alpha_crit.data[2] = alpha_krit ->effort[6];
alpha_crit.data[3] = alpha_krit ->effort[7];
}

void robot_main_control::subscriberaccelerationCallback(const
sensor_msgs::JointState::ConstPtr& acceleration_msg)
{
// callback function for acceleration_subscriber
// it wakes up every time a new message is published on "/joint_states_acceleration"

//position, velocity and acceleration values of torque_joint_roll, torque_joint_pitch and
torque_joint_yaw are always zero ("virtual" joints do not move)
q.data[0] = 0.0;
q_dot.data[0] = 0.0;
q_dotdot.data[0] = 0.0;
q.data[1] = 0.0;
q_dot.data[1] = 0.0;
q_dotdot.data[1] = 0.0;
q.data[2] = 0.0;
q_dot.data[2] = 0.0;
q_dotdot.data[2] = 0.0;
//position and velocity of force_joint_x, force_joint_y and force_joint_z is zero
("virtual" joints do not move)
//accelerations are set to base linear acceleration in subscriberimuwithoutgravityCallback
function to calculate forces
q.data[3] = 0.0;
q_dot.data[3] = 0.0;
q.data[4] = 0.0;
q_dot.data[4] = 0.0;
q.data[5] = 0.0;
q_dot.data[5] = 0.0;

for (int i=6; i<=12; i++)  //robot arm joint 1..7 is "joint_states_acceleration" at
[4]..[10}
{
q.data[i] = acceleration_msg->position[i-2];      //write joint positions from topic
"/joint_states_acceleration"" to KDL::JntArray q
}
}

void robot_main_control::subscriberimuCallback(const sensor_msgs::Imu::ConstPtr &imu_msg)
{
//callback function for imu_subscriber
//wakes up every time a new message is published on "/imu"
//gravitational acceleration (minus any movement) in m/s²
//Change to 1*imu_msg 21.10.2020 /for simulation
gravity.data[0] =  10*imu_msg -> linear_acceleration.x;     //gravity vector x
gravity.data[1] =  10*imu_msg -> linear_acceleration.y;     //gravity vector y
gravity.data[2] =  10*imu_msg -> linear_acceleration.z;     //gravity vector z
}

void robot_main_control::subscriberimuwithoutgravityCallback(const
sensor_msgs::Imu::ConstPtr &linear_acc_without_gravity_msg)
{
//callback function for imu_without_gravity_subscriber
```

```
//wakes up every time a new message is published on "/linear_acc_without_gravity"
//linear acceleration data (acceleration minus gravity) in m/s²
linear_accel.data[0] = linear_acc_without_gravity_msg -> linear_acceleration.x;
linear_accel.data[1] = linear_acc_without_gravity_msg -> linear_acceleration.y;
linear_accel.data[2] = linear_acc_without_gravity_msg -> linear_acceleration.z;

//set force joint accelerations to moving base linear accelerations for force calculation
q_dotdot.data[3] = linear_acc_without_gravity_msg -> linear_acceleration.x;
q_dotdot.data[4] = linear_acc_without_gravity_msg -> linear_acceleration.y;
q_dotdot.data[5] = linear_acc_without_gravity_msg -> linear_acceleration.z;
}

void robot_main_control::subscribercogCallback(const geometry_msgs::PointStamped::ConstPtr
&cog_msg)
{
//callback function for cog_subscriber
//wakes up every time a new message is published on "/cog/robot"
cog.x = cog_msg->point.x;   //cog x in coordinate frame of "base_link"
cog.y = cog_msg->point.y;   //cog y in coordinate frame of "base_link"
cog.z = cog_msg->point.z;   //cog z in coordinate frame of "base_link"
}

void robot_main_control::subscriber_start_moving_Callback(const std_msgs::Bool::ConstPtr
&start_msg)
{
start.data = start_msg->data;
}

geometry_msgs::Vector3 robot_main_control::crossProduct(geometry_msgs::Vector3 vector_a,
geometry_msgs::Vector3 vector_b)
{
//member function to calculate the cross product of two vectors
geometry_msgs::Vector3 vector_c;
vector_c.x = (vector_a.y * vector_b.z) - (vector_a.z * vector_b.y);
vector_c.y = (vector_a.z * vector_b.x) - (vector_a.x * vector_b.z);
vector_c.z = (vector_a.x * vector_b.y) - (vector_a.y * vector_b.x);
return vector_c;
}

double robot_main_control::dotProduct(geometry_msgs::Vector3 vector_a,
geometry_msgs::Vector3 vector_b)
{
//member function to calculate the dot product (= scalar product) of two vectors
double dotProduct = (vector_a.x * vector_b.x) + (vector_a.y * vector_b.y) + (vector_a.z *
vector_b.z);
return dotProduct;
}

geometry_msgs::Vector3 robot_main_control::addVector(geometry_msgs::Vector3 vector_a,
geometry_msgs::Vector3 vector_b, int i)
{
//member function for addition of two vectors
//integer decides if plus or minus
geometry_msgs::Vector3 vector_c;
if (i==1)
{
vector_c.x = vector_a.x + vector_b.x;
vector_c.y = vector_a.y + vector_b.y;
vector_c.z = vector_a.z + vector_b.z;
}else {
vector_c.x = vector_a.x - vector_b.x;
vector_c.y = vector_a.y - vector_b.y;
```

```
vector_c.z = vector_a.z - vector_b.z;
}
return vector_c;
}

double robot_main_control::alphacalc(double factor,double base)
{
//member function to calculate tip-over stability margin alpha
if(factor > 0)
{
return pow(base,1)*factor;
}
else
{
return pow(base,1)*factor;
}
}

double robot_main_control::alpha_cm_calc(double factor,double base)
{
//member function to calculate tip-over stability margin alpha_cm (with incorporation of
c.m. height)
if(factor > 0)
{
return pow(base,-1)*factor;
}
else
{
return pow(base,-1)*factor;
}
}

void robot_main_control::calculate_tipover(geometry_msgs::Vector3
right_wheel,geometry_msgs::Vector3 back_wheel,geometry_msgs::Vector3
left_wheel,geometry_msgs::Vector3 edge1,
geometry_msgs::Vector3 edge2,geometry_msgs::Vector3 edge3,geometry_msgs::Vector3
cog_right_wheel,geometry_msgs::Vector3 cog_back_wheel,geometry_msgs::Vector3
cog_left_wheel, KDL::JntArray q)
{
//member function to perform all calculations to determine tip-over stability
//calculate joint torques/forces with inverse dynamics solver (RNE)
//use the calculated forces/torques to determine (critical) tip-over margin

//build as many wrenches in vector f_ext as the number of segments in the chain (in this
case: 17 segments)
//wrenches represent external forces/torques acting on each chain element
std::vector<KDL::Wrench> f_ext; //external forces
KDL::Vector extforce(0.0,0.0,0.0); //set external forces to zero
KDL::Vector exttorque(0.0,0.0,0.0); //set external torques to zero

for(int i=0; i<=torque_chain.getNrOfSegments()-1; i++)
{
KDL::Wrench externalforce(extforce,exttorque);
f_ext.push_back(externalforce);
}

rnea_return.resize(torque_chain.getNrOfJoints()); //resize array to number of joints in
chain (in this case: 13)

KDL::ChainIdSolver_RNE solver(torque_chain,gravity);  //create an element of class
ChainIdSolver_RNE and initialise it with chain and gravity vector
```

```
//q, q_dot, q_dotdot and rnea_return must be the same size as the number of joints in the
chain
//f_ext must be the same size as the number of segments in the chain
if(solver.CartToJnt(q,q_dot,q_dotdot,f_ext,rnea_return)!=0) //calculate joint torques
{
ROS_ERROR("calculation of joint torques and forces failed");       //error message if
calculation fails
}

//calculate the force acting on the cog of the mobile base (due to gravitational forces and
base motion)
//Force = (linear acceleration + gravitational acceleration) * base mass
geometry_msgs::Vector3 F_base;

F_base.x =(linear_accel[0]*(-1)+gravity[0])*base_mass;
F_base.y =(linear_accel[1]*(-1)+gravity[1])*base_mass;
F_base.z =(linear_accel[2]*(-1)+gravity[2])*base_mass;

//Wrench with ALL forces and torques exerted to the base body (in point F) due to
manipulator motion, gravitational forces, inertial force and external forces/torques
//this wrench reflects the whole effect of the manipulator arm on the mobile base
(including manipulator dynamics, end-effector loading
//and reaction forces due to interaction with the environment)
geometry_msgs::Vector3 F_r, M_r;
F_r.x = rnea_return.data[3]*(-1);  //force in direction of x-axis
F_r.y = rnea_return.data[4]*(-1);  //force in direction of y-axis
F_r.z = rnea_return.data[5]*(-1);  //force in direction of z-axis
M_r.x = rnea_return.data[0]*(-1);   //torque about x-axis (roll)
M_r.y = rnea_return.data[1]*(-1);   //torque about y-axis (pitch)
M_r.z = rnea_return.data[2]*(-1);   //torque about z-axis (yaw)

//moment of forces/torques in "F" about the corner points of the support polygon can be
calculated
// M = (r x F) + n
geometry_msgs::Vector3 M_f1, M_f2, M_f3;
M_f1 = addVector(crossProduct(right_wheel,F_r),M_r,1);
M_f2 = addVector(crossProduct(back_wheel,F_r),M_r,1);
M_f3 = addVector(crossProduct(left_wheel,F_r),M_r,1);

//moment about corner points of support polygon exerted by force acting on the cog of the
mobile base
geometry_msgs::Vector3 M_base1,M_base2,M_base3;
M_base1 = crossProduct(cog_right_wheel,F_base);
M_base2 = crossProduct(cog_back_wheel,F_base);
M_base3 = crossProduct(cog_left_wheel,F_base);

//calculate total moment about each corner point by adding M_f and M_base
geometry_msgs::Vector3 Mv1,Mv2,Mv3;
Mv1 = addVector(M_f1,M_base1,1);
Mv2 = addVector(M_f2,M_base2,1);
Mv3 = addVector(M_f3,M_base3,1);

//moments about corner points (vertices) can be projected about the different edges of the
support polygon
double M1,M2,M3;
M1 = dotProduct(Mv1,edge1);
M2 = dotProduct(Mv2,edge2);
M3 = dotProduct(Mv3,edge3);

//base moment of inertia about i-th edge of support boundary (i=1..3) in kg/m²
double I_v1 = 4.673861;
double I_v2 = 4.57659;
```

```
double I_v3 = 4.724126;

//dynamic stability margin (alpha) about each boundary edge ist computed
alpha1 = alphacalc(M1,I_v1);
alpha2 = alphacalc(M2,I_v2);
alpha3 = alphacalc(M3,I_v3);

//MHS measure is computed by considering the most critical case (smallest alpha)
//  - if the minimum of all alphas is positive (which means all alphas are positive), the
system is stable
//  - a negative alpha represents an instability about the corresponding edge (robot is
tipping over)
//  - alpha value of zero represents the critical dynamic stability
alpha_critical = std::min(std::min(alpha1,alpha2),alpha3);   //alpha_critical is the
smallest of the three values

//MHS measure in the above form is not directly sensitive to the height of the center of
mass
//measurement can be improved by incorporating the c.m. height
//cog.z is expressed in coordinate frame of "base_link", needs to be converted to represent
height above ground level
double h_cm = 0.6088+(cog.z-0.452); //center of mass height (cog.z-0.452 converts from
"base_link" to coordinate frame in F, point F is 0.6088 cm above ground contact)

alpha1_cm = alpha_cm_calc(alpha1,h_cm);
alpha2_cm = alpha_cm_calc(alpha2,h_cm);
alpha3_cm = alpha_cm_calc(alpha3,h_cm);

alpha_critical_cm = std::min(std::min(alpha1_cm,alpha2_cm),alpha3_cm);
//alpha_critical_cm is the smallest of the three values
}

void robot_main_control::check_workspace()
{
//This feature decide if the estimated Point is in the Workpspace.
double angle_scitos_TCP;
double dot_product_scitos_TCP;

in_workspace.data = false;
// get current pose of Scitos
double current_position_x = transform_of_scitos.getOrigin().x();
double current_position_y = transform_of_scitos.getOrigin().y();
double current_position_z = 0.832; // Position of Arm_2_link

// Calculation of Angle between Pose Scitos and TCP to Check if it is before or behind the
robot
double a1 = 1; //coodinates of x-axsis in BaseKoordinateframe
double a2 = 0; //coodinates of x-axsis in BaseKoordinateframe

double b1 = TCP_base.point.x; //transformed Point from /map to /base_footprint
double b2 = TCP_base.point.y;

double amount_b = sqrt(pow(b1,2)+pow(b2,2));
double amount_a = sqrt(pow(a1,2)+pow(a2,2));

dot_product_scitos_TCP = a1*b1 +a2*b2;

angle_scitos_TCP = acos(dot_product_scitos_TCP/(amount_b*amount_a))*180/PI;

if(angle_scitos_TCP<90){
// if angle_scitos_TCP smaller than 90 degree use ellipsoide, TCP is in front of robot
```

```
double scitos_elipsoide =
pow((TCP_base.point.x)/X_KRIT,2)+pow((TCP_base.point.y)/MAX_WS_RADIUS,2)+pow((TCP_global.po
int.z-current_position_z)/MAX_WS_RADIUS,2);
if(scitos_elipsoide < 1){
in_workspace.data = true;
ROS_INFO("It is in front of me");
}
}
else{
// if angle_scitos_TCP higher than 90 degree use spehre, TCP is behind robot
double scitos_sphere =
pow(TCP_base.point.x,2)+pow(TCP_base.point.y,2)+pow(TCP_global.point.z-
current_position_z,2);
if(scitos_sphere < pow(MAX_WS_RADIUS,2)){
in_workspace.data = true;
ROS_INFO("It is behind me");
}
}
}
void robot_main_control::calculate_scitos_arm_target_poses()
{
//check if the given TCP is in the workingspace
check_workspace();

if(in_workspace.data == false){
//Goal not able to reach only with move_arm. First need to move the base.
//calculate the goal for the base and the arm and set the calulated goal to scitos_pose and
arm_pose

scitos_pose.pose.position.x = TCP_global.point.x -MAX_VALUE_DELTA_X;
scitos_pose.pose.position.y = TCP_global.point.y -MAX_VALUE_DELTA_Y;

scitos_pose.pose.orientation.w =  1;
scitos_pose.pose.orientation.z =  0;

target_arm_pose.pose.position.x = MAX_VALUE_DELTA_X;
target_arm_pose.pose.position.y = MAX_VALUE_DELTA_Y;
target_arm_pose.pose.position.z = TCP_global.point.z;
target_arm_pose.pose.orientation = using_tcp_goal.pose.orientation;


ROS_INFO("movement of base required");
ROS_INFO("Scitos_goal_pose: [%f], [%f], [%f]", scitos_pose.pose.position.x,
scitos_pose.pose.position.y, scitos_pose.pose.orientation.w);
ROS_INFO("arm_goal_pose: [%f], [%f], [%f]", target_arm_pose.pose.position.x,
target_arm_pose.pose.position.y, target_arm_pose.pose.position.z);

}
else{
// estimated TCP is within the workspace
target_arm_pose.pose.position.x = TCP_base.point.x;   //using relative x position of TCP
target_arm_pose.pose.position.y = TCP_base.point.y;   //using relative y position of TCP
target_arm_pose.pose.position.z = TCP_global.point.z; //using real high of TCP
target_arm_pose.pose.orientation = using_tcp_goal.pose.orientation;

ROS_INFO("arm_goal_pose: [%f], [%f], [%f]", target_arm_pose.pose.position.x,
target_arm_pose.pose.position.y, target_arm_pose.pose.position.z);
}
ROS_INFO("calculate_scitos_arm_target_poses is done");
}

int main(int argc, char** argv)
```

```
{
// ROS set-ups:
ros::init(argc, argv, "robot_main_control_dyn"); //node name

ros::NodeHandle nh;

ROS_INFO("main: instantiating an object of type robot_main_control");
robot_main_control robot_main_control(&nh);

return 0;
}
```

test_points.py [175]

```
import rospy
import roslib
import numpy as np
from geometry_msgs.msg import Pose
from geometry_msgs.msg import Twist
from std_msgs.msg import Bool
from time import sleep
import tf

estimated_goal = Pose()
offset = Pose()
tf_pose = Pose()

def callback_goal(goal):
#Angle between /map and /base_link
phi = np.arccos(tf_pose.orientation.w)*2
a= tf_pose.orientation.w
b= tf_pose.orientation.x
c= tf_pose.orientation.y
d = tf_pose.orientation.z
xbase1 = 2*(a**2+b**2)-1
xbase2 = 2*(b*c+a*d)
ybase1 =2*(b*c-a*d)
ybase2 =2*(a**2+c**2)-1

amount_x= np.sqrt(xbase1**2+xbase2**2)
amount_y= np.sqrt(ybase1**2+ybase2**2)
xbase1 = xbase1/amount_x
xbase2 = xbase2/amount_x
ybase1 =ybase1/amount_y
ybase2 =ybase2/amount_y

#Relative positions of offset, depending on
estimated_goal.position.x = tf_pose.position.x +goal.position.x*xbase1
+goal.position.y*ybase1
estimated_goal.position.y = tf_pose.position.y +goal.position.x*xbase2
+goal.position.y*ybase2
estimated_goal.position.z = goal.position.z
estimated_goal.orientation.x = goal.orientation.x
estimated_goal.orientation.y = goal.orientation.y
estimated_goal.orientation.z = goal.orientation.z
estimated_goal.orientation.w = goal.orientation.w
rospy.loginfo("I am going publish:
(x,y,z)(%f,%f,%f)",estimated_goal.position.x,estimated_goal.position.y,estimated_goal.posit
ion.z)

point_publisher = rospy.Publisher('/estimated_tcp_goal', Pose ,queue_size=10)
```

```
point_publisher.publish(estimated_goal)

def send_estimated_goal():
rospy.init_node('help_points')

StartTest_Publisher = rospy.Publisher('/start_test', Bool ,queue_size=1)
goal_subscriber=rospy.Subscriber("/tcp_offset_from_base",Pose,callback_goal)

scitos_tf = tf.TransformListener()
rate = rospy.Rate(1)
while not rospy.is_shutdown():
try:
(trans,rot) = scitos_tf.lookupTransform('/map', '/base_link', rospy.Time(0))
tf_pose.position.x = trans[0]
tf_pose.position.y = trans[1]
tf_pose.position.z = trans[2]

except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException): continue

cosphi = np.arccos(rot[3])*2*180/np.pi
sinphi = np.arcsin(rot[2])*2*180/np.pi

tf_pose.position.x = trans[0]
tf_pose.position.y = trans[1]
tf_pose.position.z = trans[2]
tf_pose.orientation.z = rot[2]
tf_pose.orientation.w = rot[3]

rate.sleep()

if __name__ == '__main__':
send_estimated_goal()
```

test_cmdvel_vali.py [175]

```
import rospy
from geometry_msgs.msg import TwistStamped
from geometry_msgs.msg import Twist
from std_msgs.msg import Bool
from time import sleep
vel_msg = TwistStamped()
vel_msg2 =Twist()

def do_movement(lin,ang):
# For Documentation
vel_msg.header.stamp = rospy.Time.now()
vel_msg.twist.linear.x = lin
vel_msg.twist.linear.y = 0
vel_msg.twist.linear.z = 0
vel_msg.twist.angular.x = 0
vel_msg.twist.angular.y = 0
vel_msg.twist.angular.z = ang

#For real cmd velocity
vel_msg2.linear.x = lin
vel_msg2.linear.y = 0
vel_msg2.linear.z = 0
vel_msg2.angular.x = 0
vel_msg2.angular.y = 0
vel_msg2.angular.z = ang
```

```
def move():
# Starts a new node
cmdvel_linear_x = 0.5
cmdvel_angular_z = 0.0
wait_time=2
MVRobot = False
rospy.init_node('validate_algo', anonymous=True)
StartTest_Publisher = rospy.Publisher('/start_test', Bool ,queue_size=1)
velocity_publisher = rospy.Publisher('/cmd_vel_test', TwistStamped, queue_size=1)
velocity_publisher2 = rospy.Publisher('/cmd_vel',Twist,queue_size=1)
do_movement(0,0)

velocity_publisher.publish(vel_msg)
velocity_publisher2.publish(vel_msg2)
MVRobot = rospy.wait_for_message('/start_test', Bool, timeout=None)
if (MVRobot):
rospy.loginfo("Begin the Test: Move %f m/s linear and %f m/s angular. Wait %i s and go
backwarts",cmdvel_linear_x,cmdvel_angular_z,wait_time)
sleep(1)
i = 0
itteration = int(wait_time/0.1) #Time for frequently messages
for i in range(itteration):
do_movement(cmdvel_linear_x,cmdvel_angular_z)
velocity_publisher.publish(vel_msg)
velocity_publisher2.publish(vel_msg2)
sleep(0.1)

for i in range(itteration):
do_movement(0,0)
velocity_publisher.publish(vel_msg)
velocity_publisher2.publish(vel_msg2)
sleep(0.1)

for i in range(itteration):
do_movement(-cmdvel_linear_x,-cmdvel_angular_z)
velocity_publisher.publish(vel_msg)
velocity_publisher2.publish(vel_msg2)
sleep(0.1)

for i in range(itteration):
do_movement(0,0)
velocity_publisher.publish(vel_msg)
velocity_publisher2.publish(vel_msg2)
sleep(0.1)

else:
do_movement(0,0)
velocity_publisher.publish(vel_msgs)
velocity_publisher2.publish(vel_msg2)
sleep(0.1)
if __name__ == '__main__':

try:
move()
except rospy.ROSInterruptException: pass
```

imu_scitos.cpp [175]

```
#include "ros/ros.h"
#include <visualization_msgs/Marker.h>
#include <sensor_msgs/Imu.h>
```

```
#include <iostream>
#include <string>
#include <unistd.h>
#include <cstdlib>
#include <boost/algorithm/string.hpp>
#include "tf/tf.h"

const double PI = 3.14159;
double markerposx,markerposy,markerposz,markerposw;
int i;
//create Global value --> That I can use it in All functions
sensor_msgs::Imu imu_without_gravity;//Create the Imu withouth gravity

void linearACCwoG(const sensor_msgs::Imu::ConstPtr &imu_msg)
{
//ROS_INFO("I AM HERE! IMU_SCITOS_VIRT WORK");
//callback function for imu_subscriber
//relevant imu without acceleration variables

imu_without_gravity.linear_acceleration.x = imu_msg -> linear_acceleration.x;
imu_without_gravity.linear_acceleration.y = imu_msg -> linear_acceleration.y;
imu_without_gravity.linear_acceleration.z = imu_msg -> linear_acceleration.z-9.81;

// Pose of the marker
markerposx = imu_msg -> orientation.x;
markerposy = imu_msg -> orientation.y;
markerposz = imu_msg -> orientation.z;
markerposw = imu_msg -> orientation.w;
}

int main(int argc, char **argv)
{
ros::init(argc, argv, "imu_scitos"); //node name
// create a node handle; need to pass this to the class constructor
ros::Rate loop_rate(10);  //set a desired run time of a cycle in Hz

ros::NodeHandle n;
ros::Publisher marker_pub = n.advertise<visualization_msgs::Marker>("marker",1);
ros::Publisher imu_without_gravity_pub =
n.advertise<sensor_msgs::Imu>("linear_acc_without_gravity",1);
ros::Subscriber imu_sub = n.subscribe("imu",1,linearACCwoG);
uint32_t shape = visualization_msgs::Marker::ARROW;

// Enter the main loop
for(i=0;1<10;)
{
// Create a marker to visualize the IMU data
visualization_msgs::Marker marker;

marker.header.frame_id = "/c_o_g";
marker.ns = "basic_shapes";
marker.id = 0;
marker.type = shape;
marker.action = visualization_msgs::Marker::ADD;

// Position of the marker
marker.pose.position.x = 5;
marker.pose.position.y = 5;
marker.pose.position.z = 5;

// Pose of Marker
marker.pose.orientation.x = markerposx;
```

```
marker.pose.orientation.y = markerposy;
marker.pose.orientation.z = markerposz;
marker.pose.orientation.w = markerposw;

// Scale of the marker
marker.scale.x = 1.0;
marker.scale.y = 1.0;
marker.scale.z = 1.0;

// Set the color -- be sure to set alpha to something non-zero!
marker.color.r = 0.0f;
marker.color.g = 1.0f;
marker.color.b = 0.0f;
marker.color.a = 1.0;

marker.lifetime = ros::Duration();

imu_without_gravity_pub.publish(imu_without_gravity);

marker_pub.publish(marker);

ros::spinOnce();
}
}
```

visualize_workingspace.py [175]

```
import rospy
import std_msgs.msg
from geometry_msgs.msg import Point
from geometry_msgs.msg import Vector3
#from vector_3_array.msg import Vector3DArray
from sensor_msgs.msg import PointCloud
import numpy as np
from geometry_msgs.msg import PoseArray
from geometry_msgs.msg import Pose

pc_workspace=PointCloud()
#new_point= Vector3DArray()
next_point = Pose()
next_point_array=PoseArray()
X_kRIT = 0.7
MAX_WS_RADIUS = 1.2

if __name__ == '__main__':
rospy.init_node('visualize_workingspace')
pub_pc_workspace = rospy.Publisher('/visi_pc', PointCloud, queue_size = 1)
rate = rospy.Rate(10.0)

pc_workspace.header.frame_id = "arm_podest_link"
pc_workspace.header.stamp = rospy.Time.now()
pc_workspace.points = 100
pc_workspace.channels = 1
next_point_array.poses = 100
x=-MAX_WS_RADIUS
y=-MAX_WS_RADIUS
s= 0
while x<0:
while y<0:
cood_ws = MAX_WS_RADIUS**2-x**2-y**2
if cood_ws >=0:
```

```
z=np.sqrt(cood_ws)
next_point.position.x = x
next_point_array.poses.append(next_point)
s= s+1
x = x+0.1
y=y+0.1
x=0
y=0
while x<X_KRIT:
while y<MAX_WS_RADIUS:
cood_ws = 1-(x/X_KRIT)**2-(y/MAX_WS_RADIUS)**2
if cood_ws >0:
z = np.sqrt(cood_ws)*MAX_WS_RADIUS
new_point.x=x
new_point.y=y
new_point.z=z
pc_workspace.points.append(new_point)
i= i+1
x = x+0.1
y=y+0.1
while not rospy.is_shutdown():
pub_pc_workspace.publis(pc_workspace)
rate.sleep()
```

Additionally, source code from the following digital repositories was employed:

For `ros_controllers` package
https://github.com/ros-controls/ros_controllers.git

For `schunk_modular_robotics` package
https://github.com/ipa320/schunk_modular_robotics.git

For `schunk_robots` package
https://github.com/ipa320/schunk_robots.git

For `scitos_common` package
https://github.com/cburbridge/scitos_common

For `scitos_driver` package
https://github.com/strands-project/scitos_drivers

For Recursive Newton Euler Inverse Dynamics Solver
http://docs.ros.org/en/kinetic/api/orocos_kdl/html/classKDL_1_1ChainIdSolver__RNE.html

For The STRANDS Project: Long-Term Autonomy in Everyday Environments
https://arxiv.org/abs/1604.04384

For Navigation control
http://library.isr.ist.utl.pt/docs/roswiki/navigation(2f)Tutorials(2f)SendingSimpleGoals.html