OPTIMAL TRAJECTORY GENERATION AND ADAPTIVE CONTROL OF AN UNDERACTUATED AND SELF-BALANCING LOWER BODY EXOSKELETON

A Dissertation

by

Ahmed Adel Ahmed Fahmy Soliman

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in the
Department of Mechanical Engineering

Özyeğin University
May 2023

Copyright © 2023 by Ahmed Adel Ahmed Fahmy Soliman

OPTIMAL TRAJECTORY GENERATION AND ADAPTIVE CONTROL OF AN UNDERACTUATED AND SELF-BALANCING LOWER BODY EXOSKELETON

Approved by:

Assistant Professor Barkan Uğurlu,
Advisor
Department of Mechanical Engineering
Özyeğin University

Associate Professor Kemalettin Erbatur
Department of Mechatronics Engineering
Sabanci University

Assistant Professor Özkan Bebek
Department of Mechanical Engineering
Özyeğin University

Associate Professor Koray Kadir Şafak
Department of Mechanical Engineering
Yeditepe University

Date Approved: 29 May 2023

Assistant Professor Ramazan Ünal
Department of Mechanical Engineering
Özyeğin University

I dedicate this dissertation to my beloved family.

ABSTRACT

This thesis presents an approach for developing three-dimensional (3D) dynamic walking capabilities in a bipedal exoskeleton with underactuated legs. The proposed framework consists of a trajectory generator and an optimized inverse kinematics algorithm designed to handle underactuation. To achieve feasible task velocities despite the underactuated legs, the inverse kinematics algorithm utilizes a task prioritization method by exploiting the null space. This approach allows lower-priority tasks, such as swing foot orientation, to be accomplished to the greatest extent possible without interfering with higher-priority tasks like the Center of Mass trajectory. Simultaneously, the trajectory generator analytically incorporates the zero moment point concept, ensuring continuous acceleration throughout the entire walking period, regardless of changes in contact and phase.

Furthermore, three locomotion controllers were developed to complement the proposed task prioritization algorithm and enhance its robustness against significant parameter uncertainty and external disturbances. These controllers include the zero moment point impedance feedback controller, along with two other state-of-the-art locomotion controllers: admittance control and centroidal momentum control. The objective is to integrate these controllers with the task prioritization algorithm, collectively improving the system's ability to handle challenging conditions and uncertainties. A series of simulation experiments were conducted using a 3D simulator to verify the validity and robustness of these controllers for thorough benchmarking. A human-robot coupled model is considered, including a 40 kg underactuated exoskeleton and 12 distinct anthropomorphic subjects. When combined with the proposed task priority-based optimization algorithm, all three controllers demonstrate adequate performance in addressing balanced locomotion behavior. The proposed zero moment point impedance controller shows statistically significant

results, indicating a comparatively more robust feature.

As the proposed locomotion controller approaches are model-based controllers, there is a desire to develop a real-time applicable inertial parameter identification algorithm to improve the locomotion controller's adaptability against inertial parameter variations. A semidefinite programming algorithm is developed to perform the identification algorithm recursively while guaranteeing the complete physical consistency of the identified inertial parameters. A recursive algorithm to update the identifiability projection matrix is developed to manage the inclusion of newly acquired samples and arrange them concerning the old identifiable parameters. The idea of the filtered regressor is used to mitigate the effect of contact transitions and noise without losing information about the identifiable parameters. To verify the validity of the proposed identification algorithm, a series of simulation experiments are conducted using 3D simulator using the human-robot coupled model. As a result, the algorithm shows feasible performance based on accuracy, computation time, and the complete physical consistency of the identified parameters.

To verify the proposed algorithm, an exoskeleton prototype was constructed. The prototype was equipped with eight series elastic actuators, sixteen force-sensitive resistors for measuring contact force, and another sixteen absolute encoders for measuring motor angular displacements and series elastic actuator spring deflections. Communication with series elastic actuators and sensors was enabled through a set of interface circuits and a desktop PC. The real-time operation was justified by employing Ubuntu 18.04 and Xenomai 3.1. The required Cartesian and joint-level controllers were programmed using the C language with the GNU scientific library. Simultaneous resolution of both algorithms was facilitated through parallel programming. Experimental development of squat, sway, and sagittal walk motions was carried out. The real-time applicability and feasibility of the proposed algorithms, as well as the developed hardware, were demonstrated by the experiments.

ÖZETÇE

Bu çalışma, eksik tahrikli bir dış iskelet robotu kullanılarak 3B dinamik öz-dengeli yürüme hareketi geliştirmek için bir yaklaşım sunmaktadır. Bu yaklaşım, hareketin kütle merkezi ile sağ ve sol ayakların merkez Kartezyen yörüngelerini oluşturmak için sıfır moment noktası (ZMP) kavramını ve bir dizi polinomu kullanmaktadır. Önerilen yaklaşım, bir optimizasyon algoritması ile robotun ters kinematiğini görev önceliklendirme formunda çözmekte ve bu açıdan önemli bir rol oynamaktadır. Söz konusu robotun eksik tarhrikli doğası nedeniyle tüm istenen Kartezyen yörüngeleri tam olarak elde etmek mümkün değildir. Bu nedenle, söz konusu yörüngelere belli bir ölçüye kadar yakın bir ters kinematik çözümü geliştirmek için bir dış bükey optimizasyon algoritması kullanılmaktadır. Bu bağlamda, kütle merkezi yörüngesinin öncelikli olduğu kısıtlamaları içeren hiyerarşik bir yapı ilgili ters kinematik için oluşturulmaktadır. İkinci olarak, daha düşük öncelikli kısıtlamaların (örneğin ayak ve gövde yönelimi) çözümü, yüksek öncelikli kısıtlamaların null uzayına yansıtılarak gerekli denge koşulunu doğrulamaktadır.

Karalı hareket için, sıfır moment noktasını baz alan empedans geribeslemeli bir hareket kontrolcüsü ve diğer iki hareket kontrolcüsü (oryantasyon kontrolü ve merkezi momentum kontrolü) önerilen görev önceliklendirme algoritmasına adapte edilmiştir. Bu kontrolcülerin geçerliliğini ve gürbüzlüğünü detaylı bir şekilde karşılaştırmak için MSC ADAMS kullanılarak bir dizi simülasyon deneyi gerçekleştirilmiştir. Bir insan-robot birleşik modeli, 40 kg alt hareketli bir dış iskelet ve 12 farklı antropomorfik insan modelini içermektedir. Sonuç olarak, önerilen görev öncelikli optimizasyon algoritmasıyla birleştirildiğinde, üç kontrolcü de dengeli hareket davranışı için yeterli performans sergilemektedir. Buna mukabil, önerilen sıfır moment noktası bazlı empedans kontrolcüsü, göreli olarak daha gürbüz bir performans göstermiş ve bu veriler istatistiksel olarak irdelenmiştir..

Önerilen hareket kontrolcü yaklaşımları model tabanlı kontrolcüler olduğundan, hareket kontrolcüsünün parametre değişikliklerine karşı uyum sağlayabilmesini iyileştirmek için gerçek zamanlı uygulanabilir bir parametre tanımlama algoritması ihtiyacı hasıl olmuştur. Tanımlama algoritması, tanımlanan hareketli parametrelerin tam fiziksel tutarlılığını garanti edecek şekilde özyinelemeli olarak yarı kesirli programlama algoritması kullanılarak geliştirilmiştir. Ek olarak, tanımlanabilirlik projeksiyon matrisini güncellemek için başka bir özyinelemeli algoritma geliştirilmiş ve yeni edinilen örneklerin eski tanımlanabilir parametrelerle düzenlenmesi icra edilmiştir. Tanımlanabilir parametrelerle ilgili bilgileri kaybetmeden kontak geçişlerinin ve gürültünün etkisini azaltmak için süzülmüş regresör kullanılmaktadır. Önerilen tanımlama algoritmasının geçerliliğini doğrulamak için, yukarıda bahsedilen insan-robot birleşik modeli kullanılarak Raisim simülatörü aracılığıyla bir dizi simülasyon deneyi gerçekleştirilmiştir. Sonuç olarak, algoritma, tanımlanan parametrelerin doğruluğu, hesaplama süresi ve tam fiziksel tutarlılık temelinde uygun performans sergilemektedir.

Önerilen algoritmayı doğrulamak için, Özyeğin Üniversitesi Biyomekatronik Laboratuvarı bünyesinde inşa edilmiş bir dış iskelet prototipi (Co-Ex) oluşturulmuştur. Sekiz seri elastik eyleyci içeren bu prototip temas kuvvetlerini ölçmek için on altı sensör içerir. Ayrıca, motor açılarıyla yay burulmalarını ölçmek için yüksek çözünürlüklü on altı mutlak kodlayıcı ile donatılmıştır. Eyleyici ve sensörlerle iletişim kurabilmek için bir dizi arayüz devresi ve bir masaüstü bilgisayar kullanılmaktadır. Gerçek zaman kısıtlamalarını doğrulamak için Linux Ubuntu 18.04 OS, Xenomai 3.1 yamasıyla kullanılmaktadır. Gerekli kontrolcüleri programlamak için GNU bilimsel kütüphanesiyle birlikte C dili kullanılmaktadır. Paralel programlama, her iki algoritmayı aynı anda çözebilmek için kullanılmaktadır. Squatting ve swaying hareketleri, Co-Ex kullanılarak deneysel olarak geliştirilmiştir. Deneyler, önerilen algoritmaların ve geliştirilen donanımın gerçek zamanlı uygulanabilirliğini kanıtlamaktadır.

ACKNOWLEDGEMENTS

I want to express my appreciation to my advisor, Assist. Prof. Barkan Uğurlu, for his wisdom, friendship, encouragement, support, patient guidance, invaluable insights, and suggestions during this experience as a Ph.D. student and research assistant. The work of the thesis would not be possible without his consistent support and guidance. I also had a great opportunity to work with him. I learned a lot from him throughout my Ph.D. studies.

I am grateful to the members of my dissertation committee, Asst. Prof. Özkan Bebek, Asst. Prof. Ramazan Ünal, Assoc. Prof. Kemalettin Erbatur and Assoc. Prof. Koray Kadir Şafak for their time serving on my committee and carefully reviewing my dissertation.

I very gratefully acknowledge the funding received towards my Ph.D. from The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grants 215E138 (01/02/2018–15/04/2020) and 118E923 (01.03.2021 – 01.02.2023). I am also grateful for technical and non-technical discussions with my colleagues in the Biomechatronics Lab research group at Özyeğin University. I have also enjoyed the collaboration and fruitful conversations.

I would like to express my deepest appreciation, especially to my parents, and more broadly to my entire family, for their unwavering support. I am eternally grateful to my family for their boundless trust and willingness to stand by me.

ACRONYMS

BAC	basic admittance controller
CI	confidence interval
CMC	centroidal momentum controller
CMM	centroidal momentum matrix
CoM	Center of Mass
DSP	Double Support Phase
E/E	Endo/Exo
F/E	Flexion/Extension
FBID	Floating Base Inverse Dynamics
FK	Forward Kinematics
FSR	Force Sensitive Resistors
ID	Inverse Dynamics
IK	Inverse Kinematics
IKCL	Inverse Kinematics Closed Loop
IPID	Inertial Parameter Identification
IPM	Interior Point Method
LMI	Linear Matrix Inequalities
MZIF	Modified ZMP impedance feedback
NE	Newton-Euler
PST	prioritized stack of tasks
RIPM	Recursive Identifiability Projection Matrix
ROM	Range of Motion
SCI	Spinal Cord Injury

SEA	Series Elastic Actuators
SDP	Semidefinite Programming
SSP	Single Support Phase
ZIF	ZMP impedance feedback
ZMP	Zero Moment Point

SYMBOLS

$f_{\varsigma,f}$	FSR sensor s attached to the foot f force signal
$v_{\varsigma,f}$	FSR sensor s attached to the foot f voltage signal
$fit(\bullet)$	MATLAB curve fitting function
$a_{\varsigma,f}, b_{\varsigma,f}$	Exponential function coefficients for sensor ς and foot f
$_f$	Subscript denotes foot index $f \in \{rf, lf\}$
$_{rf}$	Subscript denotes right foot
$_{lf}$	Subscript denotes left foot
$_\varsigma$	Subscript denotes FSR sensor index
$f_{f,z}^{pi}$	Vertical force at each point pi
$m_{f,x}^{pi}$	Moment at each point pi around foot sole center in x direction
$m_{f,y}^{pi}$	Moment at each point pi around foot sole center in y direction
$a_{z,f}^{pi}, b_{z,f}^{pi}$	Vertical force at point pi linear function coefficients
$a_{x,f}^{pi}, b_{x,f}^{pi}$	Moment in x direction at point pi linear function coefficients
$a_{y,f}^{pi}, b_{y,f}^{pi}$	Moment in y direction at point pi linear function coefficients
$\overline{a}_{z,f}, \overline{b}_{z,f}$	Aggregated vertical force concerning foot f
$\overline{a}_{x,f}, \overline{b}_{x,f}$	Aggregated moment in x direction concerning foot f
$\overline{a}_{y,f}, \overline{b}_{y,f}$	Aggregated moment in y direction concerning foot f
t_s	Sampling rate
n_s	Number of samples
$f_{f,z}$	Aggregated vertical force linear function

$m_{f,x}$	Aggregated moment in x direction
$m_{f,y}$	Aggregated moment in y direction linear function coefficients
J_m	Motor side inertia
B_m	Motor side viscous friction coefficient
J_l	Load side inertia
B_l	Load side viscous friction coefficient
θ_m	Motor side angular displacement
θ_l	Load side angular displacement
θ_s	Spring deflection
N_G	Gear ratio
K_s	Spring stiffness
d_l	Load side disturbance
d_m	Motor side disturbance
τ_s	Spring torque
τ_m	Motor torque
s	Laplace variable
$smooth(\bullet)$	MATLAB smooth function
$P_m(s)$	Motor transfer function
$P_l(s)$	Load transfer function
$P_n(s)$	Transfer function between τ_s and τ_m
τ_{ID}	Inverse dynamics solution
τ_{ff}	Feed forward torque
$Q_2(s)$	Second order filter transfer function
w_c	Cut-off frequency
ζ	Damping ratio
k_t	Motor constant

J_{eq}	Equivalent inertia as projected on the motor shaft
B_{eq1}, B_{eq2}	Equivalent viscous friction coefficients in clockwise and counterclockwise directions
τ_{st1}, τ_{st2}	Equivalent stiction friction coefficients in clockwise and counterclockwise directions
$sgn(\bullet)$	Sign function
$\overline{\boldsymbol{\phi}}_k$	Regressor k^{th} row
$\boldsymbol{\Phi}$	Regressor matrix
$\overline{\boldsymbol{\tau}}_m$	A vector that stores the actual motor torques at different samples
$\hat{J}_{eq}$	Estimated equivalent inertia
$\hat{B}_{eq1}, \hat{B}_{eq2}$	Estimated equivalent viscous friction coefficients in clockwise and counterclockwise directions
$\hat{\tau}_{st1}, \hat{\tau}_{st2}$	Estimated stiction friction coefficients in clockwise and counterclockwise directions
$\hat{\tau}_m$	Estimated motor torque
$\tau_{m,a}$	Actual motor torque
τ_l^{ref}	Reference load torque
τ_l^a	Actual load torque
$freq$	Frequency
$corr(\bullet)$	MATLAB correlation function
$\mathbf{q}$	Joint configuration vector
$\mathbf{x}_b$	Floating base joint configuration vector
$\mathbf{q}_r$	Robot joint configuration vector
n_r	Number of the robot active joints
m_i	Mass of link i
$\mathbf{J}_{i,com}$	The i^{th} links' *CoM* Jacobian matrix

$\mathbf{J}_{com}$	*CoM* Jacobian matrix
$\mathbf{J}_{torso}$	Torso Jacobian matrix
$\mathbf{J}_{rf}$	Right foot Jacobian matrix
$\mathbf{J}_{lf}$	Left foot Jacobian matrix
$\mathbf{J}$	Jacobian matrix
$\dot{\mathbf{x}}^{cmd}$	Command Cartesian velocity trajectories
$\mathbf{J}_{c,b}$	Constraint Jacobian sub-matrix that corresponds to the floating base joint coordinates variables
$\mathbf{J}_{c,r}$	Constraint Jacobian sub-matrix that corresponds to the robot joint coordinates variables
n_c	Number of feet in contact with the ground
$\mathbf{1}_n$	Identity matrix of order n
$\mathbf{x}^{des}$	Desired Cartesian trajectories
$\mathbf{x}^{FK}$	Forward kinematics solution
F_G	Global coordinate frame
F_0	Floating base coordinate frame
F_{rf}	Right foot coordinate frame
F_{lf}	Left foot coordinate frame
$\delta(\dot{\mathbf{q}})$	Residual function between the desired and actual task trajectories in terms of robot joint velocities
p	Subscript denotes the prioritization level
$\mathcal{N}_{p-1}$	Null space matrix of the prioritization level $p-1$
$\mathbf{b}$	A vector contains a set of optimized variables concerning the constraints with less priority
t_s	Sampling time
$\overline{\dot{\mathbf{q}}}_r$	A vector contains the maximum allowed joint velocities

$\underline{\dot{\mathbf{q}}}_r$	A vector contains the minimum allowed joint velocities
$\psi(\dot{\mathbf{q}}_r)$	A vector contains a set of logarithmic barrier functions
$\mathbb{C}^n$	A set of constant values of order n
$\tilde{\mathbf{x}}$	Closed loop correction term
$\overline{\boldsymbol{\mu}}$	A positive coefficients row vector
μ	A positive scalar
$ones(1, n_r)$	A Function returns a row vector of ones with a length of n_r
$\mathbf{H}(\mathbf{q})$	Inertia matrix
$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$	Coriolis and centrifugal matrix
$\mathbf{G}(\mathbf{q})$	Gravitational vector
$\mathbf{S}$	Actuated joints selection matrix
$\boldsymbol{\lambda}_c$	A vector that stores spatial forces at contacts
$\boldsymbol{\tau}$	Robot joints torque vector
$\boldsymbol{\lambda_{c,i}}$	Spatial contact force vector for body i
$\lambda_{c,j}$	Spatial Contact force component j
$m_{c,x}$	Contact moment in x direction
$m_{c,y}$	Contact moment in y direction
$m_{c,z}$	Contact moment in z direction
$\lambda_{c,x}$	Contact force in x direction
$\lambda_{c,y}$	Contact force in y direction
$\lambda_{c,z}$	Contact force in z direction
$\mathbf{I}_{fb}$	The aggregated links' spatial inertia projected onto the floating-base

$\overline{\mathbf{F}}$	Matrix whose columns store the spatial forces required at the floating-base to provide unit acceleration for each joint variable
$\mathbf{H}_r$	Robot joints inertia matrix
${}^{0}\mathbf{X}_{c}^{*}$	Spatial transformation matrix in the force space defined from the contact frame to the base frame
$\mathbf{S}_{fb}$	Floating base virtual joints selection matrix
$\mathbf{s}_{c,j}$	Contact force component selection vector
${}^{lf}\mathbf{X}_{0}^{*}$	Spatial transformation matrix in the force space defined from the base frame to the left foot frame
${}^{rf}\mathbf{X}_{0}^{*}$	Spatial transformation matrix in the force space defined from the base frame to the right foot frame
$\mathbf{A}_g$	Centroidal momentum matrix
$\mathbf{h}_g$	Centroidal momentum vector
$\mathbf{k}_g$	Angular centroidal momentum vector
$\mathbf{l}_g$	Linear centroidal momentum vector
$\dot{\mathbf{h}}_g$	Rate of change of centroidal momentum vector
$\dot{\mathbf{k}}$	Rate of change of angular centroidal momentum vector
$\dot{\mathbf{l}}_g$	Rate of change of linear centroidal momentum vector
$\dot{\mathbf{A}}_g$	Rate of change of centroidal momentum matrix
${}^{com}\mathbf{X}_{0}^{*}$	Spatial transformation matrix in the force space defined from the floating base frame to the CoM frame
χ	Optimization variable
$\mathbf{A}_{eq}$	Full row rank linear equality constraint matrix
$\mathbf{b}_{eq}$	Equality constraint vector
$\mathbf{A}_{ineq}$	Inequality constraint matrix

$\mathbf{b}_{ineq}$	Inequality constraint vector
${}^{cmd}\mathbf{p}_T$	Feet position and orientation command vector
${}^{cmd}\mathbf{p}_{com}$	CoM position command vector
${}^{cmd}\mathbf{p}_{torso}$	Torso orientation command vector
${}^{cmd}\mathbf{p}$	Task space command vector
$\mathbf{J}_T$	Feet position and orientation Jacobian
μ_s	Static friction coefficient
w_s	Support polygon width
L_s	Support polygon length
x_{zmp}	ZMP displacement in x direction
y_{zmp}	ZMP displacement in y direction
x_{com}	CoM displacement in x direction
y_{com}	CoM displacement in y direction
$\ddot{x}_{com}$	CoM acceleration in x direction
$\ddot{y}_{com}$	CoM acceleration in y direction
g	Gravitational acceleration
x_{so}	SSP initial CoM displacement in x direction
$\dot{x}_{so}$	SSP initial CoM velocity in x direction
y_{so}	SSP initial CoM displacement in y direction
$\dot{y}_{so}$	SSP initial CoM velocity in y direction
x_{do}	DSP initial CoM displacement in x direction
$\dot{x}_{do}$	DSP initial CoM velocity in x direction
y_{do}	DSP initial CoM displacement in y direction
$\dot{y}_{do}$	DSP initial CoM velocity in y direction
Str	Stride length
t_{ss}	SSP time span
t_d	DSP time span

t_{ms}	Middle SSP time span
t_{md}	Middle DSP time span
x_f	Swing foot displacement in x direction
$x_{f,0}, \dot{x}_{f,0}, \ddot{x}_{f,0}$	Swing foot initial displacement, velocity and acceleration in x direction, respectively
$x_{f,f}, \dot{x}_{f,f}, \ddot{x}_{f,f}$	Swing foot final displacement, velocity and acceleration in x direction, respectively
z_f	Swing foot displacement in z direction
$z_{f,0}, \dot{z}_{f,0}, \ddot{z}_{f,0}$	Swing foot initial displacement, velocity and acceleration in z direction, respectively
$z_{f,f}, \dot{z}_{f,f}, \ddot{z}_{f,f}$	Swing foot final displacement, velocity and acceleration in z direction, respectively
$qr(\bullet)$	QR decomposition function
$\mathbf{Q}$	Orthogonal matrix
$\mathbf{R}$	Upper triangle matrix
$\boldsymbol{\lambda}_{c,id}$	Contact force obtained via Inverse dynamic "Direct Decomposition" algorithm
t	Time variable
$\hat{\boldsymbol{\lambda}}_c$	Estimated contact force
$\boldsymbol{\tau}_{act}$	Actual robot joint torques
$\hat{\lambda}_{c,z}$	Estimated contact force in z direction
$\hat{m}_{c,x}$	Estimated contact moment in x direction
$\hat{m}_{c,y}$	Estimated contact moment in y direction
τ_k	Knee motor torque
z_{fb}^{act}	Floating base coordinate frame actual vertical position
z_{fb}^{ref}	Floating base coordinate frame reference vertical position

$\boldsymbol{\omega}_{fb}^{error}$	Floating base orientation velocity error
$\boldsymbol{\omega}_{fb}^{ref}$	Floating base orientation velocity reference
$\boldsymbol{\omega}_{fb}^{act}$	Floating base orientation velocity actual value
$\boldsymbol{\lambda}_{c}^{or}$	Orientation control contact force
$\mathbf{K}_{or}$	Diagonal positive semi-definite gain matrix
$\boldsymbol{\tau}^{or}$	Orientation control torque signal
y_{fb}^{act}	Floating base coordinate frame reference lateral position
y_{fb}^{ref}	Floating base coordinate frame reference lateral position
t_c	Computation time
e_{peak}	Peak error
$\theta_{x,fb}^{act}$	Floating base roll
${}^{ref}\boldsymbol{\lambda}_c$	Reference contact force
${}^{act}\boldsymbol{\lambda}_c$	Actual contact force
$\boldsymbol{\lambda}_{error}$	Contact force error
$\mathscr{L}^{-1}(\bullet)$	Inverse of the Laplace operator
$\mathbf{T}(\bullet)$	Diagonal matrix contains admittance blocks on its diagonal corresponding to each axis
b_j	Damping coefficient concerning the j^{th} force component
k_j	Stiffness coefficient concerning the j^{th} force component
$\dot{\mathbf{p}}_{AC}$	Admittance control action
${}^{ref}\dot{\mathbf{p}}, {}^{ref}\ddot{\mathbf{p}}$	Reference velocity and acceleration in Cartesian space
$\mathbf{K}_p$	Diagonal matrix that stores proportional force feedback gains

${}^{cmd}\dot{\mathbf{p}}, {}^{cmd}\ddot{\mathbf{p}}$	Commanded velocity and acceleration in Cartesian space
$\mathbf{M}_{p,1}, \mathbf{M}_{p,2}, \mathbf{M}_d$	Positive semidefinite diagonal gain matrices
${}^{ref}\mathbf{h}_g$	Reference centroidal momentum signal
${}^{error}\mathbf{h}_g$	Error centroidal momentum signal
${}^{ref}\boldsymbol{\nu}_{fb}$	Reference floating base velocity
${}^{act}\boldsymbol{\nu}_{fb}$	Actual floating base velocity
${}^{ref}\boldsymbol{\lambda}_{com}$	Reference contact forces as projected on the CoM
${}^{ref}\mathbf{h}_g$	Reference centroidal momentum signal
${}^{error}\mathbf{h}_g, {}^{error}\dot{\mathbf{h}}_g$	Error centroidal momentum and its rate of change signals
${}^{cmd}\mathbf{h}_g, {}^{cmd}\dot{\mathbf{h}}_g$	Command centroidal momentum and its rate of change signals
${}^{ref}\mathbf{p}_T$	Feet trajectory references
λ_{sx}	Virtual ZIF spring-damper system corresponding force
K_x	Scaling factor
$\ddot{\mathbf{p}}_{zif,x}, \ddot{\mathbf{p}}_{zif,y}$	Base acceleration that corresponds to the x-axis and y-axis ZMP correction action
$\dot{\mathbf{p}}_{zif,x}, \dot{\mathbf{p}}_{zif,y}$	Base velocity that corresponds to the x-axis and y-axis ZMP correction action
$\boldsymbol{\iota}$	Simplification variable
$\mathbf{K_{1,x}}, \mathbf{K_{1,y}}, \mathbf{K_{2,x}}, \mathbf{K_{2,y}}$	Positive definite diagonal gain matrices
x_{zmp}^{pp}	Average peak to peak value concerning x-axis ZMP
y_{zmp}^{pp}	Average peak to peak value concerning y-axis ZMP
ω_y^{pp}	Average peak to peak value concerning pitch axis torso angular velocity
$ranova(\bullet)$	Repeated Measures ANOVA MATLAB function

$multcompare(\bullet)$	Post-hoc test MATLAB function
$\Delta\mu$	Mean difference
σ_{Δ}^{2}	Mean squared error
M	Mean
SD	Standard deviation
p	pValue
F	fValue
$SC\#1$	Scenario 1
$SC\#2$	Scenario 2
$\delta\boldsymbol{\lambda}_{zif}$	Reaction forces that result from the default spring damper model of the ZIF controller
$\overline{\mathbf{x}}_{zmp}$	A vector contains the measured ZMP variations in x and y directions
$\mathbf{K}_{zif}$	Modified ZIF controller stiffness matrix
$\mathbf{B}_{zif}$	Modified ZIF controller viscous damping coefficients matrix
$\delta\boldsymbol{m}_{x,y}$	Vector representing the contact moments in x, and y directions
m_r	Robot aggregated mass
$\mathbf{L}_{zif}$	Symbol defined for simplification
$\delta\mathbf{h}_{x,y}$	Vector represent contact momentums in x, and y directions
k	Time sample index
$\mathbf{I}_{x,y}^{k}$	Integral term value at iteration k
$\delta\dot{\mathbf{q}}_{zif}$	Modified ZIF control action
$\dot{\mathbf{q}}_{ref}$	Referential joint velocities trajectories
$\dot{\mathbf{q}}_{cmd}$	Commanded joint velocities trajectories

$\dot{\mathbf{x}}^k_{cmd}$	Commanded Cartesian trajectories in the velocity level
$\dot{\mathbf{x}}^k_{ref}$	Referential Cartesian trajectories in the velocity level
${}^{ad}\ddot{\mathbf{p}}_{zif,x}$	Adaptive ZIF control action in the acceleration level
$\wedge$	Accent refers to the estimated variables
$\hat{\mathbf{I}}_{fb}$	Estimated aggregated links' spatial inertia projected onto the floating-base
$\hat{\mathbf{H}}(\mathbf{q})$	Estimated inertia matrix
$\hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})$	Estimated Coriolis and centrifugal matrix
$\hat{\mathbf{G}}(\mathbf{q})$	Estimated gravitational vector
$\hat{\mathbf{A}}_g$	Estimated centroidal momentum matrix
$\dot{\hat{\mathbf{A}}}_g$	Estimated rate of change of centroidal momentum matrix
${}^{ad}\delta\dot{\mathbf{q}}_{zif}$	Adaptive modified ZIF control action
${}^{ad}\dot{\mathbf{q}}^k_{mac}, {}^{ad}\ddot{\mathbf{q}}^k_{mac}$	Adaptive MAC joint velocities and accelerations vectors at the k^{th} iteration
$\overline{\mathbf{J}}$	Stack of Jacobian matrices
$\dot{\overline{\mathbf{J}}}$	Stack of rate of change of Jacobian matrices
$\hat{\overline{\mathbf{I}}}$	Block diagonal matrix whose blocks represent the estimated spatial inertia matrix per link
$\hat{\mathbf{B}}$	Block diagonal matrix function in the estimated inertial parameter
$\hat{\mathbf{O}}_i$	Matrix defined per link for simplification
$\overline{S}(\bullet)$	A function that returns a skew symmetric matrix for an input vector of size 3×1
$\boldsymbol{\omega}_i$	Angular velocity for each link fixed frame with respect to global frame

$\mathbf{v}_i$	linear velocity for each link fixed frame with respect to global frame
$\overline{\mathbf{X}}$	Stack of spatial spatial transformation matrix for each link
${}^{i}\mathbf{X}_0$	spatial transformation matrix for each link i fixed frame from coordinate frame $\{0\}$
${}^{B}\mathbf{E}_A$	Rotation matrix of coordinate frame A relative to coordinate frame B
${}^{B}\mathbf{r}_A$	Displacement vector of coordinate frame A relative to coordinate frame B
n_{dof}	Number of degrees of freedom
$\overline{\mathbf{h}}(\mathbf{q}, \dot{\mathbf{q}})$	Vector represents Coriolis ,centrifugal, and gravitational terms
$\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$	Regressor matrix
$\boldsymbol{\pi}$	Inertial parameter vector
$\boldsymbol{\pi}_i$	Link i inertial parameter vector
$\mathbf{f}$	Generalized force vector
n_s	Number of acquired samples
n_b	Number of rigid bodies
$\mathbf{f}_s$	Vector stores the generalized force vectors $\mathbf{f}$ over different samples
$\boldsymbol{I}_i$	Link i inertia tensor
$I_{i,xx}, I_{i,yy}, I_{i,zz}, I_{i,xy}, I_{i,xz}, I_{i,yz}$	Inertia tensor $\boldsymbol{I}_i$ components
$\boldsymbol{h}_i$	Link i first order moment
$h_{i,x}, h_{i,y}, h_{i,z}$	First order moment $\boldsymbol{h}_i$ vector components
m_i	Link i mass
$\rho_i(\mathbf{r_i})$	Link i density distribution

$\mathbf{r}_i$	Location of any arbitrary point within link i boundaries with respect to its axis of rotation
$\hat{\pi}$	Estimated inertial parameter vector
$\mathbf{I}_i^c$	Inertia tensor of link i around its center of mass
$\overline{\mathbf{O}}_i$	Orthogonal matrix used to diagonalize $\mathbf{I}_i^c$
$\mathbf{\Lambda}_i$	Diagonal matrix, whose elements represent the eigenvalues of $\mathbf{I}_i^c$
$\lambda_{i,1}, \lambda_{i,2}, \lambda_{i,3}$	Three diagonal elements of $\mathbf{\Lambda}_i$
$\lambda_{max}(\bullet)$	A function returns the maximum eigenvalue of the input matrix
$max(\bullet)$	Function that returns the maximum value of the input values
$tr(\bullet)$	Function that returns the trace of the input matrix
$\mathbf{r_{i,c}}$	Distance between the center of rotation and CoM concerning link i
$\mathbf{I}_{P,i}(\pi_i)$	Link i pseudo inertia matrix
$\mathbf{Y}_b$	Regressor columns corresponding to the base inertial parameters
$\mathbf{Y}_l$	Regressor columns corresponding to the linearly dependent inertial parameters
$\boldsymbol{\pi}_b$	Base inertial parameters
$\boldsymbol{\pi}_l$	Linearly dependent inertial parameters
n_{bp}	Number of base parameters
n_i	Number of identifiable inertial parameters
$\mathbf{Q}_b$	Column orthonormal matrix whose number of columns equals to n_{bp}
$\mathbf{R}_b$	Upper triangle square matrix of size n_{bp}

$\mathbf{K}$	Linearly dependent inertial parameters gain matrix
$\overline{\boldsymbol{\pi}}$	Vector stores the identifiable parameters
$\overline{\boldsymbol{\pi}}_i$	Vector stores the identifiable parameters for link i
$\mathbf{I}_{P,i}^0$	Pseudo inertia matrix as function in the initial guess of the non-identifiable parameters
$\overline{\boldsymbol{\Phi}}$	Identifiable inertial parameter regressor
$\bar{\mathbf{f}}$	Modified generalized force vector
γ	Regressor filter cutoff frequency
$\mathbf{I}_{s,i}$	Spatial inertia matrix for rigid link i
$\mathbf{f}_f$	Filtered force vector
$w(t)$	Impulse response of the first order filter
wi	Window index
W	Total number of the included windows
$\tilde{\boldsymbol{\Phi}}_W$	A matrix stores the regressors of the included windows in a single stack
$\tilde{f}_W$	A vector stores the force vectors of the included windows in a single stack
$\tilde{\boldsymbol{\pi}}^W$	A vector contains the identifiable parameters by $\tilde{\boldsymbol{\Phi}}_W$
$\hat{\tilde{\boldsymbol{\pi}}}^W$	The estimated value of $\tilde{\boldsymbol{\pi}}^W$
$\overline{\boldsymbol{\Phi}}_{wi}$	Window wi regressor
$\bar{\mathbf{f}}_{wi}$	Window wi generalized force vector
$n_{bp,wi}$	Number of base parameters for window wi
$n_{i,wi}$	Number of identifiable parameters for window wi
$\overline{\boldsymbol{P}}_{wi}$	Projection matrix of $\overline{\boldsymbol{\pi}}_{wi}$ onto $\tilde{\boldsymbol{\pi}}^W$
$m_{b,W}$	Sum of base parameter set sizes for all included windows
$m_{i,W}$	The size of identifiable parameters by $\tilde{\boldsymbol{\Phi}}_W$

$\overline{\mathbf{Q}}_W$	A positive definite symmetric matrix with dimension $m_{i,W}$
$\bar{\mathbf{c}}_W$	A real valued vector with dimension $m_{i,W}$
ε	Relative small positive scaler value
$\mathbf{Y}_{wi}$	Window wi regressor
$\mathbf{Y}_b^{wi}$	Regressor $\mathbf{Y}_{wi}$ linear independent columns
$\mathbf{Y}_l^{wi}$	Regressor $\mathbf{Y}_{wi}$ linear dependent columns
$\mathbf{V}_{wi}$	Matrix that stores the standard basis vectors of the non-zero columns of $\mathbf{Y}_{wi}$
$\mathbf{P}_{\mathbf{r},wi}$	Matrix $\mathbf{V}_{wi}$ permutation matrix
$\bar{\boldsymbol{I}}_{wi}$	Set of identifiable parameters by $\overline{\boldsymbol{\Phi}}_{wi}$
$\tilde{\boldsymbol{I}}_W$	Set of identifiable parameters by $\tilde{\boldsymbol{\Phi}}_W$
$\cap \boldsymbol{P}_W$	Projection matrix used to projects the inertial parameters located within $\bar{\boldsymbol{I}}_W \cap \tilde{\boldsymbol{I}}_{W-1}$ onto $\tilde{\boldsymbol{\pi}}^{W-1}$
$\bar{\boldsymbol{I}}_W \backslash \tilde{\boldsymbol{I}}_{W-1}$	Math operation returns the elements included in $\bar{\boldsymbol{I}}_W$ and excluded from $\tilde{\boldsymbol{I}}_{W-1}$
$\boldsymbol{\pi}_{\bar{i},W}^{\perp}$	Projection of $\tilde{\boldsymbol{\pi}}^W$ onto $\boldsymbol{\pi}$
$\mathrm{N}(\bullet)$	Function returns null space matrix of the input matrix
$\mathrm{C}(\bullet)$	Function returns the non-zero columns of the input matrix
$\mathrm{R}(\bullet)$	Function returns the non-zero rows of the input matrix
$\oplus$	Logical OR function
$\mathbf{I}_P^D(\tilde{\boldsymbol{\pi}}^W)$	Pseudo inertia Block diagonal matrix
$\tilde{\pi}_l^W$	The l^{th} parameter of vector $\tilde{\boldsymbol{\pi}}^W$
$\mathbf{G}_l$	Gradient matrix of $\mathbf{I}_P^D(\tilde{\boldsymbol{\pi}}^W)$ with respect to $\tilde{\pi}_l^W$
$\mathbf{I}_0^D$	Non-identifiable parameters pseudo inertia Block diagonal matrix

$\mathbf{Z}$	Slack variable matrix
$\mathbf{A}_{eq}$	Full row rank linear equality matrix
$\underline{\mathscr{L}}$	Lagrangian function
$log(\bullet)$	Logarithmic barrier function
$det(\bullet)$	Determinant function of the input matrix
$\mathbf{X}$	Positive definite symmetric matrix
$\mathbf{y}$	Lagrangian multiplier
$\mathbf{r}_\pi, \mathbf{r}_x, \mathbf{r}_z, \mathbf{r}_y$	Optimality conditions residual vectors
$\triangle\tilde{\boldsymbol{\pi}}^W, \triangle\mathbf{x}, \triangle\mathbf{z}, \triangle\mathbf{y}$	Newton steps
$\otimes_s$	Symmetric Kronecker product math operator
$svec(\bullet)$	Function returns an output vector that store the input symmetric matrix components
$smat(\bullet)$	Inverse of $svec(\bullet)$ function
h	Optimization iteration index
α	Newton step size
σ	Reduction parameter
$\varepsilon_\pi, \varepsilon_x, \varepsilon_y, \varepsilon_o$	Tolerance parameters
$\mathbf{S}_c$	Schur complement
c	Column index
e_{rel}	Relative error
$\mathbf{q}_{act}$	Actual joint displacements
$\mathbf{f}_{fb}$	Floating base generalized force vector
$\hat{\mathbf{f}}$	Estimated generalized force vector
$\mathbf{f}^{act}$	Actual generalized force vector
$\boldsymbol{\lambda}_{fb}$	Floating base inertia tensor eigen values vector
$\boldsymbol{\lambda}_h$	Hip inertia tensor eigen values vector
$\boldsymbol{\lambda}_{th}$	Thight inertia tensor eigen values vector

$\boldsymbol{\lambda}_{sh}$	Shank inertia tensor eigen values vector
$\boldsymbol{\lambda}_{f}$	Foot inertia tensor eigen values vector
m_{fb}	Floating base mass
m_{h}	Hip mass
m_{th}	Thight mass
m_{sh}	Shank mass
m_{f}	Foot mass
$\boldsymbol{\lambda}_{e,i}$	Spatial force vector specifying the external force acting on body i, expressed in frame $F_{i,0}$
i	Actuated joint/body index
$\psi(i)$	Body i parent number
N	An integer specifying the number of bodies in the mechanism
$\mathbf{X}_J(i)$	The coordinate transform from frame $F_{\lambda(i),1}$ to frame $F_{i,0}$
$\mathbf{X}_T(i)$	The Plücker coordinate transform from frame $F_{\lambda(i),0}$ to frame $F_{\lambda(i),1}$
${}^{i}\mathbf{X}_{\psi(i)}$	The Plücker coordinate transform from frame $F_{\psi(i),0}$ to frame $F_{i,0}$ for motion vectors
${}^{i}\mathbf{X}^{*}_{\psi(i)}$	The Plücker coordinate transform from frame $F_{\psi(i),0}$ to frame $F_{i,0}$ for force vectors
$\bar{\mathbf{s}}_i$	Joint axis vector
$\overline{h}_i$	Joint i type
$jcalc(\bullet)$	A function used to compute joint i transformation matrix $\mathbf{X}_J(i)$ and vector s_i
$\boldsymbol{\nu}_i, \mathbf{a}_i$	Body i frame $F_{i,0}$ velocity and acceleration expressed in global frame, respectively

$\mathbf{f}_i$	The force transmitted across joint i
$\mathbf{I}_i$	Spatial inertia matrix of body i projected on $F_{i,0}$
$\times$	Cross product in motion space
$\times^*$	Cross product in force space
$\boldsymbol{\nu}_{fb}, \mathbf{a}_{fb}$	Floating base spatial velocity and acceleration expressed in frame $F_{0,0}$
τ_i	Corresponding torque at the joint i
$\mathbf{J}_i$	Jacobian matrix for body i frame $F_{i,0}$
$\mathbf{ID}(\bullet)$	Inverse dynamics function
$\mathbf{z}_g$	Generalized z-axis selection vector
$\mathbf{J}^i_{com}$	Link i CoM Jacobian matrix
$\mathbf{r}^i_m$	Local coordinates of the i^{th} link CoM
$\boldsymbol{\tau}_{cr}$	Coriolis & centrifugal force
$\mathbf{g}_s$	Spatial vector represents the gravitational acceleration
$dim(\bullet)$	Function returns the order of an inputted square matrix
$\mathbb{S}^n$	A set of symmetric matrix with order n

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ABSTRACT . . . . . **iv**

ÖZETÇE . . . . . **vi**

ACKNOWLEDGEMENTS . . . . . **viii**

ACRONYMS . . . . . **ix**

SYMBOLS . . . . . **xi**

LIST OF TABLES . . . . . **xxxiv**

LIST OF FIGURES . . . . . **xxxv**

I INTRODUCTION **1**

1.1 Motivation 1

1.2 Previous Works 2

1.3 Problem Description 7

1.4 Proposed Solution 8

1.5 Thesis Outline 11

II SIMULATION ENVIRONMENT AND HARDWARE SETUP **13**

2.1 Simulation Environment 13

2.1.1 Human-Robot Coupled Model 13

2.2 Hardware Setup 14

2.2.1 Hardware Hierarchy 14

2.2.2 The Series Elastic Actuator (SEA) 14

2.2.3 Contact Force Sensing 15

2.2.4 FSR Signal Processing 16

2.2.5 Encoders Data Communication 17

2.2.6 Computation Module 17

2.2.7 Actuation Signal Processing 18

2.2.8 Power Source Management Module . . . 19
2.2.9 Co-Ex Mechanism . . . 19
2.3 FSR Calibration . . . 20
2.4 Joint Level Controller . . . 22
2.4.1 SEA Model . . . 22
2.4.2 SEA Impedance Controller . . . 25
2.5 Joint Level Control Verification . . . 27
2.5.1 Motor Dynamics Identification . . . 27
2.5.2 Torque Tracking . . . 32
2.5.3 Position Tracking . . . 33
2.6 Leg Mechanism Kinematics . . . 34
III CO-EX KINODYNAMICS . . . 37
3.1 Floating Base Kinematics . . . 37
3.2 Kinematics Optimization . . . 38
3.3 Kinodynamic Optimization . . . 42
3.3.1 Floating Base Dynamics . . . 42
3.3.2 Prioritized Stack of Tasks Optimization Block . . . 44
3.3.3 Contact Force Constraints . . . 46
3.4 Generation of Task Space Trajectories . . . 46
3.4.1 ZMP based trajectory . . . 47
3.4.2 Swing Foot Trajectories . . . 50
3.5 Floating Base Inverse Dynamics . . . 52
3.5.1 QR decomposition . . . 52
3.5.2 Direct Decomposition . . . 53
3.6 Experimental Verification . . . 54
3.6.1 Squat . . . 54
3.6.2 Sway . . . 56
3.6.3 Sagittal Walking . . . 58
3.6.4 Push Recovery . . . 59

IV LOCOMOTION CONTROL 69
4.1 Basic Admittance Controller 69
4.2 Centroidal Momentum Controller 70
4.3 ZMP Impedance Feedback (ZIF) Controller 71
4.4 Locomotion Controllers Simulation 72
4.4.1 Scenario 1 (SC#1): Dynamic Walking 74
4.4.2 Scenario 2 (SC#2): Perturbed Dynamic Walking 74
4.4.3 Statistical Analysis 75
4.5 Modified ZMP Impedance Feedback 77
V INERTIAL PARAMETER IDENTIFICATION AND ADAPTIVE CONTROL 80
5.1 Adaptive Locomotion Controller 80
5.2 Inertial Parameter Identification 82
5.3 Preliminaries 84
5.3.1 Rigid-Body Dynamics 84
5.3.2 Complete Physical Consistency 85
5.3.3 Identifiable Inertial Parameters 87
5.4 Filtered Regressor 88
5.5 Recursive Inertial Parameters Identification 90
5.6 Recursive Identifiability Projection Matrix (RIPM) 92
5.7 Recursive SDP Interior Point Method (SDP-IPM) 96
5.7.1 Interior Point Method 96
5.7.2 KKT system Resolution 99
5.8 Inertial Parameters Identification Algorithm Simulation Results 101
5.8.1 Forward Walking Pattern Generation 101
5.8.2 Discussion 102
5.9 Adaptive Locomotion Controller Verification 106
VI CONCLUSION 112
APPENDIX A — CO-EX KINEMATIC AND DYNAMIC MODELING . . . 115

APPENDIX B — RECURSIVE INVERSION ALGORITHMS 124

APPENDIX C — SYMMETRIC KRONECKER PRODUCT 129

APPENDIX D — DOUBLE FOUR BAR MECHANISM KINEMATIC ANALYSIS 130

REFERENCES 133

VITA 141

LIST OF TABLES

1 Right foot fitting equation coefficients $(a_{\varsigma,rf}, b_{\varsigma,rf})$ for each FSR. 21

2 Left foot fitting equation coefficients $(a_{\varsigma,lf}, b_{\varsigma,lf})$ for each FSR. 22

3 The coefficients of the linear functions (2.2-2.4) for each point pi. The parameter $\overline{pi}$ represents the mean value. 22

4 Gain values for each locomotion controller 73

5 Statistical significance among the controller performances. 73

6 Semi-Definite Programming Interior Point Method (SDP-IPM) 99

7 RMS of e_{rel} concerning 12 subjects with different walking pattern 104

8 Jacobian computation algorithm. The superscript square brackets in lines 31 and 32 indicate the column indexes, e.g. [a:b] indicates the matrix columns from a to b. 120

9 Nomenclature for NE and MNE algorithms [1]. 121

10 The conventional Newton Euler algorithm that can be used to compute inverse dynamics of a floating base robot [1]. 122

11 The MNE algorithm that is employed to compute Coriolis & centrifugal matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and the gravitation force vector $\mathbf{G}(\mathbf{q})$. 123

LIST OF FIGURES

1 The first lower limb exoskeleton developed in 1972 at Mihailo Pupin Institute. 2

2 Two active joints per leg Lower body exoskeletons. 3

3 More than two active joints per leg Lower body exoskeletons. 3

4 Self-balanced lower-body exoskeletons 6

5 Co-Ex hardware setup, its joint configuration, and its MSC.ADAMS model. 8

6 General Framework used to control the Lower Body Exoskeleton 9

7 Co-Ex joint configuration and its RaiSim simulation model. 14

8 Co-Ex hardware framework. 15

9 Co-Ex rendered CAD model with hardware component distribution, robot joints, and motion transmission methods. 16

10 Series elastic actuator (SEA) internal structure. 17

11 a) An inner view of the robot foot showing the force-sensitive resistors (FSR) fixation and distribution on the top metal plate. b) An isometric view of the robot foot after assembly. c) A detailed scheme for the metal plate bottom with installed FSRs. 18

12 Protection circuit schematic diagram. 19

13 Right foot measured data used for FSRs calibration are represented by blue markers. The obtained exponential fitted function is shown in a solid orange line. 23

14 Left foot measured data used for FSRs calibration are represented by blue markers. The obtained exponential fitted function is shown in a solid orange line. 24

15 Right foot measured vertical force regarding point *pi* is represented by blue markers. The obtained linear fitted function is shown in a solid orange line. 25

16 Right foot measured moment around x axis regarding point *pi* is represented by blue markers. The obtained linear fitted function is shown in a solid orange line. 26

17 Right foot measured moment around y axis regarding point *pi* is represented by blue markers. The obtained linear fitted function is shown in a solid orange line. 27

18 Left foot measured vertical force regarding point *pi* is represented by blue markers. The obtained linear fitted function is shown in a solid orange line. 28

19 Left foot measured moment around x axis regarding point pi is represented by blue markers. The obtained linear fitted function is shown in a solid orange line. 29

20 Left foot measured moment around y axis regarding point pi is represented by blue markers. The obtained linear fitted function is shown in a solid orange line. 30

21 SEA schematic diagram. 31

22 SEA dynamic model. 31

23 SEA impedance control loop. 32

24 The motor torque curves used for identification, in which the actual torque $\tau_{m,a}$ is shown in solid blue and estimated torque $\hat{\tau}_m$ is shown in solid orange. 33

25 Load torque tracking for different referential sinusoidal trajectories with different frequencies. 34

26 Load position tracking for different referential sinusoidal trajectories with different frequencies. 34

27 Four bar mechanism schematic diagram. 35

28 Bipedal exoskeleton Co-Ex joint configuration 39

29 The block diagram of the inverse kinematics algorithm. Forward Kinematics (FK) refers to forward kinematics. 39

30 CoM trajectory in blue line vs. ZMP trajectory in red line concerning the footprints. 48

31 x-axis CoM trajectory and feet trajectories concerning the global frame. . . 50

32 y-axis CoM trajectory and feet trajectories concerning the global frame. . . 51

33 Floating base coordinate frame vertical position response z_{fb}^{act} and the reference signal z_{fb}^{ref} 55

34 CPU computation time per sample. 56

35 The ZMP response is shown in a solid blue line and the support polygon boundaries (SP_b) are shown in a solid red line. 57

36 Squatting experiments with 2 distinct leg mechanisms: a) Mechanism $M1$ with ankle D/PF motor is located near the hip joint. b) Mechanism $M2$ with ankle D/PF motor is located on the shank. 58

37 Knee motor torque for $M1$ and M_2. 59

38 Floating base coordinate frame lateral position response y_{fb}^{act} and the reference signal y_{fb}^{ref} 60

39 Floating base roll $\theta_{x,fb}^{act}$. 61

40 The ZMP response is shown in a solid blue line, and the support polygon boundaries (SP_b) are shown in a solid red line. 62

41 The computation time of the primary and secondary threads per sample. . . 63

42 Knee joint angular displacement tracking for both dynamic and quasi-static walk. The subscripts ref and act denotes the reference and actual values, respectively. 63

43 ZMP response for both dynamic and quasi-static walk. 64

44 Force stick used to apply perturbations. 64

45 Perturbations direction: a)Lateral, and b)Forward. 65

46 Pitch axis torso angular velocity under the effect of basic PD and Orientation controller. 65

47 ZMP in x-direction under the effect of basic PD and Orientation controller. . 66

48 Knee joint angular displacement and torque response corresponding to forward pushes. The subscripts ref, act, and cmd denotes the reference, actual and command values, respectively. 66

49 Pitch axis torso angular velocity under the effect of Orientation controller. The perturbation force and impulse momentum values are denoted by F_i and I_i, respectively. 67

50 Knee joint angular displacement and torque response corresponding to lateral pushes. The subscripts ref, act, and cmd denotes the reference, actual and command values, respectively. 67

51 Roll axis torso angular velocity under the effect of Orientation controller. The perturbation force and impulse momentum values are denoted by F_i and I_i, respectively. 68

52 Block diagram of the locomotion controllers implementation. 70

53 ZMP impedance schematic 71

54 scenario 1 x_{zmp}, y_{zmp}, and ω_y response. 75

55 scenario 2 x_{zmp}, y_{zmp}, and ω_y response. 76

56 Average values of x_{zmp}^{pp}, y_{zmp}^{pp}, and ω_y^{pp} for 12 anthropomorphic subjects. . . 78

57 Relative error e_{rel} for four different simulation runs with different filter cutoff frequency γ. 102

58 Computation time t_c for four different subjects walking with different velocities over different simulation runs. 103

59 Relative error e_{rel} for four different subjects walking with different velocities over different simulation runs. 105

60 The estimated contact wrench components as projected on the floating base versus the actual measured value. 106

61 The estimated right leg joint torques versus the actual measured value. . . . 107

62 The variation in the inertia tensors eigenvalues and the triangle constraint over a single simulation run. 108

63 The variation in the estimated link masses for a single subject over six simulation runs with different velocities. 108

64 Average x-axis ZMP variations over 10 steps for five different subjects with respect to the global frame. 109

65 Average y-axis ZMP variations over 10 steps for five different subjects with respect to the global frame. 110

66 The robot joints demanded torque root mean square for each subject. 110

67 Average values of x_{zmp}^{pp}, y_{zmp}^{pp}, and τ^{rms} for 12 anthropomorphic subjects. . . 111

68 A general manipulator structure 116

69 Coordinate frames and transformations concerning the i^{th} joint. 117

70 Co-Ex leg mechanism 131

71 The angular displacements of the ankle motor θ_m, desired ankle ${}^{d}\theta_a$, and actual ankle ${}^{act}\theta_a$ for one single step. 132

CHAPTER I

INTRODUCTION

1.1 Motivation

Spinal Cord Injury (SCI) impacts 10.4 to 83.3 per million people worldwide each year [2], and a great majority of these patients lose their lives in case proper rehabilitation treatment is not provided [3]. In this connection, engineers devised several bipedal exoskeleton systems, most of which were clinically tested and are commercially available [4]. To provide walking support while satisfying weight and power consumption criteria, most of these systems provide active support solely for hip and knee Flexion/Extension (F/E) joints [5, 6]. Although very useful in its own right, the long-term use and applicability of such an approach are deemed questionable due to several issues, e.g., limited gait capability and the constant need for upper body engagement via crutches to maintain balance [7, 8].

Thus, an attractive research point regarding a self-balanced 3D dynamic walking motion with a limited number of active joints is introduced. The literature in underactuated biped robot control includes several inspiring studies; yet, a great majority of them focus on planar systems [9] or 3-D bipeds with point feet [10, 11], rendering their applicability to exoskeletons challenging. Furthermore, the controllers still have limited capability for exoskeleton control due to their reliance on an accurate model, which is hampered by inertial parameter variation. Hence, robot dynamic parameter identification could be an important topic in exoskeleton control, where the issue of physical feasibility must be addressed.

Consequently, this work is motivated in two directions. The first direction shows the development of locomotion controllers applied to an underactuated bipedal exoskeleton with an arbitrary human user. It enables the exoskeleton system to exhibit robust dynamic walking behavior despite severe parameter uncertainty and external disturbances. The second

direction concerns improving the locomotion controllers' performance against the variation of inertial parameters due to different users. To tackle this problem, the inertial parameter identification is converted to a recursive procedure with real-time applicability.

1.2 Previous Works

Various researches and commercial robot exoskeletons were directed to help patients with SCI to restore locomotion and improve their quality of life. Nearly, 70% of the individuals with SCI are bounded to manual wheelchairs [12], for long periods which leads to a set of complications, e.g. degradation of bone density [13], muscle atrophy [14], and pressure sores [15]. Furthermore, ascending/descending stairs and ramps and handling different terrains would be difficult and require special infrastructure.

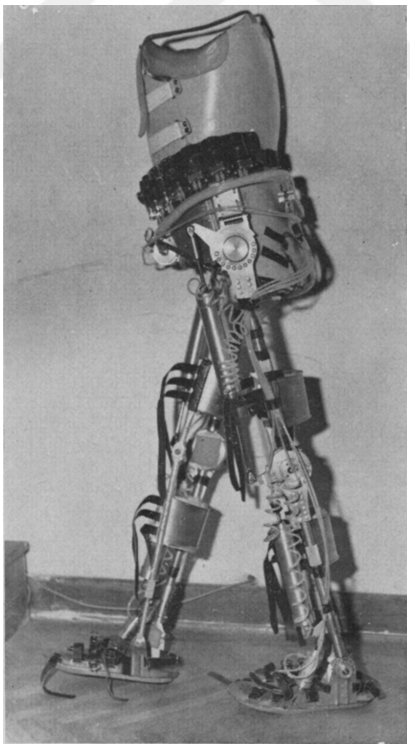

Figure 1: The first lower limb exoskeleton developed in 1972 at Mihailo Pupin Institute.

To address these issues, a set of commercial and research devices were developed. The

efforts are started in early time in 1972 by developing the world's first active lower body exoskeleton [16]; see Fig. 1. It has three active joints per leg, namely hip, knee F/E, and ankle D/PF, with a set of pneumatic pistons. It was being used solely for rehabilitation purposes. In the last decades, efforts were conducted to assist impaired populations with additional locomotion functions.

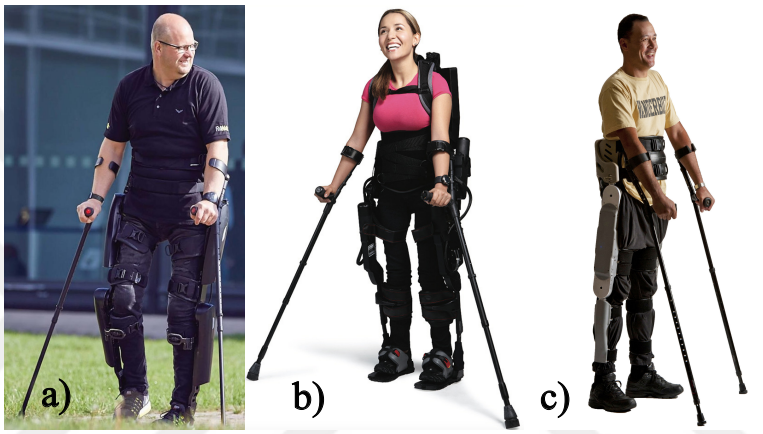

Figure 2: The exoskeletons a) Rewalk, b) Ekso, and c) Indego are shown while being used with the aid of crutches.

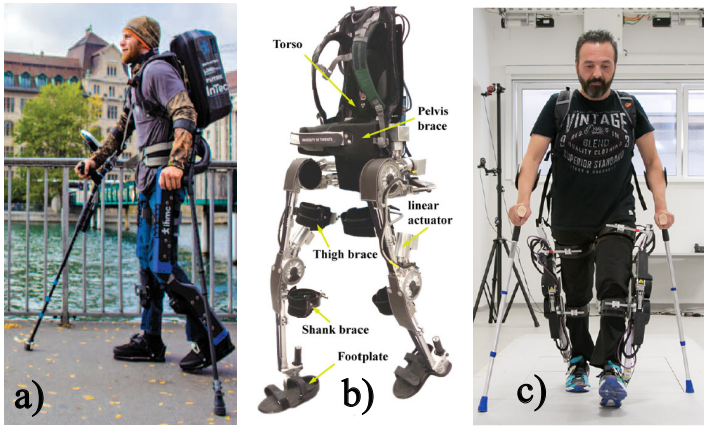

Figure 3: a) Mina v2 is shown while being used with the aid of crutches. b) MINDWALKER and c) Symbitron exoskeleton robots.

One of the first commercial devices is the assistive lower-body exoskeleton ReWalk,

which was developed by Argo Medical Technologies [17]; see Fig.2.a. It only has active joints on the hip and knee F/E axes. It enables sit-to-stand and walking motions, with a forward speed of up to 0.61 m/s. It can support a patient's weight of up to 100 kg. Its total weight is 23.3 kg. A similar device, Ekso was developed by Ekso Bionics with the same actuated joints; see Fig. 2.b. Its total weight is 23 kg, and it can support a patient up to 100 kg. The device is considered the fastest exoskeleton, with a maximum speed of 0.89 m/s [18].

Another commercially available assistive exoskeleton, namely Indego, was developed by Parker Hannifin Corporation; see Fig. 2.c. It has active hip, and knee F/E joints and passive hip Abduction/Adduction (A/A) joints equipped with springs embedded into the hip segment. The joints can be operated either using a simple PD position tracking control loop, or impedance mode controller, in which the joints act as a virtual spring-damper system. The high-level controller is implemented via a set of recorded trajectories from an able subject wearing the device and doing different activities. The device weighs 12 kg and can support a patient with a maximum weight of 113 kg while walking at an average speed of 0.24 m/s [19, 20].

Although the mentioned exoskeletons provide gait restoration, there are critical differences from their biological counterparts [21]. The able people use around 40-50% of their positive power exploiting their ankle D/PF. In response to that matter, the developed exoskeleton in [22], namely Mina v2, is equipped with three active joints per leg, namely hip F/E, knee F/E, and ankle D/PF; see Fig. 3.a. Its joints are controlled in a position control mode with a set of predesignated gaits, stairs, and ramp climbing trajectories. The robot exhibits a maximum speed of 0.29 m/s with a total exoskeleton mass of 34 kg and a supported subject of 82.5 kg [23].

In a simulation study, it was observed that human gait is quite sensitive to hip abductors [24], and any fatigue related to this joint may affect the whole leg kinematics [25]. In light of this information, it is argued that exoskeletons with active hip A/A joints could provide

relatively more comprehensive walking support. Thus, MINDWALKER is developed [26] as a step towards improved walking support; see Fig. 3.b. It is equipped with three active joints per leg, namely hip A/A, hip F/E, and knee F/E, and two passive joints, denoted hip Endo/Exo (E/E), and ankle D/PF. Its maximum walking speed is 0.8 m/s. It weighs around 30 kg and can support a patient up to 100 kg. To generate the joint trajectories it uses a quasi-static walking motion algorithm, namely Finite-State Machine based controller. It guarantees that the projection of Center of Mass (CoM) on the ground is always located within the foot support polygon.

A developed device Symbitron Exoskeleton, used four actuators per leg, namely hip A/A, hip F/E, knee F/E, and ankle D/PF [27]. In addition to two passive joints per leg namely hip E/E, and ankle I/E; see Fig. 3.c. The whole system weighs 37.8 kg and can support a patient with a complete SCI of a weight of 85kg and height of 1.8 m. It can perform a walking motion with a speed of 0.1m/s. A Series Elastic Actuators (SEA) were used to enable low-level controller impedance based. The high-level control is a neuromuscular model based which is compiled on Simulink and run in parallel to a joint level controller to generate the required torques.

To develop a crutchless self-balancing walking motion in 3D, the number of active joints per leg has to be increased. Meanwhile, an increase in the active joints would lead to high power demand due to weight increase and demanded power by the equipped motors. One of the earliest self-balancing exoskeletons is the soft XoR with a series of passive and active joints. It is developed in Computational Neuroscience Laboratories in ATR [28]; see Fig. 4.a. It has hip F/E, knee F/E, and ankle D/PF joints powered via pneumatic artificial muscles. Because of the advantages provided by pneumatic muscles, it has low friction. Its total mass is 22.2 kg, and it is able to support a subject weighing up to 70 kg. The robot was solely used for squatting, and sit-to-stand exercises. Its self-balancing was proven experimentally using the Zero Moment Point (ZMP) concept, and the variable physical stiffness

method is embodied at the ankle joints to improve the compensation of the external disturbance [29].

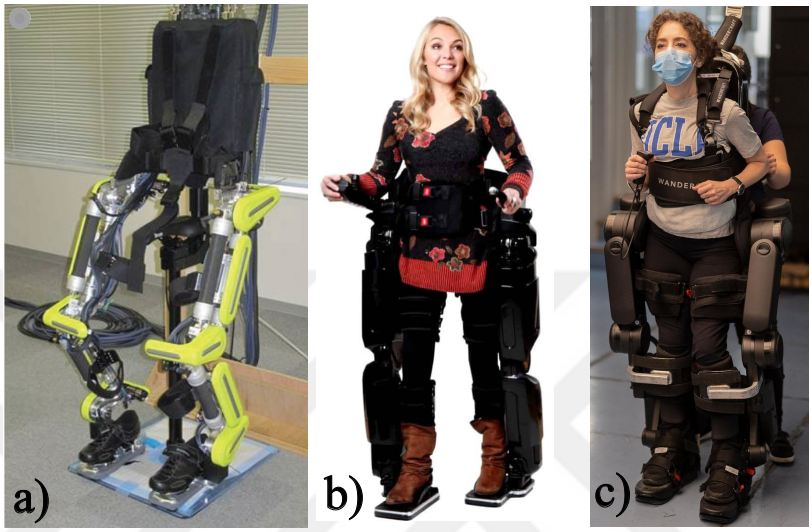

Figure 4: Self-balanced lower-body exoskeleton a) XoR, b) REX, and c) ATALANTE.

To allow hand-free walking motion the commercial device REX was developed by REX Bionics [30]; see Fig. 4.b. It is equipped with five actuators per leg, two at the hip, two at the ankle, and one at the knee. Its basic disadvantages are the large bulky size and the slow speed, which is around 0.05 m/s due to the nature of the static walking algorithm. REX can provide the power to move a combined human and exoskeleton of a total weight of 100 kg. REX solely weights 38 kg.

Another hand-free walking motion device, namely ATLANTA, is developed by Wandercraft [31]; see Fig. 4.c. Each leg is equipped with 6 actuators to provide full actuation for each leg, namely 3 DOF hip joints, two DOFs ankle joints, and a single DOF knee joint. The weight of the device is nearly 60 kg and can support a human up to 90 kg. It provides a dynamic stable gait with a forward speed of 0.15 m/s. Some simulation-based studies were able to achieve 0.4 m/s of stable walking motion [32].

The discussed exoskeletons use different methods to define joint trajectories. Several approaches use predetermined gait parameters and joint trajectories to assist users. The trajectories are either based on data obtained by able subjects or predefined Cartesian trajectories for the swing foot [22,33,34]. In the case of MINDWALKER and REX a trajectory generation approach relayed on a statically balanced gait algorithm is used [26,30]. A dynamically self-balanced gait algorithm is developed for six DoFs per leg ATALANTE exoskeleton. The control approach presented in [32, 35] used hybrid zero dynamics to stabilize periodic motions and a set of virtual constraints to generate the desired joint trajectories.

In this work, a framework is proposed that can generate a crutchless robust 3D dynamically self-balanced walking motion for four DoFs per leg exoskeleton. It relays on a task prioritization-based optimization algorithm for inverse kinematics resolution and the ZMP concept for walking pattern generation. These methods were used due to their popular usage in the field of legged robots [36–39]. The framework is enforced with a set of locomotion controllers and recursive inertial parameter identification algorithms to improve its robustness against underactuation, external disturbances, and inertial parameter uncertainty. It is applied to A built *in-house* 55.3 kg lower-body exoskeleton, namely Co-Ex, that is developed in the Biomechatronics Lab at Ozyegin University [40]. The Co-Ex is shown in Fig. 5.a and .b. It uses four active joints per leg, namely hip A/A, hip F/E, knee F/E, and ankle D/PF, as illustrated in Fig. 5.c.

1.3 Problem Description

The main challenges in using this 4-DoF per leg joint configuration are the underactuation in addition to high inertial parameters uncertainty. Conventional gait generation techniques may not be directly applicable to this case since the number of constraints exceeds the number of active robot joints. The whole-body control approaches exhibited enhanced balancing and walking behaviors for fully-actuated legged robots [41, 42]. Nevertheless, replicating such locomotion capabilities in underactuated systems is quite challenging. The

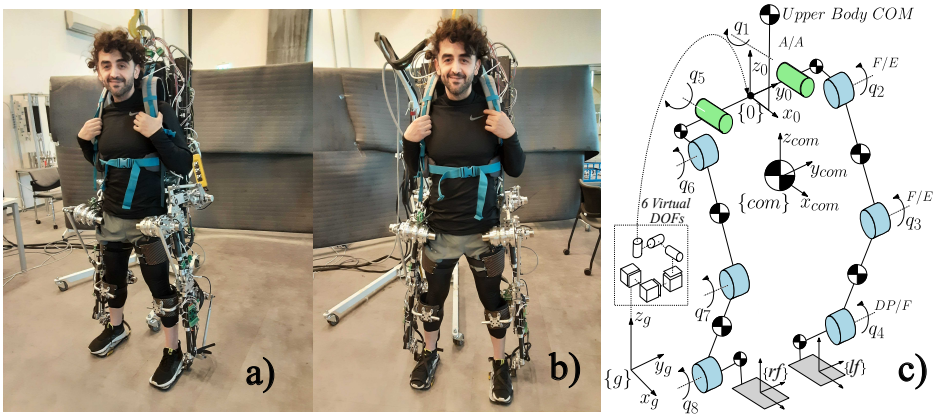

Figure 5: Co-Ex being worn by the able body is shown in a) and b) from the side and front views, respectively. c) Joint configuration of the bipedal exoskeleton Co-Ex.

floating base of a legged robot can move in 3-D space with any orientation when the robot leg has 6 active joints or more and when there are sufficient contact forces [43]. Likewise, contact force optimization is possible by manipulating the floating base in different orientations [44,45]. In the case of underactuation, the joint space does not have enough variables to fully satisfy the task space constraints, and thus, the solution space is much more limited.

To mitigate the effect of the underactuation locomotion controllers can be exploited, e.g Centroidal Momentum Controller, Admittance Controller [42,46]. The application of such controllers to exoskeletons is still challenging. While being useful in their own right, the controllers are still of limited capability for exoskeleton control due to their dependence on an accurate model, which is hindered by the uncertainty of the inertial parameters. Hence, a real-time inertial parameter identification algorithm is required to be developed to improve the locomotion controller's adaptation to the inertial parameter variation.

1.4 Proposed Solution

Fig. 6 shows the basic block diagram for the proposed solution. The blue lines show the basic blocks used to control Co-Ex in a simulation environment. The joint level controller block with a green line is added to control SEA in the Co-Ex hardware setup. The inertial

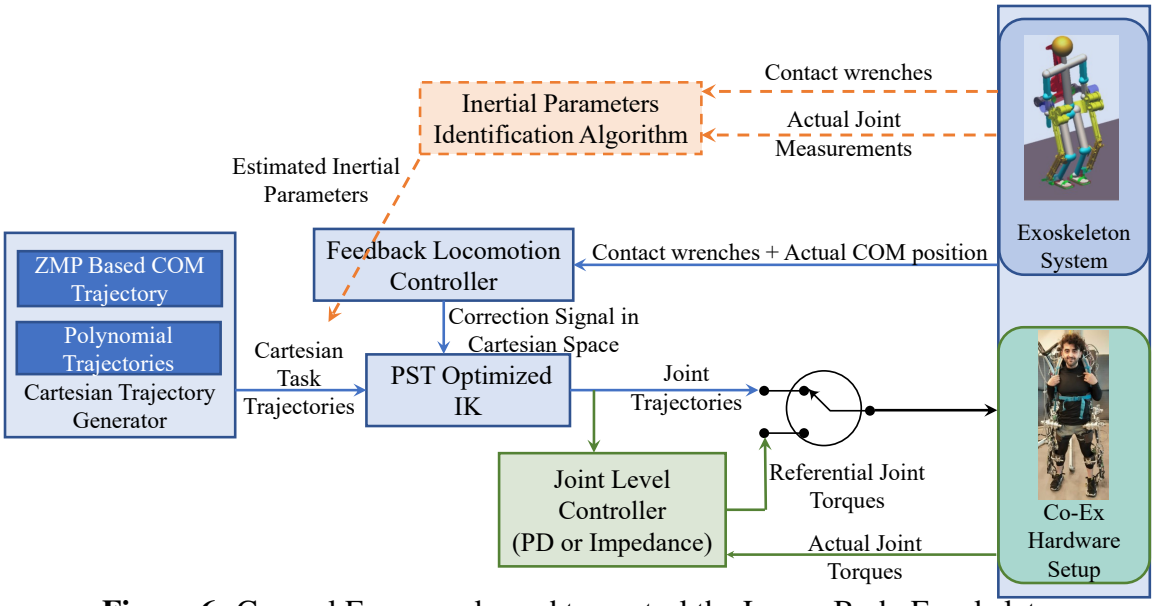

Figure 6: General Framework used to control the Lower Body Exoskeleton

parameters identification block was added with orange dashed lines to improve the performance of the locomotion controllers by estimating a feasible set of inertial parameters, due to their dependency on the exoskeleton system dynamic model.

Cartesian Trajectory Generation The proposed bipedal exoskeleton can be considered a manipulator with three end effectors represented by two feet and a Center of Mass (CoM). The two feet trajectories for walking motion can be handled using a set of polynomial trajectories to control each foot clearance, and the required forward proceeding during walking, in addition to providing a smooth transition between different phases. The regulation of CoM is an essential part of the proposed framework for locomotion balance during walking [37]. A Zero Moment Point (ZMP) based algorithm is employed to generate a balanced walking motion pattern. It requires minimal knowledge about robot dynamics while solving a simple inverted pendulum dynamic model to generate a CoM trajectory. In this framework, the work presented in [39] is utilized to solve the ZMP equations analytically in both x and y directions, to guarantee the trajectories continuity in acceleration level.

Inverse Kinematics (IK) Optimization Inspired by the efficacy of task prioritization methods [47–51], an optimization algorithm is proposed that employs a prioritized stack

of tasks (PST) to satisfy the constraints hierarchically. The aim is to achieve dynamically equilibrated 3D walking using a bipedal exoskeleton in an underactuated joint configuration, e.g., 4 active joints per leg. Since achieving the exact predesignated trajectories is not possible due to underactuation, the algorithm makes use of a prioritized stack of tasks approach to sort the CoM (Center of Mass), and left and right foot task variables based on the contact phase.

Feedback Locomotion Controller To improve the robustness of the proposed framework against parameter variation and external disturbances, two locomotion controllers were utilized, namely centroidal momentum controller (CMC), and basic admittance controller (BAC), in addition to a proposed ZMP impedance feedback (ZIF) in [52].
In CMC, the gathered contact wrenches are projected onto centroidal dynamics using spatial transformation matrices and then mapped to the referential joint trajectories using centroidal momentum matrix (CMM) [42]. CMC allows the inclusion of the upper body torso, and arms in the case of humanoids, in maintaining the whole locomotion balance through synchronization and responsive motions to external disturbances.

For BAC, a virtual spring-damper model is designed to provide compliant behavior concerning the contact forces at the transition instants [46]. The contact forces are measured and compared to the corresponding referential signals. Afterward, the error is injected into a spring-damper model, and the obtained displacements are passed to the PST Optimized IK block in the form of modified Cartesian trajectories.

The ZIF controller can be regarded as a modified version of BAC in which it places the virtual spring-damper model between the support polygon boundaries and the ZMP. The variation of the actual measured ZMP is injected into the virtual spring-damper model and mapped to the centroidal dynamics via spatial transformation and a full-body dynamic model. A detailed description with a full mathematical proof of the locomotion controllers is provided in chapter III.

Inertial Parameters Identification Algorithm The applicability of locomotion controllers to the lower body exoskeletons is still of limited capability due to their dependence on an accurate model, as it is hindered by the uncertainty in inertial parameter variation. Hence, robot dynamic parameter identification could be an important topic in exoskeleton control, where the issue of physical feasibility must be addressed. To tackle this problem, the inertial parameters identification is converted to a recursive procedure that can estimate the inertial parameters recursively with real-time applicability. In the proposed solution, the inertial parameters are identified in a complete physical consistency form [53], in which the well-known XZ+ZX semidefinite programming algorithm is modified to be solved recursively [54]. To mitigate the effect of noise and contact transitions, and carry out the identification procedure without the acceleration signal, the filtered regressor is used [55].

Joint Level Controller In the case of an assistive exoskeleton, the robot joints are supposed to manage the motion without expecting any actuation from its pilot. Two control modes are provided to control SEA. The first control mode is a simple PD controller to control the angular displacement. The second control mode is an impedance controller using cascaded PID [56]. A torque control loop is used to achieve the accurate torque tracking that is necessary for the feed-forward torque inputs. Moreover, the impedance control renders active compliance to mitigate the effects of contact wrenches or uncertain variations in contact transitions.

1.5 Thesis Outline

The thesis chapters are distributed as follows: **The second chapter** explains the simulation model and the dynamic environment. It discusses in detail the Co-Ex hardware setup and its different components. It shows a complete description of the used sensors, actuators, conditioning circuits, and controllers. Furthermore, the structure of the hardware and the interconnection between its components is explained. **The third chapter** discusses the

kinodynamics resolution of Co-Ex and the method of generating the Cartesian task trajectories. Three different locomotion controllers were explained in **the fourth chapter**. Moreover, a set of modifications were carried out on the proposed controllers to improve their practical applicability and to enable the inclusion of the estimated inertial parameter depicted in **the fifth chapter**. **The sixth chapter** describes the proposed complete physical consistency inertial parameters identification algorithm with its recursive features. The chapter also discusses the semidefinite programming algorithm and newly added modifications to enable the recursion process with its online applicability. Moreover, It shows the adaptive locomotion controller against inertial parameter uncertainty. **Chapter seven** discusses the conclusions and future works.

CHAPTER II

SIMULATION ENVIRONMENT AND HARDWARE SETUP

2.1 Simulation Environment

To verify the feasibility of the proposed algorithms, a series of dynamic walking simulations were conducted using a 3D dynamic RaiSim simulator [57] running on an operating system (64-bit Ubuntu 18.04); see Fig. 7.a. MSC ADAMS was used with the operating system Windows 10; see Fig. 7.b. These simulations were executed using a PC with an Intel (R) Xeon (R) W-2155 processor (3.3 GHz) and 32.0 GB of RAM.

2.1.1 Human-Robot Coupled Model

A 40 kg bipedal exoskeleton robot model was coupled to 12 distinct anthropomorphic user models ($66 \sim 102$ kg and $170 \sim 190$ cm). The coupled models were used for different walking simulation runs. The coupled model consists of a human and a bipedal exoskeleton model that are mechanically tied together at the torso, lower leg, and foot via cuffs. The model is constructed on the RaiSim and MSC ADAMS simulators; see Fig. 7. The bipedal exoskeleton model was built based on an actual built *in-house* exoskeleton Co-Ex [40], with joint configuration given in Fig. 5.c. Each leg has 4 DoFS, denoted as, 2 DoF hip joints along the Abduction/Adduction (A/A) and Flexion/Extension (F/E) axes, 1 DoF knee joint along the F/E axis, and 1 DoF ankle joint along the Dorsi-plantar/Flexion (DP/F) axis.

The human model possesses 26 DoFs (8 DoF leg, 5 DoF arm) with non-motorized joints as the target population is spinal cord injury patients. A certain amount of rotational damping is introduced to the joints to prevent unexpected excitations. For a thorough analysis, 12 anthropomorphic subjects with different inertial parameters were built using [58] and [59]. The parameters of human subjects are unknown to the controllers and identification algorithm.

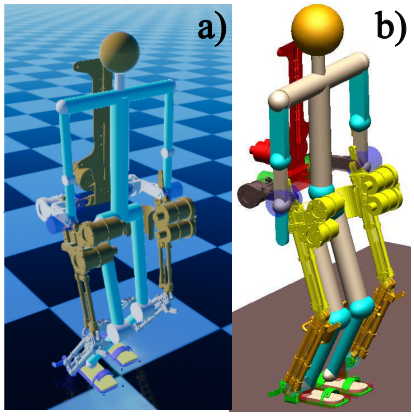

Figure 7: Left: RaiSim simulation model. Right: MSC ADAMS simulation model.

2.2 Hardware Setup

To provide an experimental a built *in-house* Co-Ex robot is used.

2.2.1 Hardware Hierarchy

The Co-Ex hardware structure contains a wide set of components that serves the actuation, sensory feedback, and computation tasks. Fig. 8 shows the hardware framework and how each component functions over the proposed structure. The distribution of the components over the Co-Ex robot mechanism is depicted in Fig. 9.

2.2.2 The Series Elastic Actuator (SEA)

Co-Ex is equipped with eight series elastic actuators (SEA), used for robot active joints. Each actuator provides 44.09 rpm as maximum angular speed and 96 Nm as the continuous stall torque. Each motor weighs 2.87 kg and has an outer radius of 53.5 mm and a total length of 108 mm. The SEA components are arranged as shown in Fig. 10.

- **Absolute Encoders**: Two absolute encoders (Broadcom Avago AS38H39ES135) are equipped to measure the spring deflection and motor angular displacement. Each encoder uses 23 bits of resolution to render a torque measuring resolution of 3.9 mNm.

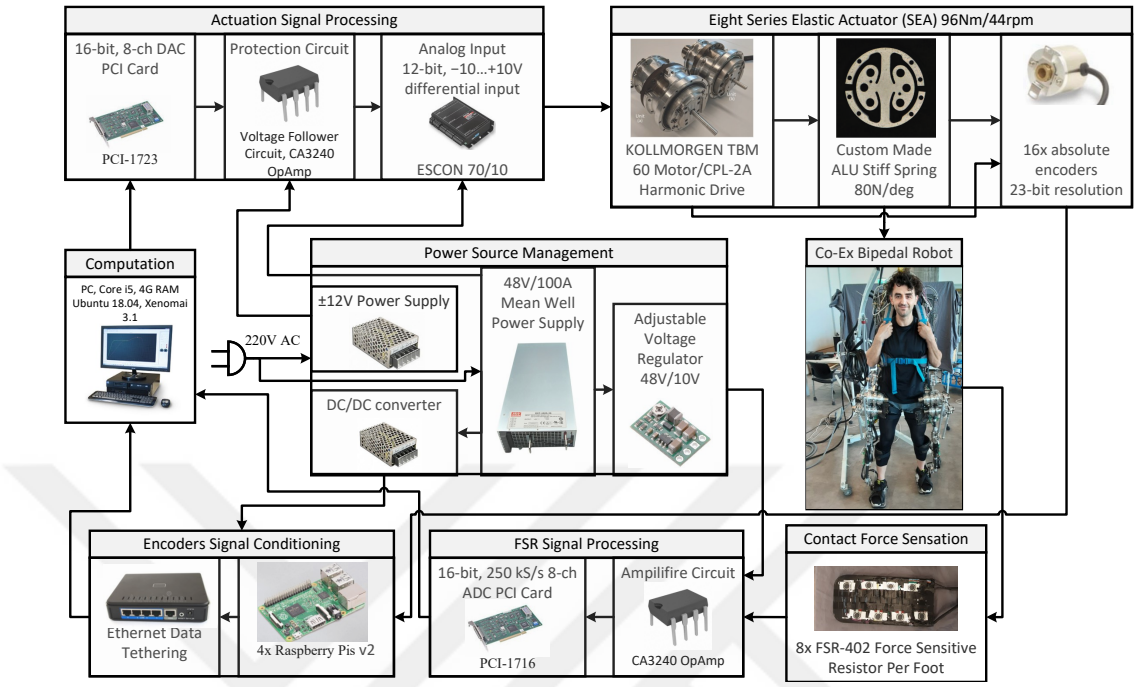

Figure 8: Co-Ex hardware framework.

- **Torsional Spring**: To fulfill the lightweight constraint, the SEA spring is manufactured from Al-7075 aluminum alloy with a life of 5.10^8 cycles. The spring can provide a load torque of 180 Nm and a stiffness value of 80 N/deg [60].

- **Gear Box**: Ultra low backlash gearbox (Harmonic Drive CSG-25, 1:100 ratio) is used.

- **Brush-less Motor**: Frame-less brush-less DC motor (Kollmorgen TBMs-7615) with maximum angular velocity of 4025 rpm, and stall torque of 0.996 Nm is equipped.

2.2.3 Contact Force Sensing

Each robot foot chassis consists of an aluminum plate with square perturbations that are used to fix an array of eight Force Sensitive Resistors (FSR) as shown in Fig. 11. Each FSR is used to measure the vertical force at a specific horizontal distance from the foot's sole center. A transparent filament is added to protect the sensor from scratches and shear

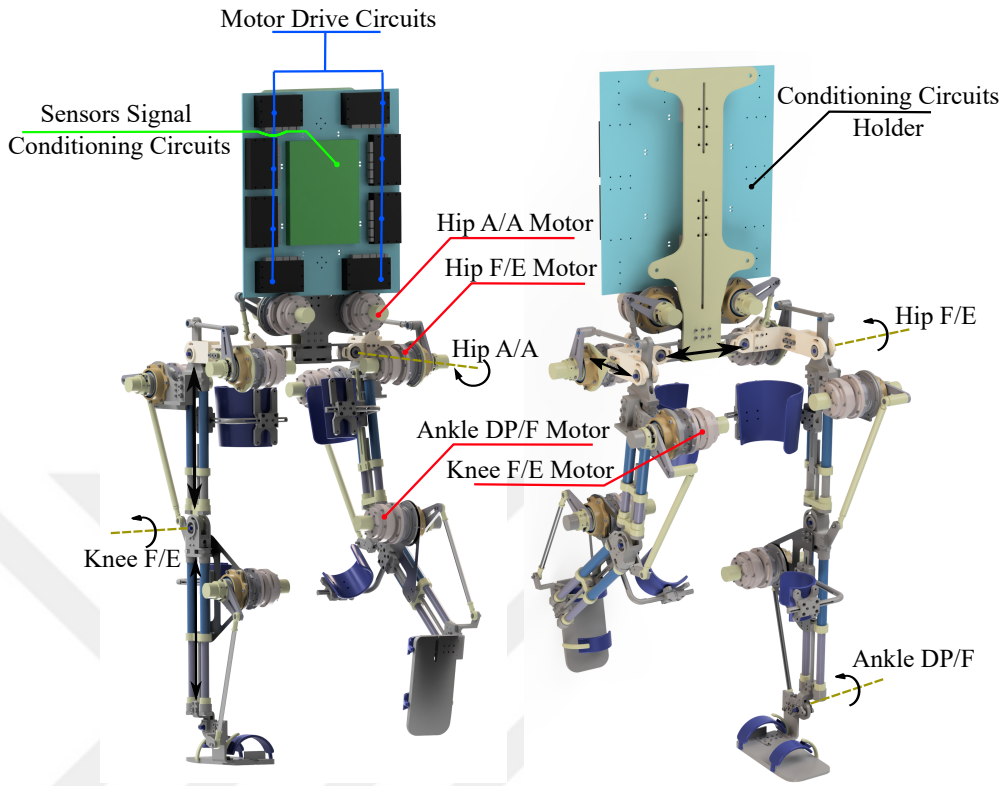

Figure 9: Co-Ex rendered CAD model with hardware component distribution, robot joints, and motion transmission methods.

stresses. 3D-printed load concentrators are stuck on top of the filament to concentrate the load on the sensor's active area. Black taps were used to fix the cables along specific routes.

2.2.4 FSR Signal Processing

The variation in each FSR resistance is converted to an equivalent voltage value using a custom-made amplifier circuit based on an OpAmp integrated circuit with commercial number CA3240. The amplified voltage signal is passed to the PC via an Advantech PCI card with commercial number PCI-1716. The card can convert the analog signal to a digital signal with a 16-bit resolution. The transmission rate capability is up to 250 KS/s. The input voltage range is $\pm 5\ V$.

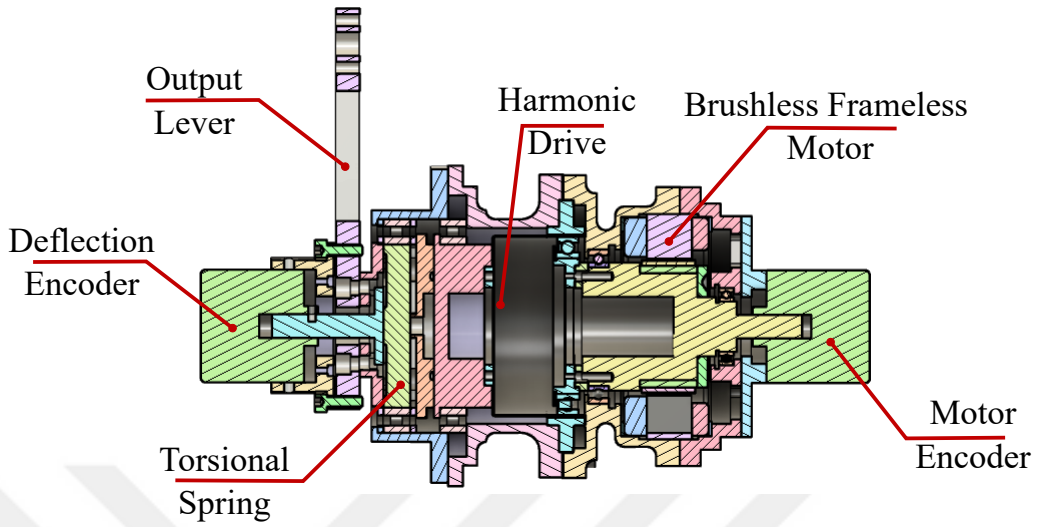

Figure 10: Series elastic actuator (SEA) internal structure.

2.2.5 Encoders Data Communication

The encoder's signals are processed via four Raspberry Pi cards (V2). They are used to collect the encoders' signals and convert them to equivalent angular displacements and velocities, then the data are transmitted to the PC via an Ethernet router at a 2kHz rate.

2.2.6 Computation Module

A Core i5-3470 desktop PC with 4 GB RAM and a 250 GB SSD hard disk drive is used to process both low-level joint control signals and high-level task space control signals. The PC is operated via Ubuntu 18.04 64-bit and equipped with Xenomai 3.1 for a robust and efficient real-time interface. The codes were written in C using the Clion IDE [61]. The Gnu scientific library is utilized for solving linear algebra operations [62]. The provided software/hardware setup allowed for a fast communication rate with a 0.5 *ms* sampling rate. To solve the task space control and joint level control problems simultaneously, multithreading functions were used. It enables parallel computation with different sampling rates. The joint level control loop is solved in 0.5 *ms* while the task space control loop is solved in 10 *ms*.

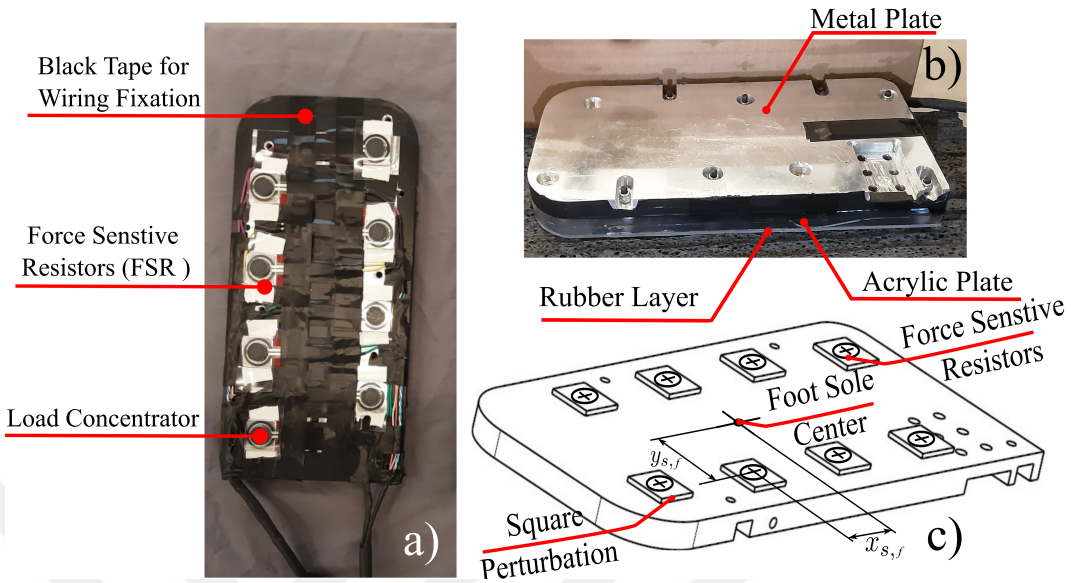

Figure 11: a) An inner view of the robot foot showing the force-sensitive resistors (FSR) fixation and distribution on the top metal plate. b) An isometric view of the robot foot after assembly. c) A detailed scheme for the metal plate bottom with installed FSRs.

2.2.7 Actuation Signal Processing

To process the actuation signals based on the control action signals generated by the PC CPU, an interface card PCI-1723 is used. The card is connected to the PC via a PCI port. It provides eight analog output channels with 16-bit resolution and a voltage range of $\pm 10\,V$. The DAC card is capable of providing 5 mA with the maximum output impedance of 0.1 Ω. Due to the sensitivity of DAC card components to electric variations, a voltage-splitting circuit using OpAmps is designed and manufactured, which acts as buffer components. A dual bipolar OpAmp IC (CA3240) is used with a maximum input/output current of 50 pA/12 mA. The circuit schematic diagram for a single OpAmp is shown in Fig. 12. It provides a 1:1 input-to-output voltage ratio. Afterward, the control signals are processed via the brushless motor drive circuit ESCON 70/10 using its differential analog input terminals. Its internal impedance is 100 $k\Omega$ with a resolution of 5.64 mV. The received input control voltage is mapped to the motor current control signal with a 1:1 ratio and sent to the motor in the form of a current signal using an auto-tuned current control loop.

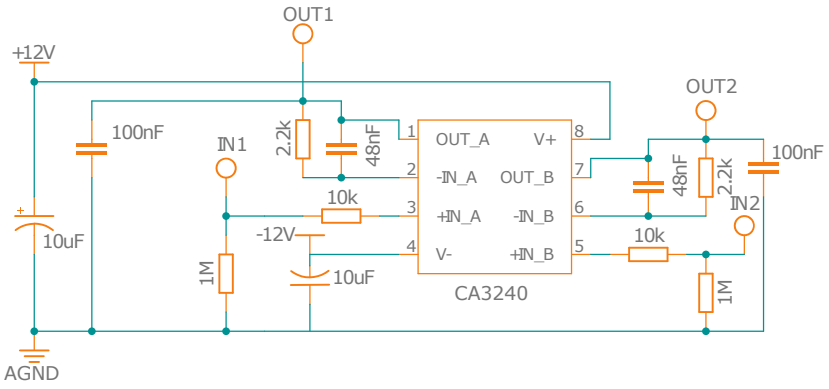

Figure 12: Protection circuit schematic diagram.

2.2.8 Power Source Management Module

The module is used to supply the required power for different hardware components at specific voltage levels. The module is supplied by a 220 V AC voltage source. A MEAN WELL (RST-5000-48) power supply is used, which is supplied by 220 VAC, generating 100 $A/48\ VDC$. A DC/DC converter (SD-25C-5) is used to step down the voltage from 48 V to 5 V to power the encoder's signal conditioning module. An adjustable step-down voltage regulator circuitry (D36V6AHV) is used to supply the FSR amplifier circuit with 10 V from 48 V. A 220 V AC is used to supply power supplies (MT-24-12) connected in series to provide $\pm 12\ V$ to energize the DAC protection circuit.

2.2.9 Co-Ex Mechanism

The Co-Ex mechanism is shown in Fig. 9. It shows the distribution of the hardware components over the robot. The set of conditioning circuits is fixed on the robot's back plate shown in cyan color. The plate is made of acrylic and enforced with an aluminum bar. The shown black boxes represent the motor drive circuits (ESCON 70/10), and the green box represents the encoder's signal conditioning module. To improve robot dynamic features in terms of mass distribution, and compactness, and achieve ergonomic usage the SEA are distributed on the robot links as shown in Fig. 9. Meanwhile, a set of four bar mechanisms is used to provide the required motion transmission from the motors' output shafts to the

robot joints.

The human body is supposed to be attached to the exoskeleton structure via cuffs, which are shown in dark blue. As the robot is supposed to be worn by different users with distinct anthropomorphic structures, the robot links are designed to be re-sizable to fit different users. Moreover, the cuffs are movable with respect to the Co-Ex links to achieve the optimum ergonomic human body attachment. Technically speaking, a resizable robot structure would allow a better alignment between the robot and its contained human joints.

2.3 FSR Calibration

An array of FSR sensors is used to measure the contact vertical force and horizontal moments for each foot. Each array contains eight sensors distributed in a deterministic location from the foot sole center. Each FSR shows an electrical resistance variation equivalent to the applied vertical force. The variation in sensor resistance is converted to a corresponding change in voltage measured via PC using the FSR signal processing module; see Fig. 8. The calibration procedure is executed in two stages. In the first stage, each sensor is calibrated distinctly using the measured voltage via PC and a set of known applied forces that are applied using a clamp and precisely calibrated load cell (HEX 6-axis force/torque sensor-OnRobot). The obtained measurements and the fitted curves are shown in Fig. 13, and 14, for the right and left foot, respectively. The blue markers show the measured data, and the solid orange line shows the fitted curve. An exponential function

$$f_{\varsigma,f} = a_{\varsigma,f} e^{(b_{\varsigma,f} v_{\varsigma,f})} \tag{2.1}$$

is used to represent the fitted curve functions. The parameters $f_{\varsigma,f}$ and $v_{\varsigma,f}$ represent each sensor force and voltage signal, respectively. The symbols $a_{\varsigma,f}$ and $b_{\varsigma,f}$ are the fitted curve coefficients. The subscript $f \in \{r,l\}$ denotes either right foot rf or left foot lf. The subscript ς is the sensor index. The coefficients $a_{\varsigma,f}$, and $b_{\varsigma,f}$ are tuned using MATLAB $fit(\bullet)$ function based on the measured data. The coefficients for each sensor ς are given in table 1, and 2.

In the second calibration stage, the foot is enclosed with the acrylic plate, as shown in Fig. 11.b. Deterministic force values are applied to the bottom of the robot's feet using a set of dead weights with known values at four specified points. The applied forces are acquired, and the output force of each sensor is obtained using (2.1). Linear functions are used to represent the relation between the aggregated forces measured by the FSRs and the applied forces as follows:

$$f_{f,z}^{pi} = a_{z,f}^{pi} \sum_{\varsigma=1}^{8} f_{\varsigma,f} + b_{z,f}^{pi}, \tag{2.2}$$

$$m_{f,x}^{pi} = a_{x,f}^{pi} \sum_{\varsigma=1}^{8} f_{\varsigma,f} y_{\varsigma,f} + b_{x,f}^{pi}, \tag{2.3}$$

$$m_{f,y}^{pi} = a_{y,f}^{pi} \sum_{\varsigma=1}^{8} f_{\varsigma,f} x_{\varsigma,f} + b_{y,f}^{pi}, \tag{2.4}$$

The vertical force at each point is denoted by $f_{c,z}^{pi}$ and their moments around the foot sole center in x and y directions are symbolized by $m_{c,x}^{pi}$, and $m_{c,y}^{pi}$, respectively. The symbols $x_{\varsigma,f}$ and $y_{\varsigma,f}$ are forward and lateral distances between each sensor and the foot sole center, as shown in Fig. 11.c. The superscript pi is the point index. The obtained results for calibrating equations (2.2-2.4) for each foot with four different points per foot are shown in Figs. 15 to 20. The obtained coefficients of linear functions given by equations (2.2-2.4) for each foot are shown in tables 3.

To find the final equations that represent the relation between aggregated sensors forces signals and each foot contact force, the coefficients average of each four points per foot are

Table 1: Right foot fitting equation coefficients $(a_{\varsigma,rf}, b_{\varsigma,rf})$ for each FSR.

Foot	Right Foot							
Sensor (ς)	1	2	3	4	5	6	7	8
$a_{\varsigma,rf}$	1.87	4.16	20.65	3.49	5.85	7.16	9.7	11.59
$b_{\varsigma,rf}$	1.04	0.94	0.57	0.93	0.92	0.7	0.76	0.66

Table 2: Left foot fitting equation coefficients $(a_{\varsigma,lf}, b_{\varsigma,lf})$ for each FSR.

Foot	Left Foot							
Sensor (ς)	1	2	3	4	5	6	7	8
$a_{\varsigma,lf}$	2.75	11.82	5.12	5.76	13.58	9.12	9.08	9.48
$b_{\varsigma,lf}$	1.01	0.63	0.81	0.76	0.61	0.68	0.64	0.64

Table 3: The coefficients of the linear functions (2.2-2.4) for each point pi. The parameter $\overline{pi}$ represents the mean value.

Foot	Right Foot					Left Foot				
Point(pi)	1	2	3	4	$\overline{pi}$	1	2	3	4	$\overline{pi}$
$a^{pi}_{z,f}$	1.90	1.48	1.35	2.62	1.84	2.15	2.12	1.70	1.99	1.99
$b^{pi}_{z,f}$	-7.38	-7.50	-10.1	-9.80	-8.7	-19.1	-13.2	-7.85	-11.4	-12.9
$a^{pi}_{x,f}$	1.90	1.54	1.49	2.62	1.89	2.15	3.47	2.11	2.49	2.55
$b^{pi}_{x,f}$	-0.26	-1.03	-0.24	-0.34	-0.47	0.67	0.58	-0.58	0.58	0.31
$a^{pi}_{y,f}$	2.18	1.43	1.93	3.78	2.33	2.11	2.76	1.90	2.01	2.11
$b^{pi}_{y,f}$	0.35	0.26	0.84	-0.61	0.21	0.49	-0.08	-0.37	0.32	0.09

used as follows:

$$f_{f,z} = \overline{a}_{z,f} \sum_{\varsigma=1}^{8} f_{\varsigma,f} + \overline{b}_{z,f}, \tag{2.5}$$

$$m_{f,x} = \overline{a}_{x,f} \sum_{\varsigma=1}^{8} f_{\varsigma,f} y_{\varsigma,f} + \overline{b}_{x,f}, \tag{2.6}$$

$$m_{f,y} = \overline{a}_{y,f} \sum_{\varsigma=1}^{8} f_{\varsigma,f} x_{\varsigma,f} + \overline{b}_{y,f}, \tag{2.7}$$

where, $\overline{a}_{z,f} = 0.25\sum_{pi=1}^{4} a^{pi}_{z,f}$, $\overline{a}_{x,f} = 0.25\sum_{pi=1}^{4} a^{pi}_{x,f}$, $\overline{a}_{y,f} = 0.25\sum_{pi=1}^{4} a^{pi}_{y,f}$, $\overline{b}_{z,f} = 0.25\sum_{pi=1}^{4} a^{pi}_{z,f}$, $\overline{b}_{x,f} = 0.25\sum_{pi=1}^{4} a^{pi}_{x,f}$ and $\overline{b}_{y,f} = 0.25\sum_{pi=1}^{4} a^{pi}_{y,f}$. The average coefficients are given by the 6^{th} and 11^{th} columns in table 3.

2.4 Joint Level Controller

2.4.1 SEA Model

Each active joint in the Co-Ex is actuated via SEA. It is equipped due to its high fidelity in torque handling. Moreover, it is equipped with a stiff spring at its output shaft that reduces the coupling effect, provides accurate torque measurement due to the linearity between the

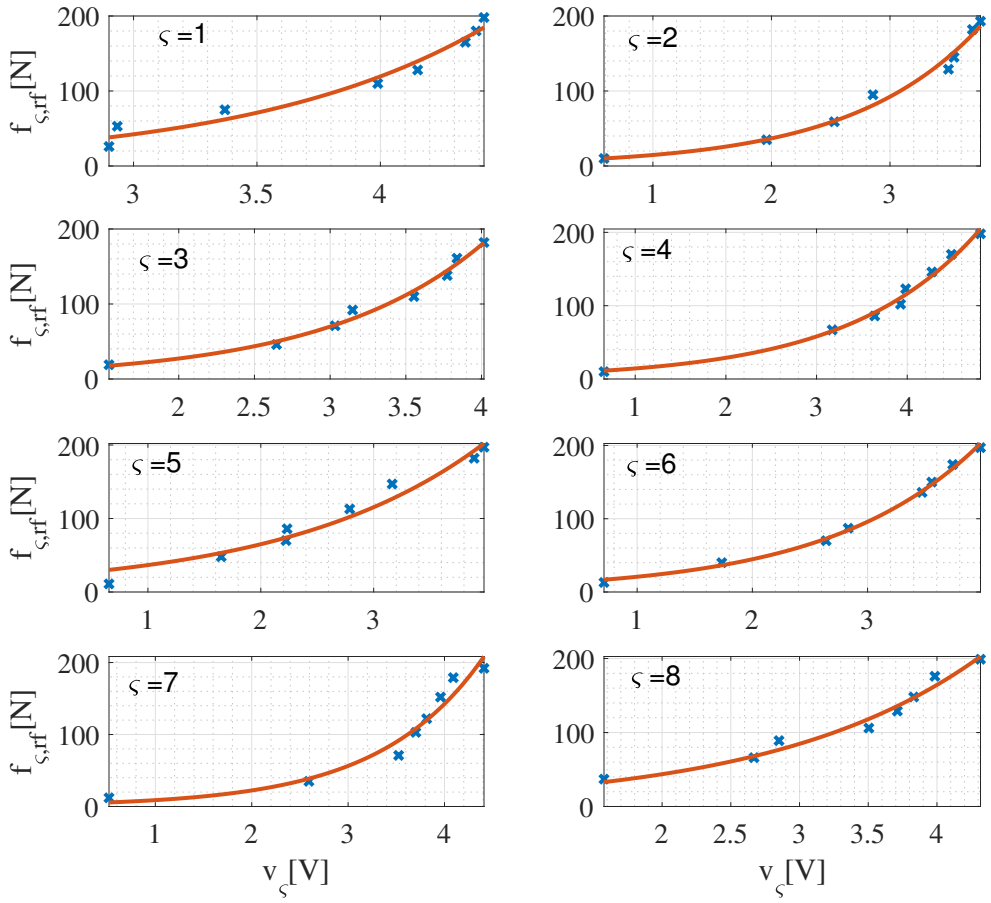

Figure 13: Right foot measured data used for FSRs calibration are represented by blue markers. The obtained exponential fitted function is shown in a solid orange line.

spring deflection and loaded torque, and improves the yielded torque by the motor through the exploitation of a torque control loop. The SEA schematic is shown in Fig. 21, and the dynamic model block diagram is represented by Fig. 22. It shows the motor model as

$$P_m(s) = \frac{\theta_m(s)}{\tau_m - \frac{K_s}{N_G}\theta_s - \frac{d_m}{N_G}} = \frac{1}{J_m s^2 + B_m s}, \tag{2.8}$$

and load motor as

$$P_l(s) = \frac{\theta_l}{K_s\theta_s - d_l} = \frac{1}{J_l s^2 + B_l s}, \tag{2.9}$$

respectively. The symbols (J_m, B_m), (J_l, B_l), and (θ_m, θ_l) are the motor and load side inertia, viscous friction, and angular displacement, respectively. The angular displacement, θ_s

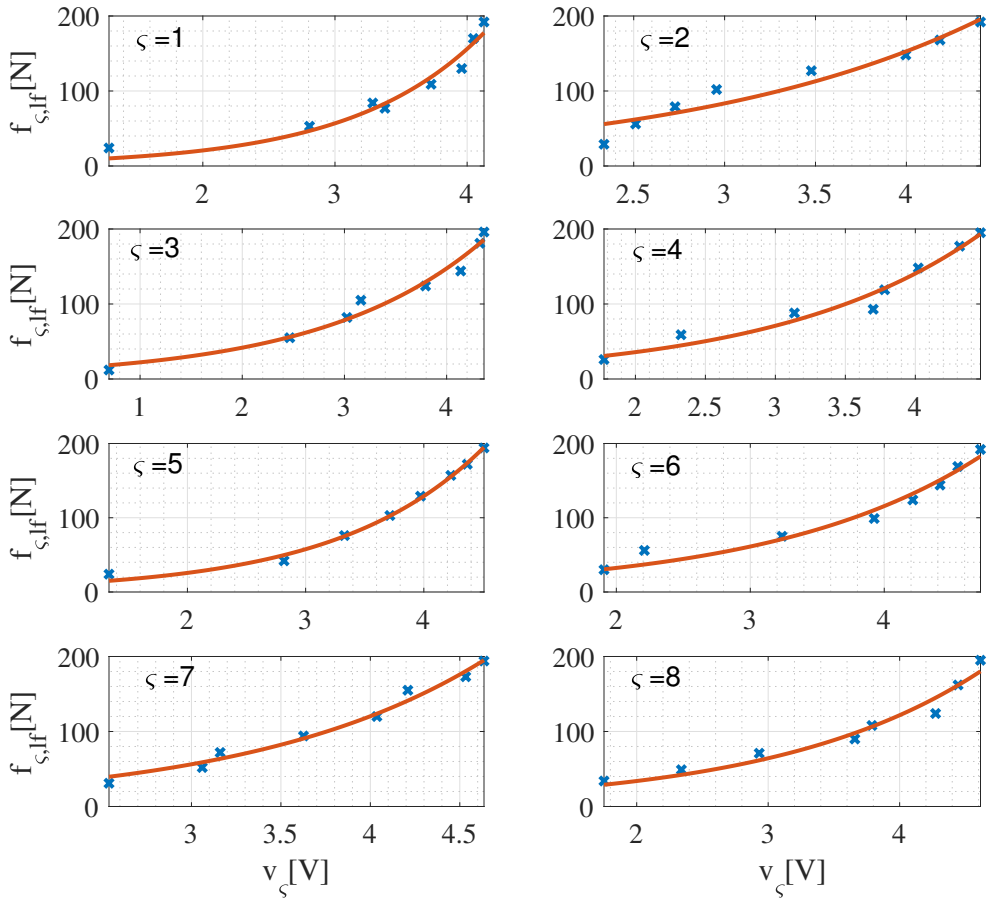

Figure 14: Left foot measured data used for FSRs calibration are represented by blue markers. The obtained exponential fitted function is shown in a solid orange line.

represents the spring deflection. The gear ratio is symbolized by N_G, the Laplace variable is given by s, and the spring stiffness is denoted by K_s. The load and motor sides disturbances are denoted by d_l, and d_m, respectively. The spring deflection can be computed as:

$$\theta_s = \theta_m N_G^{-1} - \theta_l. \tag{2.10}$$

Using equations (2.8-2.10), a relation between the motor torque and spring deflection is given as follows:

$$\theta_s = \tau_m \frac{P_m(s) N_G^{-1}}{1 + K_s P_l(s) + K_s P_m(s) N_G^{-2}} \tag{2.11}$$

where, the disturbances d_l, d_m are assumed to be zeros. Using equations (2.8), (2.9), and (2.11) a simplified relation between spring torque τ_s as an output and motor torque τ_m as

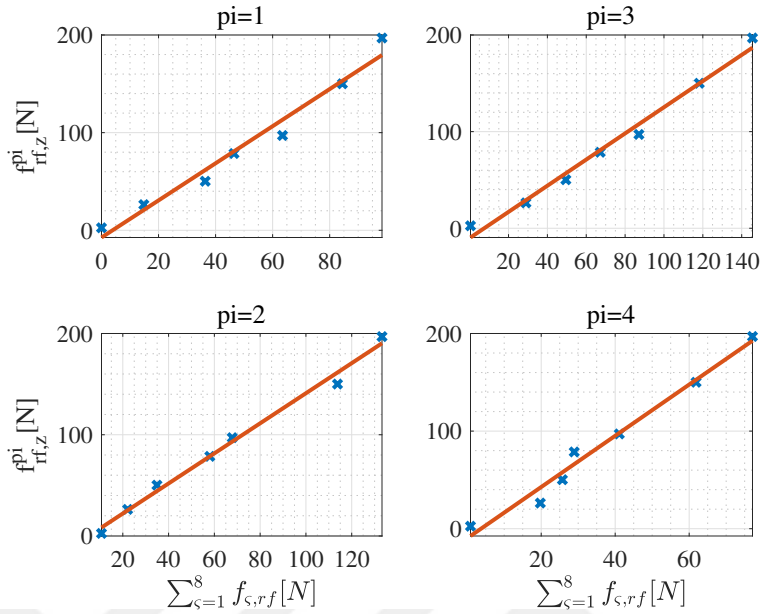

Figure 15: Right foot measured vertical force regarding point *pi* is represented by blue markers. The obtained linear fitted function is shown in a solid orange line.

an input is obtained as follows:

$$P_n(s) = \frac{\tau_s}{\tau_m} = \frac{K_s N_G^{-1}(J_l s + B_l)}{J_l J_m s^3 + (J_m B_l + J_l B_m)s^2 + (B_m B_l + K_s(J_m + J_l N_G^{-2}))s + K_s(B_m + B_l N_G^{-2})}. \tag{2.12}$$

2.4.2 SEA Impedance Controller

To provide a proper controller, a simple PD controller for angular position tracking or an impedance controller with a torque control loop was used; see Fig. 23. The second choice is favorable due to guaranteeing torque tracking as well as providing compliance, which allows the joints to behave as a spring-damper system.

A comprehensive study proposed by Ugurlu et. al. [56] shows a set of controllers applied to the SEA model. The study showed a set of efficient controllers that can be used with similar motor configurations. A cascaded PID controller was able to handle torque tracking with a frequency of up to 16.7 Hz and an amplitude of 20 Nm with a motor side disturbance observer (DOB). In this work, a similar approach is adopted due to its simplicity, with DOB installed on the load side as shown in Fig. 23.

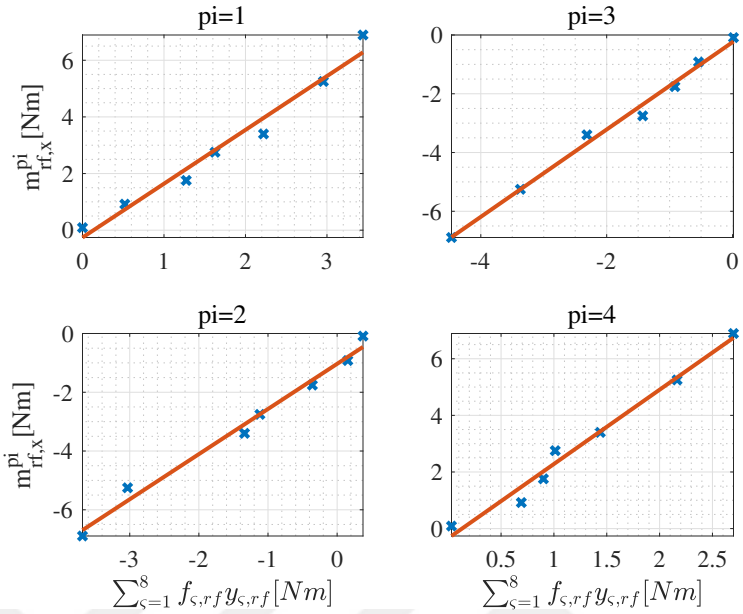

Figure 16: Right foot measured moment around x axis regarding point pi is represented by blue markers. The obtained linear fitted function is shown in a solid orange line.

The cascaded PID control loops are made up of an inner PI velocity control loop that receives feedback from the motor velocity $\dot{\theta}_m$, and an output of the torque control loop that computes the referential motor velocity signal, $\dot{\theta}_m^{ref}$.A P controller is used to impose torque tracking. The scaled spring deflection θ_s with spring stiffness K_s is considered the torque feedback. The load referential torque signal τ_l^{ref} is computed as a position control loop output.

The torque control loop also can be used to include the exoskeleton robot inverse dynamics solution τ_{ID}. In the case of disabling the position control loop by setting $\tau_l^{ref} = 0\ Nm$, a transparent mode is triggered and the motor can be back driven. The load angular displacement is considered as $(\theta_l^{act} = \theta_s + \theta_m)$, and referential load angular displacement is denoted as θ_l^{ref}. The position control loop on top of the torque control loop allows a virtual impedance with θ_l^{ref} as a nominal value. The controller enables compliance with the motion that can softly absorb the unmodeled contact transitions softly.

Two additional control signals are used to improve the motor performance. A feedforward control signal τ_{ff} is used to improve the torque bandwidth, and DOB is used to

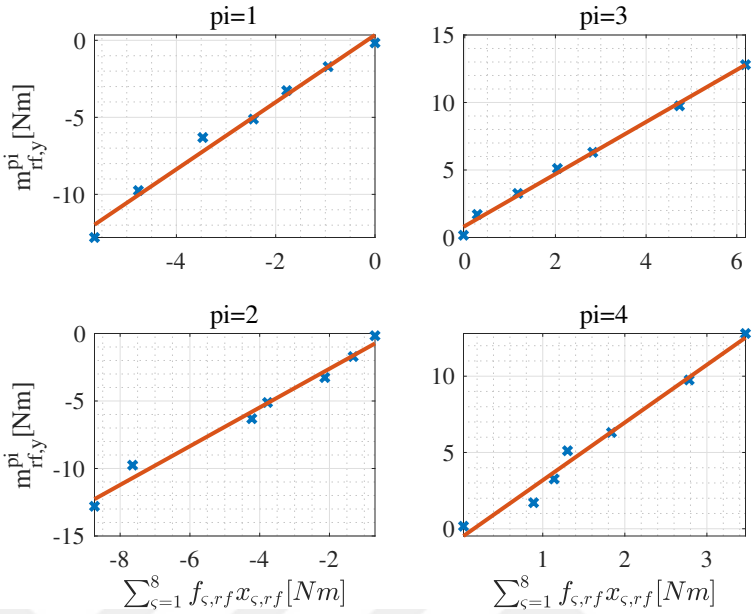

Figure 17: Right foot measured moment around y axis regarding point pi is represented by blue markers. The obtained linear fitted function is shown in a solid orange line.

improve the robustness of the controller against uncertainties and disturbances. Due to the presence of P_n^{-1} in the controller model, a second-order filter

$$Q_2(s) = \frac{w_c^2}{s^2 + 2\zeta w_c s + w_c^2} \tag{2.13}$$

is applied with damping ratio $\zeta = \frac{1}{\sqrt{2}}$ and properly tuned cut-off frequency w_c to make P_n^{-1} realizable, due to the presence of third and second order differentiation term as given by (2.12).

2.5 Joint Level Control Verification

2.5.1 Motor Dynamics Identification

To apply the control algorithm provided by Fig. 23 it is required to find the SEA model's (2.12) parameters. A simplified identification experiment is carried out by injecting a motor torque τ_m in a square waveform while acquiring the motor position, and velocity $(\theta_m,\ \dot{\theta}_m)$ using a motor side encoder. The motor torque is injected in the form of a motor current signal scaled with motor constant $k_t = 0.11\ Nm/A$ given by the motor data sheet [63]. A

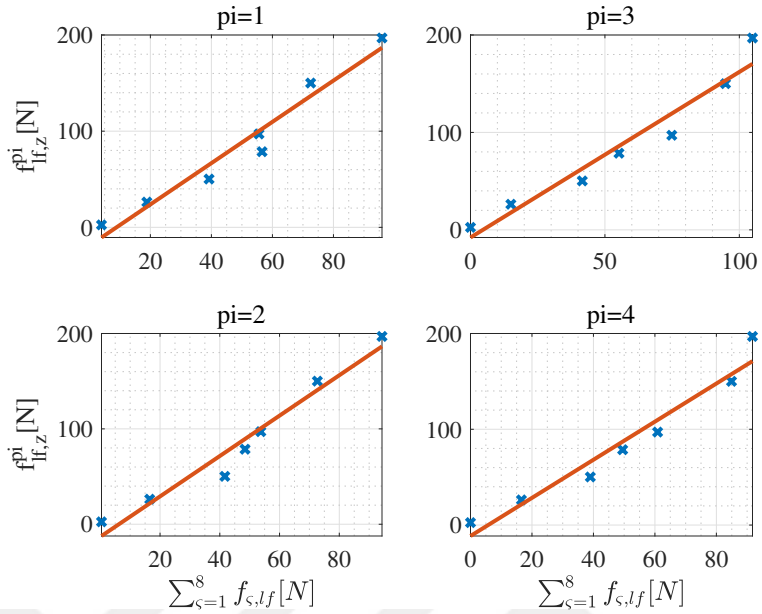

Figure 18: Left foot measured vertical force regarding point *pi* is represented by blue markers. The obtained linear fitted function is shown in a solid orange line.

simplified motor model with an unloaded shaft is given as follows:

$$\tau_m = J_{eq}\ddot{\theta}_m + (B_{eq1}K_1 + B_{eq2}K_2)\dot{\theta}_m + \tau_{st1}K_1 + \tau_{st2}K_2, \tag{2.14}$$

is used. The parameter J_{eq} is denoted as the equivalent inertia projected on the motor shaft. Symbols (B_{eq1}, B_{eq2}), and (τ_{st1}, τ_{st2}) are viscous and stiction friction coefficients in clockwise and counterclockwise directions, respectively. The variables K_1 and K_2 are used to enable and disable the friction terms depending on the rotation direction. $sgn(\bullet)$ is a piecewise function used to generate 1 for positive input, 0 for zero input, and -1 for negative input. The variables K_1 and K_2 are developed, similar to [64], as follows:

$$K_1 = 0.5|sgn(\dot{\theta}_m)||(1 + sgn(\dot{\theta}_m))|, \tag{2.15}$$

$$K_2 = 0.5|sgn(\dot{\theta}_m)||(1 - sgn(\dot{\theta}_m))|, \tag{2.16}$$

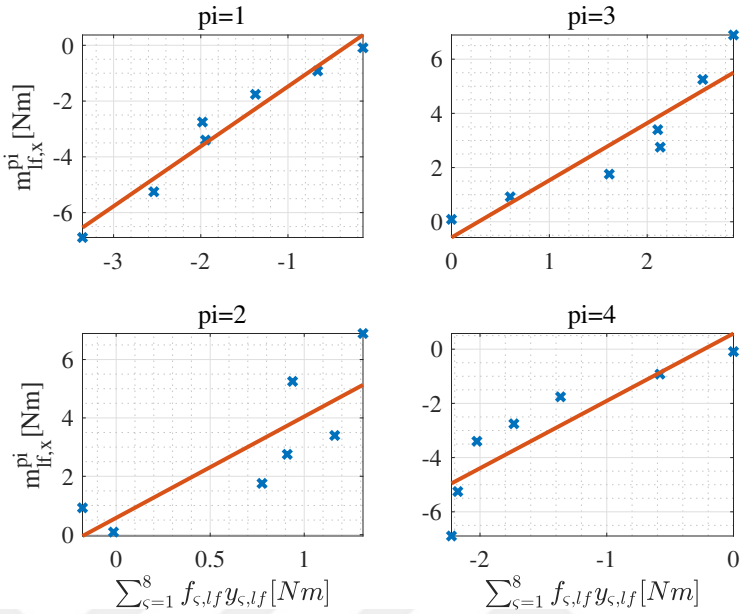

Figure 19: Left foot measured moment around x axis regarding point pi is represented by blue markers. The obtained linear fitted function is shown in a solid orange line.

The simplified motor model given by (2.14), can simply be represented in a linear form with respect to the motor parameters as follows:

$$\tau_m = \begin{bmatrix} \ddot{\theta}_m & K_1\dot{\theta}_m & K_2\dot{\theta}_m & K_1 & K_2 \end{bmatrix} \begin{bmatrix} J_{eq} \\ B_{eq1} \\ B_{eq2} \\ \tau_{st1} \\ \tau_{st2} \end{bmatrix}. \tag{2.17}$$

A coefficients matrix function in $\dot{\theta}_m$, and $\ddot{\theta}_m$ is to be obtained using the measured encoder data. To obtain the acceleration signal, a double differentiation of θ_m is needed, which may not be practically possible. Hence MATLAB $smooth(\bullet)$ function is used, and the identification procedure is executed offline. The actual motor torque $\tau_{m,a}$ and motor angular position, velocity, and acceleration are acquired for n_s samples. Hence, equation

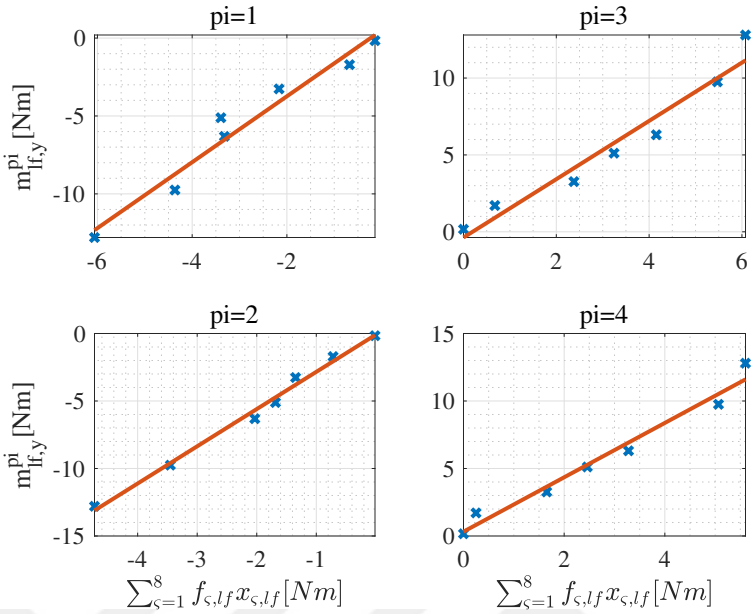

Figure 20: Left foot measured moment around y axis regarding point pi is represented by blue markers. The obtained linear fitted function is shown in a solid orange line.

(2.17) can be rewritten in terms of multi-sampled data as follows:

$$\overline{\boldsymbol{\tau}}_m = \boldsymbol{\Phi} \begin{bmatrix} J_{eq} \\ B_{eq1} \\ B_{eq2} \\ \tau_{st1} \\ \tau_{st2} \end{bmatrix}, \tag{2.18}$$

where $\overline{\boldsymbol{\tau}}_m \in \mathbb{R}^{n_s}$ is a vector that stores the actual motor torque for different samples. The matrix variable $\boldsymbol{\Phi} \in \mathbb{R}^{n_s \times 5}$ denotes the regressor and can be obtained using (2.17) such that $\boldsymbol{\Phi} = [\overline{\boldsymbol{\phi}}_1^\intercal \; \overline{\boldsymbol{\phi}}_2^\intercal \; ... \; \overline{\boldsymbol{\phi}}_{n_s}^\intercal]^\intercal$, for any arbitrary matrix $\overline{\boldsymbol{\phi}}_k = \begin{bmatrix} \ddot{\theta}_m^k & K_1^k \dot{\theta}_m^k & K_2^k \dot{\theta}_m^k & K_1^k & K_2^k \end{bmatrix}$ at sample

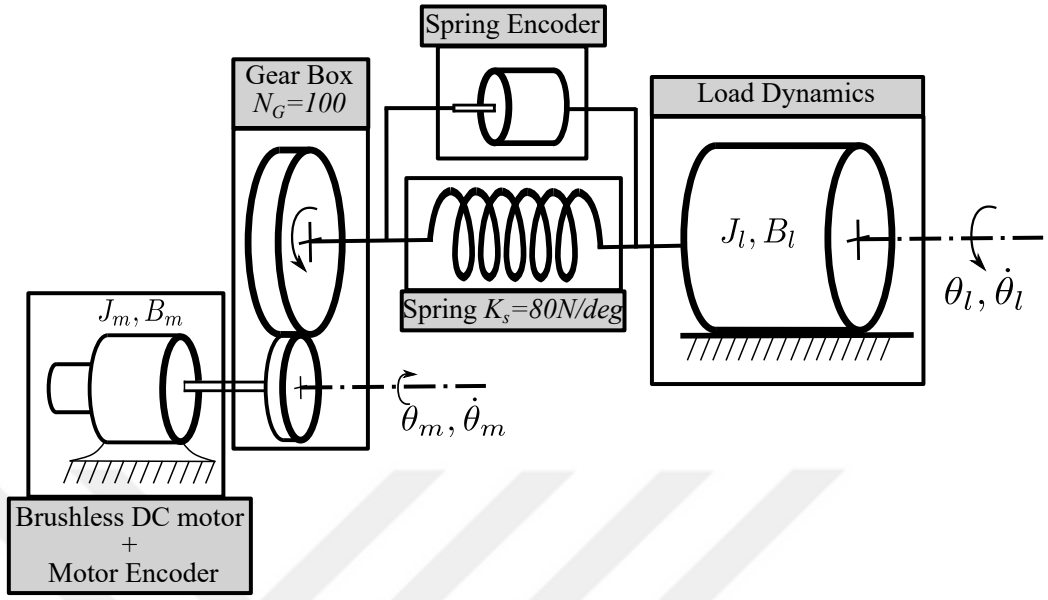

Figure 21: SEA schematic diagram.

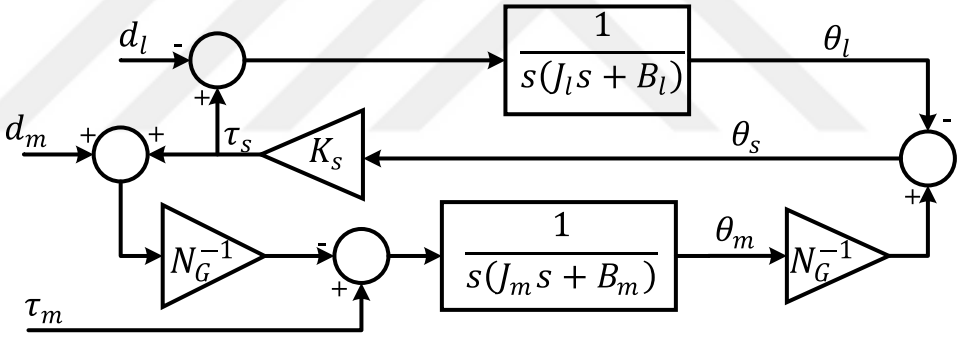

Figure 22: SEA dynamic model.

k. The estimated motor parameters can be readily obtained using (2.18) such that:

$$\begin{bmatrix} \hat{J}_{eq} \\ \hat{B}_{eq1} \\ \hat{B}_{eq2} \\ \hat{\tau}_{st1} \\ \hat{\tau}_{st2} \end{bmatrix} = \mathbf{\Phi}^{\dagger} \bar{\boldsymbol{\tau}}_m, \tag{2.19}$$

where, superscript † represents the pseudo inverse operation given by $\mathbf{\Phi}^{\dagger} = (\mathbf{\Phi}^{\intercal}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\intercal}$. Through applying equation (2.19) over $n_s = 30000$ samples with sampling rate $t_s = 0.5\ ms$.

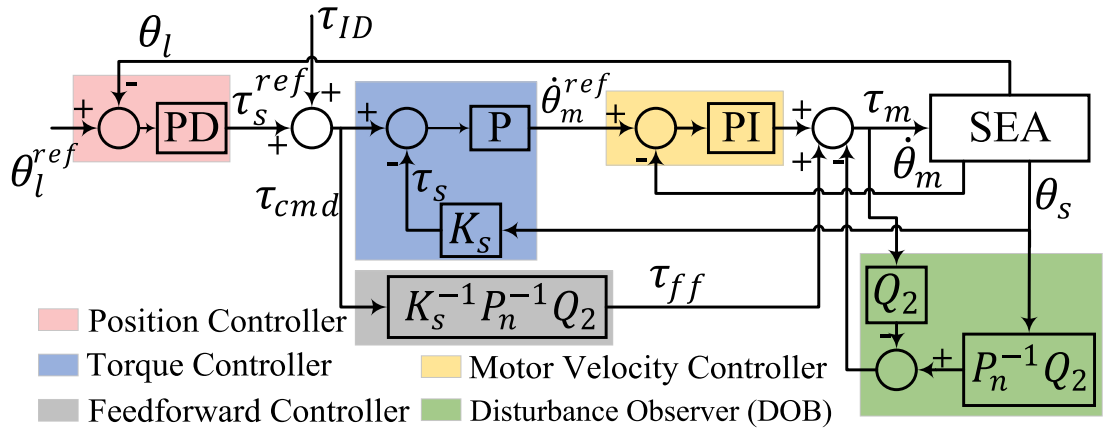

Figure 23: SEA impedance control loop.

The motor input torque signal is shown by the solid blue line in Fig. 24. Its peak-to-peak value is 1 Nm, the peak value is 0.5 Nm, and the signal frequency is 1 Hz. The estimated parameters are $\hat{J}_{eq} = 1.52 \times 10^{-4}\ kgm^2$, $\hat{B}_{eq1} = 0.001781\ Nms/rad$, $\hat{B}_{eq2} = 0.001639\ Nms/rad$, $\hat{\tau}_{st1} = 0.1132\ Nms/rad$, and $\hat{\tau}_{st2} = -0.1188\ Nms/rad$. The estimated torque $\hat{\tau}_m$ is computed using (2.18) based on the estimated parameters. Fig. 24 shows $\hat{\tau}_m$ via a solid orange line. The correlation between $\hat{\tau}_m$ and τ_m^a is computed using MATLAB function $corr(\bullet)$ and it is obtained value is 0.98. To obtain the motor and load parameters shown in Fig. 22, the following relations are assumed:

$$J_m = 0.1 J_l, \tag{2.20}$$

$$B_m = 0.1 B_l, \tag{2.21}$$

$$B_{eq} = \frac{B_{eq1} + B_{eq2}}{2}, \tag{2.22}$$

$$J_{eq} = J_m + \frac{J_l}{N_G^2}, \tag{2.23}$$

$$B_{eq} = B_m + \frac{B_l}{N_G^2}, \tag{2.24}$$

2.5.2 Torque Tracking

To verify the torque tracking performance of the torque control loop shown in Fig. 23, different sinusoidal referential load torque τ_l^{ref} with distinct frequencies were used. The

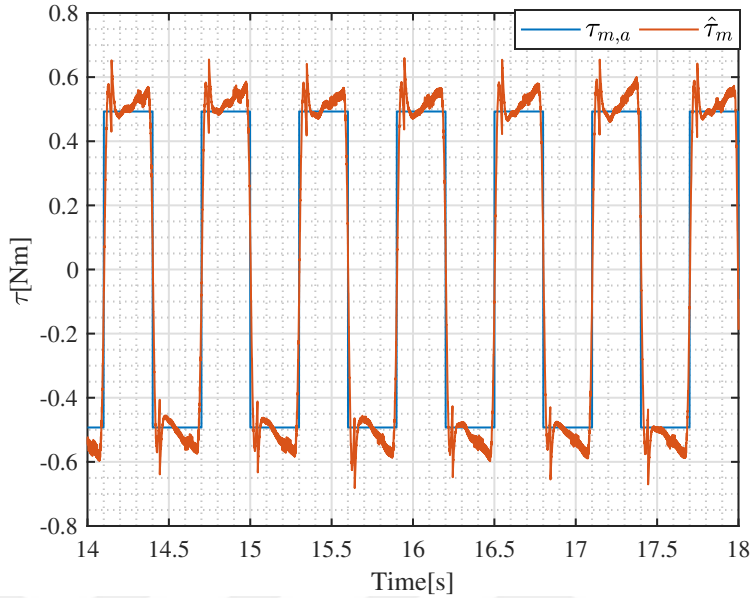

Figure 24: The motor torque curves used for identification, in which the actual torque $\tau_{m,a}$ is shown in solid blue and estimated torque $\hat{\tau}_m$ is shown in solid orange.

actual load torque τ_l^a is shown in Fig. 25. The frequency is denoted by $freq$ and the amplitude ratio is symbolized by M. The referential trajectories are shown in a solid blue line while the actual signal is shown in a solid black line. The sinusoidal frequencies $freq$ were given in 0.5, 1, and 2 Hz in Fig. 25.a, .b, and .c, respectively. The referential torque peak value is -0.5 Nm and the peak-to-peak value is 0.5 Nm. The torque response magnitude ratios corresponding to frequencies $freq = 0.5$, 1, and 2 Hz are $M = 1.006$, 0.98022, and 0.9358 which indicate good torque tracking performance for the available hardware setup with respect to the required walking motion frequency.

2.5.3 Position Tracking

To test the position control loop, a set of referential sinusoidal load angular displacement trajectories θ_l^{ref} is injected into the control loop. The actual link angular displacement θ_l^a is acquired as the sum of motor and spring angular displacements. Fig. 26 shows the responses in the solid black line and the referential signals in the solid blue line. The referential signals are injected in three different frequencies given as $freq = 1$, 2,, and 4 Hz.

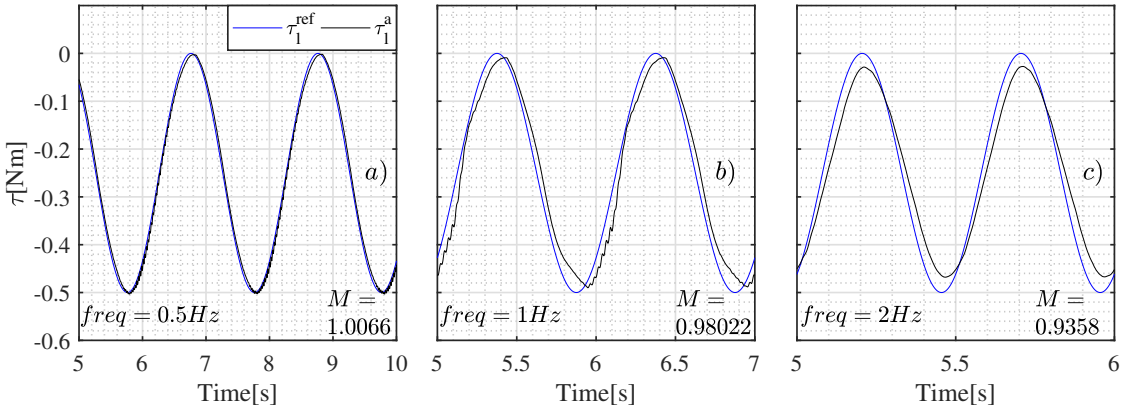

Figure 25: Load torque tracking for different referential sinusoidal trajectories with different frequencies.

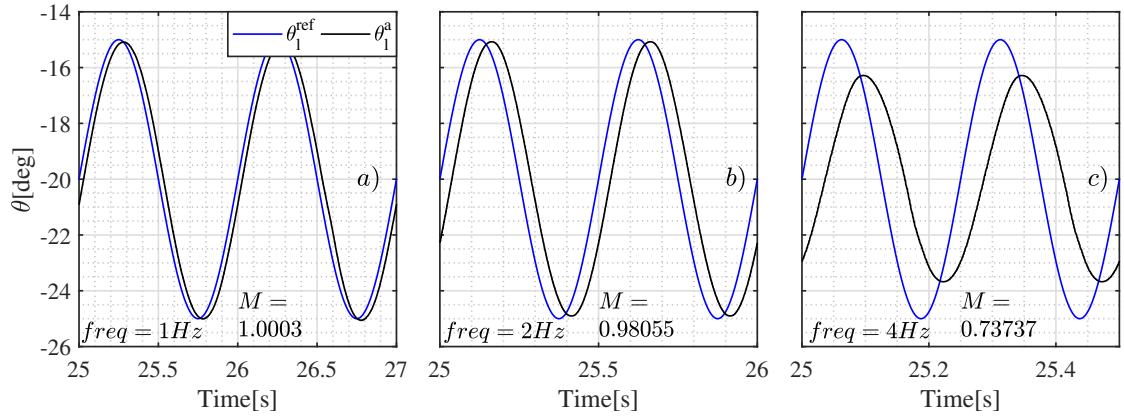

Figure 26: Load position tracking for different referential sinusoidal trajectories with different frequencies.

The responses show good tracking in terms of the amplitude ratio $M = 1.0003,\ 0.98055,$ and 0.73737 for $freq = 1,\ 2,$ and 4 Hz, respectively. Hence, the joint position tracking shows an acceptable performance up to 2 Hz which is sufficient to develop the dynamic walking motion for a given Co-Ex robot.

2.6 Leg Mechanism Kinematics

Due to the indirect connection between the motors and the Co-Ex joints, as shown in Fig. 9, it is required to solve the Co-Ex leg mechanism kinematics (internal kinematics). Thus, a proper mapping between motor and joint angular displacement, velocity, and torque is

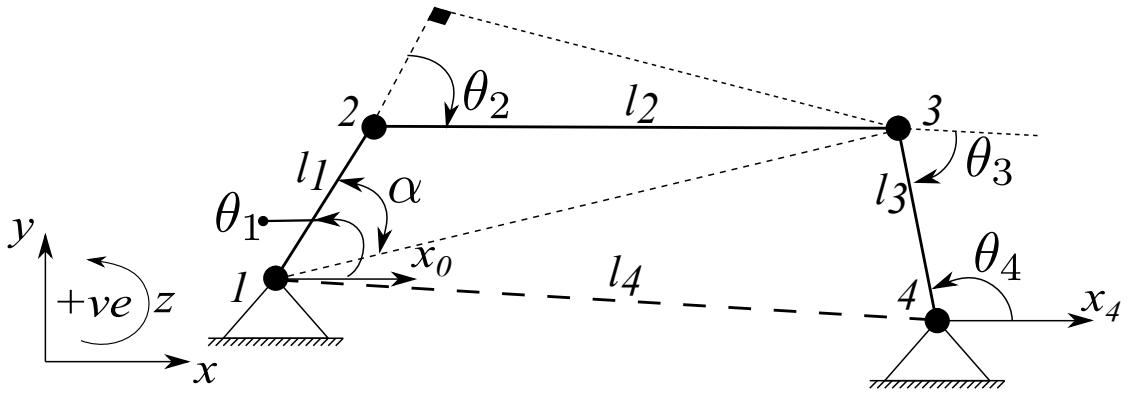

Figure 27: Four bar mechanism schematic diagram.

obtained.

In Co-Ex a set of four bar mechanisms is used to transmit motion from the motors to the joints. In this work, an analytical solution is used to solve the four bar mechanism kinematically using the general scheme shown in Fig. 27. The motor and robot joints are assumed to be installed at θ_1 and θ_4, respectively. The link lengths are denoted by $l_1, l_2, l_3,$, and l_4 are assumed to be known from the physical robot dimensions. The desired robot joint variables are assumed to be known and the objective is to find an equivalent motor joint angular displacement, velocity, and torque. In the schematic diagram shown in Fig. 27, four nodes are available, representing the mechanism of revolute joints. The origin is assumed to be at node 1. The Cartesian displacement of node 3 with respect to 1 is given as:

$$x_3 = l_{4,x} + l_3 cos(\theta_4), \tag{2.25}$$

$$y_3 = l_{4,y} + l_3 sin(\theta_4), \tag{2.26}$$

where, $l_{4,x}$ and $l_{4,y}$ represents the Cartesian displacement of node 4 with respect to 1. The angular displacement, θ_2 can be obtained using the cosine rule as follows:

$$D = \frac{x_3^2 + y_3^2 - l_1^2 - l_2^2}{2 l_1 l_2}, \tag{2.27}$$

$$\theta_2 = atan2(\sqrt{1 - D^2}, D). \tag{2.28}$$

Using θ_2 angle α can be computed as follows:

$$\alpha = atan2(l_2 sin(\theta_2), l_1 + l_2 cos(\theta 2)). \tag{2.29}$$

From Fig. 27 angle θ_1 can be obtained as:

$$\theta_1 = \alpha + atan2(y_3, x_3). \tag{2.30}$$

To obtain the motor velocity in terms of desired joint velocity, x_3 and y_3 are firstly obtained in terms of θ_1. Using the scheme shown in Fig. 27, the following equations can be obtained.

$$x_3 = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2), \tag{2.31}$$

$$y_3 = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2), \tag{2.32}$$

Exploiting equations (2.25-2.26) and (2.31-2.32), following equality can be concluded:

$$l_{4,x} + l_3 cos(\theta_4) = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2), \tag{2.33}$$

$$l_{4,y} + l_3 sin(\theta_4) = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2). \tag{2.34}$$

By differentiating equations (2.33) and (2.34) with respect to time, the following equations are obtained:

$$l_3 \omega_4 sin(\theta_4) = l_1 \omega_1 sin(\theta_1) + l_2(\omega_1 + \omega_2) sin(\theta_1 + \theta_2), \tag{2.35}$$

$$l_3 \omega_4 cos(\theta_4) = l_1 \omega_1 cos(\theta_1) + l_2(\omega_1 + \omega_2) cos(\theta_1 + \theta_2), \tag{2.36}$$

where, ω_1, ω_2, and ω_4 are the angular velocities of joints 1, 2, and 4, respectively. Through multiplying equation (2.35) by $cos(\theta_1 + \theta_2)$ and (2.36) by $sin(\theta_1 + \theta_2)$ then subtracting the resultant equations together, the following relation can be concluded

$$\omega_1 = \omega_4 \frac{l_3 sin(\theta_4 - \theta_1 - \theta_2)}{l_1 sin(-\theta_2)}. \tag{2.37}$$

Assuming an ideal mechanism with zero power loss, an input/output torque relation can be obtained by inverting (2.37)

$$\tau_1 = \tau_4 \frac{l_1 sin(-\theta_2)}{l_3 sin(\theta_4 - \theta_1 - \theta_2)}, \tag{2.38}$$

where, τ_1 and τ_4 are the torques at joints 1 and 4, respectively.

CHAPTER III

CO-EX KINODYNAMICS

3.1 Floating Base Kinematics

Co-Ex bipedal exoskeleton is a floating base robot that is freely moving in 3D Cartesian space. To provide a rigorous framework to solve its kinematics, 6 virtual joints are introduced to connect a fixed global frame $\{F_G\}$ in the space and floating base rigidly attached frame $\{F_0\}$ as shown in Fig. 28 [43]. Consequently, the joint vector is expressed as:

$$\mathbf{q} = [\mathbf{x}_b^\intercal \ \mathbf{q}_r^\intercal]^\intercal, \tag{3.1}$$

where $\mathbf{x}_b \in \mathbb{R}^6$ represents the floating base coordinates, $\mathbf{q}_r \in \mathbb{R}^{n_r}$ is the robot joint coordinates, and n_r represents the number of the robot active joints. Since the proposed robot is bipedal, three end effectors are considered, namely, right foot rf, left foot lf, the center of mass *CoM* position $\mathbf{x}_{com}$, and torso orientation. The *CoM* is the weighted average of the individual links' *CoMs*. Its *CoM* Jacobian can be computed as:

$$\mathbf{J}_{com} = \frac{\sum_{i=1}^{n_b} m_i \mathbf{J}_{i,com}}{\sum_{i=1}^{n_b} m_i}, \tag{3.2}$$

where, $\mathbf{J}_{com} \in \mathbb{R}^{3\times(6+n_r)}$ is the *CoM* Jacobian matrix, $\mathbf{J}_{i,com} \in \mathbb{R}^{3\times(6+n_r)}$ is i^{th} links' *CoM* Jacobian matrix, and m_i is the mass of each individual link. The symbol n_b represents the number of rigid bodies. The Jacobian matrix $\mathbf{J} \in \mathbb{R}^{(18\times(6+n_r))}$ is a stack of end effector Jacobian matrices

$$\mathbf{J} = [\mathbf{J}_{lf}^\intercal \ \mathbf{J}_{rf}^\intercal \ \mathbf{J}_{torso}^\intercal \ \mathbf{J}_{com}^\intercal]^\intercal, \tag{3.3}$$

where, the symbols $\mathbf{J}_{lf} \in \mathbb{R}^{(6\times6+n_r)}$, $\mathbf{J}_{rf} \in \mathbb{R}^{(6\times6+n_r)}$, and $\mathbf{J}_{torso} \in \mathbb{R}^{(3\times6+n_r)}$, represents the left foot, right foot, and torso orientation Jacobian matrices respectively. In principle, the developed Jacobian matrix in (3.3) is used to map the commanded Cartesian velocity

trajectories $\dot{\mathbf{x}}^{cmd} \in \mathbb{R}^{18}$ to the joint velocities such that:

$$\dot{\mathbf{x}}^{cmd} = \mathbf{J}\dot{\mathbf{q}}. \tag{3.4}$$

Since it is required to drive the robot through its joints via $\dot{\mathbf{q}}_r$, (3.4) is needed to be solved. Firstly obtain $\dot{\mathbf{q}}$ given $\dot{\mathbf{x}}^{cmd}$.

Then, $\dot{\mathbf{q}}_r$ is obtained using $\dot{\mathbf{q}}$ by exploiting the constrained Jacobian matrix $\mathbf{J}_C \in \mathbb{R}^{6n_c \times (6+n_r)}$. The parameter n_c is the number of feet in contact with the ground. The contact feet differential kinematics can be represented as:

$$\mathbf{0} = \begin{bmatrix} \mathbf{J}_{c,b} & \mathbf{J}_{c,r} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_b \\ \dot{\mathbf{q}}_r \end{bmatrix}, \tag{3.5}$$

where $\mathbf{J}_{c,b} \in \mathbb{R}^{(6n_c \times 6)}$ is the constraint Jacobian sub-matrix that corresponds to the floating base coordinates. The sub-matrix $\mathbf{J}_{c,r} \in \mathbb{R}^{(6n_c \times n_r)}$ of the constraint Jacobian corresponds to the robot joint coordinates. Assuming $Rank(\mathbf{J}_{c,b} \in \mathbb{R}^{(6n_c \times 6)}) \geq 6$, $\dot{\mathbf{q}}_r$ can be readily obtained using (3.5) such that:

$$\dot{\mathbf{q}}_r = \begin{bmatrix} -\mathbf{J}_{c,b}^{\dagger}\mathbf{J}_{c,r} & \mathbf{1}_{n_r} \end{bmatrix}^{\dagger} \dot{\mathbf{q}}, \tag{3.6}$$

where, $\mathbf{1}_{n_r}$ is an identity matrix of order n_r.

3.2 Kinematics Optimization

As each leg possesses solely four active joints, the robot Jacobian matrix $\mathbf{J}$, given by (3.4), has the number of columns less than rows $(6+n_r < 18)$. Consequently, there is no possible solution to justify all the required constraints. Thus, an optimization algorithm could be employed to obtain the best-fitted trajectories. Kinematically feasible joint velocities for given task velocities can be generated in terms of the best-fitted task velocities concerning the desired gait trajectories using an optimization algorithm. The optimization is performed by taking into account the following: i) it minimizes the residual terms that represent the difference between the optimized solution and the desired gait trajectories; ii) the optimization problem is solved in multiple priority levels such that the tasks with lower priority are

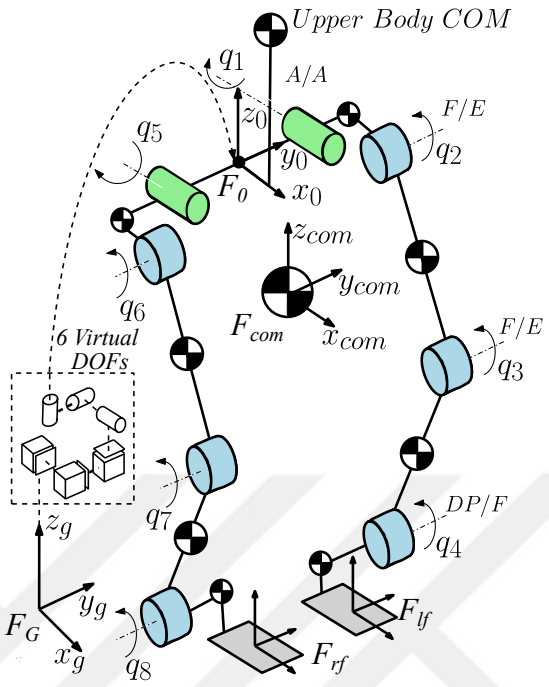

Figure 28: Bipedal exoskeleton Co-Ex joint configuration and the definition of frames.

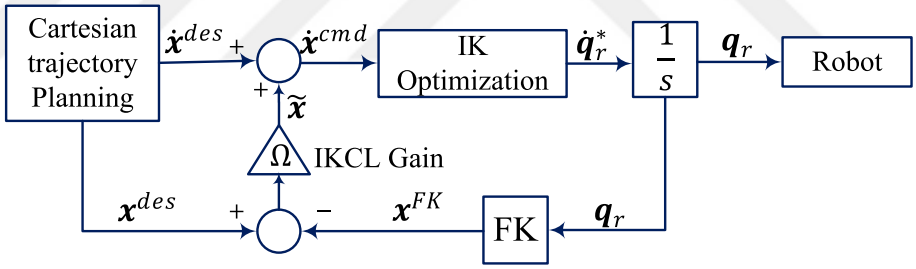

Figure 29: The block diagram of the inverse kinematics algorithm. FK refers to forward kinematics.

solved without disrupting the tasks with higher priority [36,65], iii) during the optimization process, joint limitations are taken into account; and iv) due to numerical integration drift, a closed loop term is added.

Holonomic Constraints: These constraints represent the desired feet, torso, and CoM trajectories $\mathbf{x}^{des} \in \mathbb{R}^{18}$ in the task space which is defined between the robot coordinate frames and the global coordinate frame $\{F_G\}$. The objective of the optimization problem is to minimize the deviation between the robot coordinates and gait trajectories. Since the optimization problem is solved in terms of differential kinematics, it is required to impose

the assigned constraint as a differential expression: $\dot{\mathbf{x}}^{des}$. Hence, the error norm between the desired and actual trajectories is calculated below:

$$\delta(\dot{\mathbf{q}}) = ||\dot{\mathbf{x}}^{cmd} - \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}_r|| \tag{3.7}$$

The term $\delta(\dot{\mathbf{q}}) : \mathbb{R}^{6+n_r} \in \mathbb{R}^8$ represents the residual function between the desired and actual task trajectories in terms of robot joint velocities $\dot{\mathbf{q}}_r$.

Task Prioritization: The proposed approach depends on dividing the task trajectories into two sets, based on their priority levels. The first set is responsible for achieving the generated gait that is based on ZMP; see section 3.4. The second set is responsible for reducing the variations in CoM height, and torso and swinging foot orientation. The first set is solved at a higher priority level since it is related to robot balance. The joint velocity set $\dot{\mathbf{q}}_r$ is represented as the summation of two terms that depict the previous prior solution and the null space motion concerning the previous prior tasks:

$$\dot{\mathbf{q}}_r = \mathbf{J}^{\dagger}_{p-1}\dot{\mathbf{x}}^{cmd}_{p-1} + (\mathbf{1} - \mathbf{J}^{\dagger}_{p-1}\mathbf{J}_{p-1})\mathbf{b}. \tag{3.8}$$

In (3.8), $\mathbf{1}$ is an identity matrix with proper size to the number of task trajectories selected in the prioritization level. The subscript p indicates the priority level, e.g, $p-1$ is the prior level. The † superscript indicates matrix pseudo-inverse, while the vector $\mathbf{b} \in \mathbb{R}^{n_r}$ contained a set of optimized variables concerning the constraints with less priority. The operator $\mathcal{N}_{p-1} = (\mathbf{1} - \mathbf{J}^{\dagger}_{p-1}\mathbf{J}_{p-1})$ is used to point the joint velocity modification variable $\mathbf{b}$ to the null space of the prior constraint Jacobian $\mathbf{J}_{p-1}$. The term $\dot{\mathbf{q}}^{*}_{r_{p-1}} = \mathbf{J}^{\dagger}_{p-1}\dot{\mathbf{x}}^{cmd}_{p-1}$ represents the optimized joint velocities concerning the prior constraints. The term $(\mathbf{1} - \mathbf{J}^{\dagger}_{p-1}\mathbf{J}_{p-1})\mathbf{b}$ stands for the joint modification term that prevents less prior constraints to interfere with higher prior constraints.

Joint Range of Motion (ROM): The ROM represents the feasible limit for the gait motion in the joint space. As the exoskeleton robot is coupled with the human, the feasible joint limits are dictated by the human joint limits [66]. The joints ROM are proposed

in terms of joint angular displacement limits: $\{\underline{\dot{\mathbf{q}}_r} = \frac{\underline{\mathbf{q}_r} - \mathbf{q}}{t_s},\ \overline{\dot{\mathbf{q}}_r} = \frac{\overline{\mathbf{q}_r} - \mathbf{q}}{t_s}\}$, where t_s is the sampling time, $\underline{\dot{\mathbf{q}}_r}$, and $\overline{\dot{\mathbf{q}}_r}$ are the minimum and maximum joint velocities, respectively. The joints ROM are imposed to the objective function in the form of a barrier function $\psi(\dot{\mathbf{q}}_r)$. It should satisfy the following conditions to preserve the differentiability and convexity of the objective function: i) $\psi(\dot{\mathbf{q}}_r)$ is a convex function in $\dot{\mathbf{q}}_r$, ii) $\psi(\dot{\mathbf{q}}_r)$ is continuously differentiable with $\dot{\mathbf{q}}_r$, iii) $\psi(\dot{\mathbf{q}}_r) \to \infty, \forall\ \dot{\mathbf{q}}_r \in \{\underline{\dot{\mathbf{q}}_r}, \overline{\dot{\mathbf{q}}_r}\}$, iv) $\psi(\dot{\mathbf{q}}_r) \to \mathbb{C}^{n_r}, \forall\ \dot{\mathbf{q}}_r \notin \{\underline{\dot{\mathbf{q}}_r}, \overline{\dot{\mathbf{q}}_r}\}$. The set $\mathbb{C}^{n_r}$ is a constant valued set, where the barrier function $\psi(\dot{\mathbf{q}}_r)$ tends to output constant values if the joint variables are not close to the joint limits $\{\underline{\dot{\mathbf{q}}_r}, \overline{\dot{\mathbf{q}}_r}\}$.

Auxiliary Functions: To tackle the numerical drift problem, an Inverse Kinematics Closed Loop (IKCL) term is embedded in the optimization objective function. It adds a correction factor to the optimization term that accounts for numerical approximation error:

$$\tilde{\mathbf{x}} = \Omega(\mathbf{x}^{des} - \mathbf{x}^{FK}), \tag{3.9}$$

In (3.9), $\mathbf{x}^{des}$ is the desired task trajectories, $\mathbf{x}^{FK}$ is the forward kinematics solution, and $\Omega \in \mathbb{R}^{18\times 18}$ is a positive definite diagonal gain matrix to tune the convergence rate of the closed loop. The gain matrix in this work is tuned in a way to guarantee fast convergence and recovery of the joint trajectories. The objective function obtained from (3.7) and (3.8) constitutes the following unconstrained optimization problem,

$$\begin{aligned} \mathbf{b}^* &= \arg\min_{\forall \mathbf{b} \in \mathbb{R}^{n_r}} (||\dot{\mathbf{x}}_p^{cmd} - \mathbf{J}_p(\dot{\mathbf{q}}_{r,p-1} + \mathscr{N}_{p-1}\mathbf{b})||^2 \\ &+ \overline{\boldsymbol{\mu}}\,\psi(\dot{\mathbf{q}}_{r,p})) \\ \dot{\mathbf{q}}_{r,p}^* &= \dot{\mathbf{q}}_{r,p-1}^* + \mathscr{N}_{p-1}\mathbf{b}^*, \end{aligned} \tag{3.10}$$

where $\dot{\mathbf{x}}_p^{cmd}$ is commanded variable: $\dot{\mathbf{x}}_p^{cmd} = \dot{\mathbf{x}}_p^{des} + \tilde{\mathbf{x}}_p$. It is used to embody the closed loop correction term $\tilde{\mathbf{x}}_p$ and the desired task trajectories vector $\dot{\mathbf{x}}_p^{des}$ at the priority level p. The positive coefficient row vector $\overline{\boldsymbol{\mu}}$ was used to approximate the interference between the barrier function $\psi(\dot{\mathbf{q}}_r) = [-log(\dot{q}_1)\ \ -log(\dot{q}_1)\ \ ...\ \ -log(\dot{q}_{n_r})]^\intercal$ and the optimization

algorithm. It can be assigned as $\overline{\boldsymbol{\mu}} = \mu ones(1, n_r)$, in which the parameter μ is a positive scalar initially started with a large value, and then gradually reduced with optimization iterations. The function $ones(1, n_r)$ represents a row vector of ones with a length of n_r. See Fig. 29 for the block diagram concerning the joint trajectory generation.

3.3 Kinodynamic Optimization

To enable the locomotion controllers, the optimization problem is resolved into velocity and acceleration levels. Thus, a direct connection with the robot dynamic model is established to permit a linear mapping between the optimization variables and the contact forces.

3.3.1 Floating Base Dynamics

The floating base dynamics play a key role in the locomotion controllers to compute the CMM (Centroidal Momentum Matrix) and multibody ZMP via contact forces. The dynamics of a floating-base system can be expressed as in the following [67],

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{S}^{\intercal}\boldsymbol{\tau} + \mathbf{J_c}(\mathbf{q})^{\intercal}\boldsymbol{\lambda}_c, \tag{3.11}$$

where, $\boldsymbol{\tau}$ is robot joints torque vector, $\mathbf{H}(\mathbf{q}) \in \mathbb{R}^{(6+n_r)\times(6+n_r)}$ is the inertia matrix,$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{(6+n_r)\times(6+n_r)}$ is Coriolis and centrifugal matrix, $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^{(6+n_r)\times 1}$ stands for the gravitational terms, $\mathbf{S}$ is the selection matrix for actuated joints ($\mathbf{S} = [\mathbf{0}_{n_r\times 6}\ \mathbf{1}_{n_r}]$), n_c is the number of contact bodies, and $\boldsymbol{\lambda}_c \in \mathbb{R}^{6n_c\times 1}$ is the vector that stores spatial forces at contacts. For multiple contacts, $\boldsymbol{\lambda}_c$ is expressed as $\boldsymbol{\lambda}_c = [{}^1\boldsymbol{\lambda}_c^{\intercal}\ {}^2\boldsymbol{\lambda}_c^{\intercal}\ \cdots\ {}^{n_c}\boldsymbol{\lambda}_c^{\intercal}]^{\intercal}$. Each contact force component is defined as $\lambda_{c,j} \in \{m_{c,x}, m_{c,y}, m_{c,z}, \lambda_{c,x}, \lambda_{c,y}, \lambda_{c,z}\}$ with three rotational and translational components along x, y, and z axes, respectively. Matrix $\mathbf{H}(\mathbf{q})$ in (3.11) can be decomposed into four sub-matrices [1]:

$$\mathbf{H}(\mathbf{q}) = \begin{bmatrix} \mathbf{I}_{fb} & \overline{\mathbf{F}} \\ \overline{\mathbf{F}}^{\intercal} & \mathbf{H}_r \end{bmatrix} \tag{3.12}$$

where $\mathbf{I}_{fb} \in \mathbb{R}^{6\times6}$ represents the aggregated links' spatial inertia projected onto the floating-base, $\overline{\mathbf{F}} \in \mathbb{R}^{6\times n_r}$ is a matrix whose columns store the spatial forces required at the floating-base to provide unit acceleration for each joint variable, and $\mathbf{H}_r \in \mathbb{R}^{n_r\times n_r}$ is the joint space inertia matrix. The inertia matrix $\mathbf{H}(\mathbf{q})$ can be obtained using the composite rigid body algorithm (CRBA), $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is obtained analytically using a modified recursive Newton-Euler (NE) method[1], and $\mathbf{G}(\mathbf{q})$ is computed using the recursive NE algorithm [1]. The contact Jacobian matrix $\mathbf{J}_c(q)$ can be described as follows:

$$\mathbf{J}_c(q) = \begin{bmatrix} {}^0\mathbf{X}_c^* \\ \mathbf{J}_{c,r}^\intercal \end{bmatrix}^\intercal . \tag{3.13}$$

In (3.13), ${}^0\mathbf{X}_c^*$ represents the spatial transformation matrix in the force space defined from the contact frame to the base frame $\{F_0\}$. In case of multiple contact points, ${}^0\mathbf{X}_c^*$ can be expressed as $({}^0\mathbf{X}_c^* = [{}^0\mathbf{X}_1^*\ {}^0\mathbf{X}_2^*\ \cdots\ {}^0\mathbf{X}_{n_c}^*])$. The floating base portion of the robot dynamics can be extracted if both sides of (3.11) are multiplied by the floating base selection matrix $\mathbf{S}_{fb} = [\mathbf{1}_6\ \mathbf{0}_{6\times n_r}]$:

$$[\mathbf{I}_{fb}\ \mathbf{F}]\ddot{\mathbf{q}} + \mathbf{C_{fb}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G_{fb}}(\mathbf{q}) = {}^0\mathbf{X}_c^*\boldsymbol{\lambda}_c \tag{3.14}$$

In (3.14), $\mathbf{C_{fb}}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{S}_{fb}\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ and $\mathbf{G_{fb}}(\mathbf{q}) = \mathbf{S}_{fb}\mathbf{G}(\mathbf{q})$.

Contact Force Computation: Using (3.14), the joint variables can be linearly mapped to the contact force:

$$\lambda_{c,j} = \mathbf{s}_{c,j}{}^c\mathbf{X}_0^*([\mathbf{I}_{fb}\ \overline{\mathbf{F}}]\ddot{\mathbf{q}} + \mathbf{C_{fb}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G_{fb}}(\mathbf{q})), \tag{3.15}$$

where $\mathbf{s}_{c,j} \in \mathbb{R}^{1\times6n_c}$ is force component selection vector. The transformation matrix ${}^c\mathbf{X}_0^*$ depends on the contact phase. During the left single support phase ${}^c\mathbf{X}_0^* = {}^{lf}\mathbf{X}_0^*$. Likewise, ${}^c\mathbf{X}_0^* = {}^{rf}\mathbf{X}_0^*$ for the right single support phase. For the double support phase, ${}^c\mathbf{X}_0^* = [{}^0\mathbf{X}_{rf}^*\ {}^0\mathbf{X}_{lf}^*]^\dagger$.

[1] https://github.com/AhmedFahmySoliman/RobustExoControl.

Centroidal Momentum Matrix (CMM): As described in [42], the centroidal momentum matrix, $\mathbf{A_g}$, constitutes a linear mapping between joint velocities and the centroidal momentum of the system,

$$\mathbf{h}_g = \mathbf{A}_g \dot{\mathbf{q}}, \tag{3.16}$$

where $\mathbf{h}_g = \begin{bmatrix} \boldsymbol{k}_g^\intercal & \boldsymbol{l}_g^\intercal \end{bmatrix}^\intercal$ is the centroidal momentum vector comprising the angular, $\boldsymbol{k}_g$, and translational, $\boldsymbol{l}_g$, components. In this connection, the CMM is yielded by multiplying the inertia matrix of (3.14) with the transformation matrix between the CoM and floating base:

$$\mathbf{A_g} = {}^{com}\mathbf{X}_0^* [\mathbf{I}_{fb}\ \overline{\mathbf{F}}]. \tag{3.17}$$

Note that the external wrench projected onto the CoM is equal to the rate of change of centroidal momentum: $\dot{\mathbf{h}}_g = [\dot{\boldsymbol{k}}_g^\intercal \quad \dot{\boldsymbol{l}}_g^\intercal]^\intercal$ The connection between $\dot{\mathbf{h}}_g$ and joint variables is obtained as follows:

$$\dot{\mathbf{h}}_g = \dot{\mathbf{A}}_\mathbf{g} \dot{\mathbf{q}} + \mathbf{A_g} \ddot{\mathbf{q}}. \tag{3.18}$$

The time derivative of CMM ($\dot{\mathbf{A}}_\mathbf{g}$) is obtained using the following expression:

$$\dot{\mathbf{A}}_\mathbf{g} = {}^{com}\mathbf{X}_0^* \mathbf{C_{fb}}(\mathbf{q}, \dot{\mathbf{q}}). \tag{3.19}$$

3.3.2 Prioritized Stack of Tasks Optimization Block

To realize the prioritization, a sequence of QP is solved such that the prior constraints are imposed on the lower prior constraints in the form of equality constraints. The proposed PST optimization block can be configured in centroidal momentum mode or CoM trajectory mode. To this end, the general form of a constrained least square optimization is given below:

$$\chi^* = \arg \min_{\forall \chi \in R^{2(n+6)}} ||\mathbf{C}_{ls}\chi - \mathbf{d}_{ls}||^2 \tag{3.20}$$

$$s/t \quad \mathbf{A}_{eq}\chi = \mathbf{b}_{eq} \tag{3.21}$$

$$\mathbf{A}_{ineq}\chi \leq \mathbf{b}_{ineq}, \tag{3.22}$$

where χ is the optimization variable represents the joint velocities and accelerations $[\dot{\mathbf{q}}^\intercal\ \ddot{\mathbf{q}}^\intercal]^\intercal$. The joint displacements are obtained per sample as $\mathbf{q}_k = \mathbf{q}_{k-1} + \dot{\mathbf{q}}_k t_s + 0.5\ddot{\mathbf{q}}_k t_s^2$, where t_s is the sampling period, and k is the sampling index. The vector $\mathbf{d}_{ls}$ includes the command variables. Equations (3.21)-(3.22) represents the equality and inequality constraints.The matrices $\mathbf{C}_{ls}$, $\mathbf{A}_{eq}$, $\mathbf{d}_{ls}$, and $\mathbf{b}_{eq}$ are formulated as block matrices:

$$\mathbf{C}_{ls} = \begin{bmatrix} \overline{\mathbf{C}} & \mathbf{0} \\ \dot{\overline{\mathbf{C}}} & \overline{\mathbf{C}} \end{bmatrix},\ \mathbf{A}_{eq} = \begin{bmatrix} \overline{\mathbf{A}} & \mathbf{0} \\ \dot{\overline{\mathbf{A}}} & \overline{\mathbf{A}} \end{bmatrix}, \mathbf{d}_{ls} = \begin{bmatrix} \overline{\mathbf{d}} \\ \dot{\overline{\mathbf{d}}} \end{bmatrix}, \mathbf{b}_{eq} = \begin{bmatrix} \overline{\mathbf{b}} \\ \dot{\overline{\mathbf{b}}} \end{bmatrix}. \tag{3.23}$$

In (3.23), the submatrices, and subvectors $(\overline{\mathbf{C}}, \overline{\mathbf{A}}, \overline{\mathbf{d}}, \overline{\mathbf{b}})$ are assigned based on the configuration: i) CoM trajectory mode, ii) centroidal momentum mode.

CoM Trajectory-based PST Optimization: In this mode the submatrices $\overline{\mathbf{C}}$ and $\overline{\mathbf{A}}$ are defined as $\overline{\mathbf{C}} = \mathbf{J}_p$ and $\overline{\mathbf{A}} = \mathbf{J}_{p-1}$ in (3.23). Task space command variables (${}^{cmd}\mathbf{p}$) can be expressed as below:

$${}^{cmd}\mathbf{p} = [{}^{cmd}\mathbf{p}_T^\intercal\ {}^{cmd}\mathbf{p}_{torso}^\intercal\ {}^{cmd}\mathbf{p}_{com}^\intercal]^\intercal \tag{3.24}$$

in which the task space commands ${}^{cmd}\mathbf{p}_{com}$, ${}^{cmd}\mathbf{p}_{torso}$, and ${}^{cmd}\mathbf{p}_T$ stand for CoM position, torso orientation, feet position, and orientation, respectively. The submatrices $\overline{\mathbf{d}}$ and $\overline{\mathbf{b}}$ are defined as $\overline{\mathbf{d}} = {}^{cmd}\dot{\mathbf{p}}_p$ and $\overline{\mathbf{b}} = {}^{cmd}\dot{\mathbf{p}}_{p-1}$.

Centroidal Momentum-based PST Optimization: In this mode, $\mathbf{J}_{com}$ and $\mathbf{J}_{torso}$ are replaced with the centroidal momentum matrix $\overline{\mathbf{A}}_g$ defined in (3.17). The feet Jacobian matrix $\mathbf{J}_T$, and $\mathbf{A}_g$ are stored in the matrix $\mathbf{\Gamma}$ such that $\mathbf{\Gamma} = [\mathbf{J}_T^\intercal\ \mathbf{A}_g^\intercal]^\intercal$. Task space command variables (${}^{cmd}\mathbf{p}$) in this mode can be expressed as in the following:

$${}^{cmd}\dot{\mathbf{p}} = [{}^{cmd}\dot{\mathbf{p}}_T^\intercal\ {}^{cmd}\mathbf{h}_g^\intercal]^\intercal \tag{3.25}$$

where task space commands ${}^{cmd}\dot{\mathbf{p}}_T$ and ${}^{cmd}\mathbf{h}_g$ stand for feet spatial velocity, and centroidal momentum, respectively. Accordingly, the submatrices in (3.23) for this framework are defined as $\overline{\mathbf{C}} = \mathbf{\Gamma}_p$, $\overline{\mathbf{A}} = \mathbf{\Gamma}_{p-1}$, $\overline{\mathbf{d}} = {}^{cmd}\dot{\mathbf{p}}_p$, and $\overline{\mathbf{b}} = {}^{cmd}\dot{\mathbf{p}}_{p-1}$.

3.3.3 Contact Force Constraints

The contact forces are imposed on the proposed PST optimization block to define feasible contact force boundaries via the linear inequality constraint (3.22). To extract a linear relation between contact force components and joint variables, the decomposed floating base dynamics given in (3.14) is used.

Unilateral Constraint: The purpose of this constraint is to admit only repulsive forces. This condition ensures that the vertical component of the contact force meets the following condition:

$$\lambda_{c,z} \geq 0. \tag{3.26}$$

Friction Constraint: Coulomb friction model is used to constraint the tangential contact forces $(\lambda_{c,x}, \lambda_{c,y})$ relative to $\lambda_{c,z}$ to avoid support foot slippage. The friction pyramid is used to approximate the friction cone used to describe the Coulomb friction model:

$$|\lambda_{c,x}|, |\lambda_{c,y}| \leq \mu_s \lambda_{c,z}. \tag{3.27}$$

where static friction coefficient is denoted by μ_s.

Center of Pressure Constraint: To ensure dynamic balance, the horizontal contact moments $(m_{c,x}, m_{c,y})$ are constrained relative to $\lambda_{c,z}$ by considering the support polygon. The feasible region for the horizontal contact moments is defined by the inequalities below:

$$|m_{c,x}| < \frac{w_s}{2}\lambda_{c,z}, \quad |m_{c,y}| < \tfrac{L_s}{2}\lambda_{c,z}, \tag{3.28}$$

where w_s and L_s represent the lateral and longitudinal dimensions of the support polygon, respectively.

3.4 Generation of Task Space Trajectories

This section explains the methods used to design the Cartesian task trajectories.

3.4.1 ZMP based trajectory

The CoM trajectory was analytically synthesized based on the ZMP concept, which can be expressed as follows when considering the classical point mass model with a massless rod and a foot with a finite size [37].

$$y_{zmp} = y_{com} - \frac{\ddot{y}_{com}}{g} z_{com}, \quad x_{zmp} = x_{com} - \frac{\ddot{x}_{com}}{g} z_{com} \tag{3.29}$$

In (3.29), the CoM height is kept constant: z_{com}. The Cartesian coordinates of the CoM are denoted as $(x_{com}, y_{com}, z_{com})$, while the Cartesian coordinates of the ZMP are denoted as $(x_{zmp}, y_{zmp}, 0)$. To synthesize CoM trajectories for given ZMP inputs, the analytical trajectory generator presented in [39] is employed due to its favorable features, such as smooth and continuous CoM trajectories for all position, velocity, and acceleration levels and experimental applicability in real-time.

Single Support Phase: Fig. 30 displays the CoM trajectory in blue and the ZMP trajectory in red for the world frame for a single walking period. During the left Single Support Phase (SSP), the footprints on the ground in Fig. 30 are given by the supporting polygon $(ABCDA)$, while the right foot swings in the air. In this phase, the ZMP trajectory is fixed at the point $(1')$, while the CoM position is moving from point (1) to (2).

During SSP, the ZMP reference input stays constant; $x_{zmp} = p_x$ and $y_{zmp} = p_y$, where p_x and p_y are constants. The CoM trajectory can be obtained by solving (3.29).

$$x_{com} = p_x + \frac{\dot{x}_{so}}{\omega} sinh(\omega t') + (x_{so} - p_x) cosh(\omega t'), \tag{3.30}$$

$$y_{com} = p_y + \frac{\dot{y}_{so}}{\omega} sinh(\omega t') + (y_{so} - p_y) cosh(\omega t'), \tag{3.31}$$

where $\omega = \sqrt{\frac{z_{com}}{g}}$ and $t' = t - t_o$. t represents the time since the walking motion started, and t_o is the initial time at the beginning of the corresponding phase.
$(x_{so}, \dot{x}_{so}, y_{so}, \dot{y}_{so})$ are CoM SSP initial conditions, and g is the gravitational acceleration.

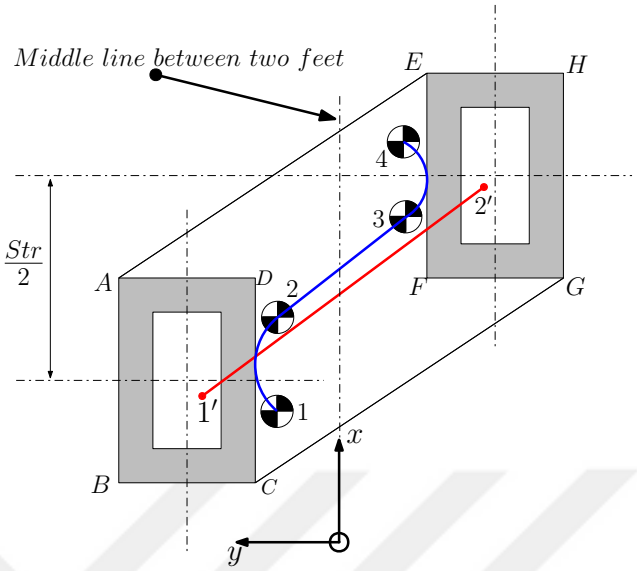

Figure 30: CoM trajectory in the blue line vs. ZMP trajectory in the red line concerning the footprints. Left SSP: CoM moves from 1 to 2, while ZMP stays constant at 1'. DSP: CoM moves from 2 to 3, while ZMP linearly shifts from 1' to 2'. Right SSP: CoM moves from 3 to 4 while ZMP stays constant at 2'.

Double Support Phase: In Double Support Phase (DSP), both feet are on the ground; hence, the supporting polygon is $(ABCGHEA)$. The ZMP is moved from $(1')$ to $(2')$ linearly, while the CoM position is transferred from (2) to (3). The ZMP trajectory used in this phase shifts it from one foot to another: $x_{zmp} = p_x + k_x t'$ and $y_{zmp} = p_y + k_y t'$. The parameters (k_x, k_y) represent the time slope of linear ZMP trajectories. In this case, the analytical solution of the ZMP equations is as below:

$$x_{com} = p_x + k_x t' + \frac{\dot{x}_{do} - k_x}{\omega} sinh(\omega t') + (x_{do} - p_x) cosh(\omega t') \tag{3.32}$$

$$y_{com} = p_y + k_y t' + \frac{(\dot{y}_{do} - k_y)}{\omega} sinh(\omega t') + (y_{do} - p_y) cosh(\omega t') \tag{3.33}$$

where $(x_{do}, y_{do}, \dot{x}_{do}, \dot{y}_{do})$ are CoM DSP initial conditions, i.e., previous SSP terminal conditions.

Boundary Conditions and ZMP References: The boundary conditions and ZMP references were obtained in a way to guarantee the continuity of CoM acceleration regardless of the gait phases [39]. Figs. 31 and 32 display the x-axis and y-axis position, velocity, and

acceleration of CoM trajectories for 10 seconds. The variables $t_{ss}, t_{ms}, t_d, t_{md}$ are the SSP, middle SSP, DSP, and middle DSP periods, respectively. Observing Figs. 31 and 32, one may observe the following features: i) In SSP, the y-axis CoM trajectory is symmetrical, i.e., it bounces back to the initial position, ii) because of this reason, the CoM velocity is zero in the middle of SSP, iii) y-axis CoM position and acceleration crosses over zero in the middle of SSP, iv) an average forward velocity term, υ_m can be used to define terminal x-axis position, v) x-axis CoM acceleration crosses over zero in the middle of DSP. Using these symmetry conditions, one can compute the necessary parameters in eqs. (3.30)-(3.33) as follows:

$$x_{ts} = v_m t_{ss} + x_{so} \tag{3.34}$$

$$p_x = \frac{x_{ts} sinh(\omega t_{ms}) + x_{t_{sp}} sinh(\omega(t_s - t_{ms}))}{sinh(\omega(t_{sp} - t_{ms}) + sinh(\omega t_{ms})} \tag{3.35}$$

$$\dot{x}_{so} = (p_x - x_{so})\omega coth(\omega t_{ms}) \tag{3.36}$$

$$k_x = \dot{x}_{do} + (x_{do} - p_x)\omega coth(\omega t_{md}) \tag{3.37}$$

$$Str = 2k_x t_d \tag{3.38}$$

$$\dot{y}_{so} = \omega(p_y - y_{so}) tanh(\omega t_{ms}) \tag{3.39}$$

$$\dot{y}_{do} = (p_y - y_{do})\omega coth(\frac{\omega t_d}{2}) + k_y \tag{3.40}$$

$$k_y = \frac{-2p_y}{t_d} \tag{3.41}$$

$$p_y = \frac{y_{do}\omega coth(\omega t_{md}) + \omega tanh(\omega(t_{ms}))}{\omega coth(\omega t_{md}) + \omega tanh(\omega(t_{ms})) - \frac{2}{t_d}}. \tag{3.42}$$

In (3.34)-(3.42), t_{sp} and t_d periods are determined beforehand. Initial values of x_{so} and y_{so} are assigned based on the geometric properties and initial joint configuration of the robot. Furthermore, the following equalities are required to ensure a seamless connection between an SSP and a DSP: $x_{do} = x_{ts}$, $\dot{x}_{do} = \dot{x}_{so}$, $\dot{y}_{do} = -\dot{y}_{so}$, and $y_{do} = y_{so}$, where x_{ts} is the CoM forward displacement during SSP.

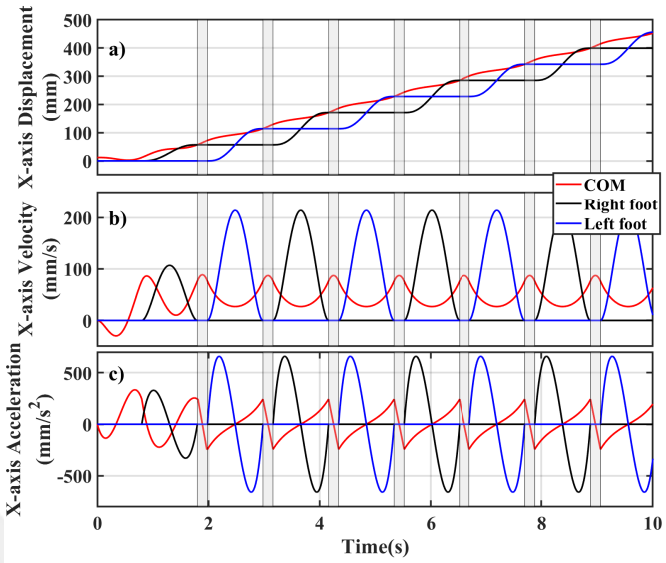

Figure 31: x-axis CoM trajectory and feet trajectories concerning the global frame. Gray regions represent double support phases.

3.4.2 Swing Foot Trajectories

A set of polynomial functions was used to plan the swing foot trajectories in the task space to guarantee a smooth transition between phases. For forward walking motion, it is just required to trigger the swinging foot motion in forward x_f and vertical z_f directions. A fifth-order polynomial is chosen to describe the motion in x_f direction. Thus, the initial and final positions, velocities, and accelerations in x_f direction are tunable. A sixth-order polynomial is chosen to design the trajectory in z_f direction, such that the initial and final positions, velocities, and accelerations are uniquely determined, as well as the foot apex height.

Quintic Trajectory: The fifth order trajectory can simply describe the relationship between x-axis swing foot displacement x_f and time during SSP as follows:

$$x_f = a_{x,0} + a_{x,1}t' + a_{x,2}t'^2 + a_{x,3}t'^3 + a_{x,4}t'^4 + a_{x,5}t'^5. \tag{3.43}$$

To find the set of coefficients in (3.43), the terminal conditions of the SSP are used such that:

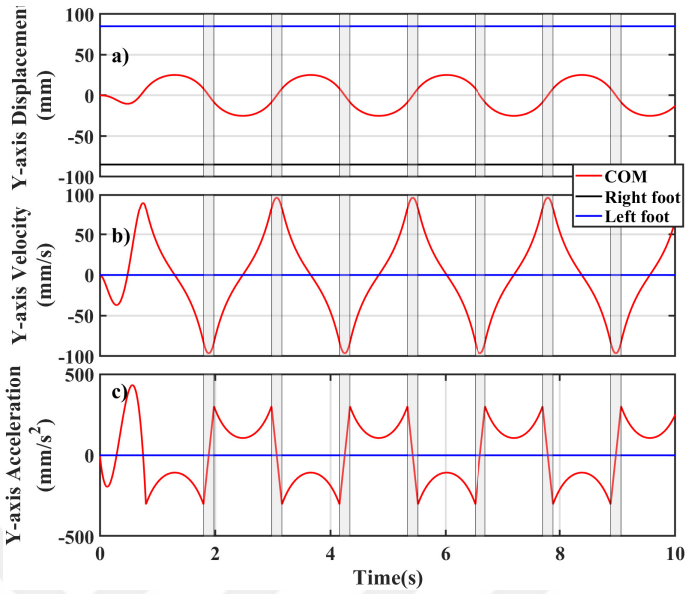

Figure 32: y-axis CoM trajectory and feet trajectories concerning the global frame. Gray regions represent double support phases.

$$\begin{bmatrix} x_{f,0} \\ \dot{x}_{f,0} \\ \ddot{x}_{f,0} \\ x_{f,f} \\ \dot{x}_{f,f} \\ \ddot{x}_{f,f} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & t_{ss} & t_{ss}^2 & t_{ss}^3 & t_{ss}^4 & t_{ss}^5 \\ 0 & 1 & 2t_{ss} & 3t_{ss}^2 & 4t_{ss}^3 & 5t_{ss}^4 \\ 0 & 0 & 2 & 6t_{ss} & 12t_{ss}^2 & 20t_{ss}^3 \end{bmatrix} \begin{bmatrix} a_{x,0} \\ a_{x,1} \\ a_{x,2} \\ a_{x,3} \\ a_{x,4} \\ a_{x,5} \end{bmatrix}. \tag{3.44}$$

Sixth Order Trajectory: Similar to x_f the vertical displacement z_f is designed using sixth-order trajectory to describe the relation with time as follows:

$$z_f = a_{z,0} + a_{z,1}t' + a_{z,2}t'^2 + a_{z,3}t'^3 + a_{z,4}t'^4 + a_{z,5}t'^5 + a_{z,6}t'^6. \tag{3.45}$$

To obtain equation (3.45) coefficients, the terminal conditions of the SSP are used as follows:

$$\begin{bmatrix} z_{f,0} \\ \dot{z}_{f,0} \\ \ddot{z}_{f,0} \\ z_{f,f} \\ \dot{z}_{f,f} \\ \ddot{z}_{f,f} \\ z_{f,p} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & t_{ss} & t_{ss}^2 & t_{ss}^3 & t_{ss}^4 & t_{ss}^5 & t_{ss}^6 \\ 0 & 1 & 2t_{ss} & 3t_{ss}^2 & 4t_{ss}^3 & 5t_{ss}^4 & 6t_{ss}^5 \\ 0 & 0 & 2 & 6t_{ss} & 12t_{ss}^2 & 20t_{ss}^3 & 30t_{ss}^4 \\ 1 & t_{ms} & t_{ms}^2 & t_{ms}^3 & t_{ms}^4 & t_{ms}^5 & t_{ms}^6 \end{bmatrix} \begin{bmatrix} a_{z,0} \\ a_{z,1} \\ a_{z,2} \\ a_{z,3} \\ a_{z,4} \\ a_{z,5} \\ a_{z,6} \end{bmatrix}. \tag{3.46}$$

3.5 Floating Base Inverse Dynamics

The solution of floating base inverse dynamics is non-trivial due to the underactuation of the floating base in addition to constraints switching behavior. Generally speaking, inverse dynamics provides a proactive control signal that enforces the joint level controllers to improve joints trajectories tracking. Traditional high gain error feedback controllers are impractical to achieve these goals, as high tracking performance is directly related to high gains, hence a little compliance and a high noise effect. Two approaches are discussed here to resolve Floating Base Inverse Dynamics (FBID), the first approach exploits the **QR decomposition**, and the second approach relies on the **direct decomposition** of the floating base model (3.11).

3.5.1 QR decomposition

This method is used to decompose the constraint Jacobian matrix into two matrices as follows [67]:

$$qr(\mathbf{J}_c(q)^\intercal) = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}, \tag{3.47}$$

where, $qr(\bullet)$ is a function that is used to decompose the input matrix into $\mathbf{Q} \in \mathbb{R}^{(6+n_r)\times 6n_c}$ orthogonal matrix, and an upper triangle matrix $\mathbf{R} \in \mathbb{R}^{6n_c \times 6n_c}$. By inserting (3.47) in (3.11) the following equations can be obtained.

$$\mathbf{S}_c\mathbf{Q}^\intercal(\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})) = \mathbf{S}_c\mathbf{Q}^\intercal\mathbf{S}^\intercal\boldsymbol{\tau} + \mathbf{R}\boldsymbol{\lambda}_c, \tag{3.48}$$

$$\mathbf{S}_n\mathbf{Q}^\intercal(\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})) = \mathbf{S}_n\mathbf{Q}^\intercal\mathbf{S}^\intercal\boldsymbol{\tau}_{ID}, \tag{3.49}$$

where, $\mathbf{S}_c = \begin{bmatrix}\mathbf{1}_{6n_c} & \mathbf{0}_{6n_c\times(6+n_r-6n_c)}\end{bmatrix}$, and $\mathbf{S}_n = \begin{bmatrix}\mathbf{0}_{(6+n_r-6n_c)\times 6n_c} & \mathbf{1}_{(6+n_r-6n_c)}\end{bmatrix}$ are the constraint and non-constraint equations selection matrices, respectively. The parameter $\boldsymbol{\tau}_{ID}$ denotes the inverse dynamic solution. It is obvious that equation (3.49) is independent of contact forces, hence a feasible FBID solution can be obtained. A pseudo-inverse can be used to resolve (3.49) such that:

$$(\mathbf{S}_n\mathbf{Q}^\intercal\mathbf{S}^\intercal)^\dagger\mathbf{S_n}\mathbf{Q}^\intercal(\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})) = \boldsymbol{\tau}_{ID}, \tag{3.50}$$

In the case of $n_c = 1$, the pseudo inversion can be replaced by direct inversion. If $n_c > 1$, then the constraints are redundant and the pseudo inversion can be used to obtain the minimal torque norm vector that can achieve the desired motion.

3.5.2 Direct Decomposition

This method provides two advantages over the **QR decomposition** approach. It can exploit the internal structure of $\mathbf{J}_c(q)$ in (3.13) for the case of SSP to compute the direct inversion. Moreover, it can directly manipulate contact force optimization in case of redundancy. The floating base model in (3.11) can be decomposed as follows:

$$[\mathbf{I}_{fb}\ \overline{\mathbf{F}}]\ddot{\mathbf{q}} + \mathbf{C_{fb}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G_{fb}}(\mathbf{q}) = \mathbf{S}_{fb}\mathbf{J}_c(q)^\intercal\boldsymbol{\lambda}_c \tag{3.51}$$

$$[\overline{\mathbf{F}}^\intercal\ \mathbf{H}_r]\ddot{\mathbf{q}} + \mathbf{C_r}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G_r}(\mathbf{q}) = \boldsymbol{\tau}_{ID} + \mathbf{S}\mathbf{J}_c(q)^\intercal\boldsymbol{\lambda}_c \tag{3.52}$$

where, $\mathbf{C_r} = \mathbf{S}\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, and $\mathbf{G_r} = \mathbf{S}\mathbf{G}_(\mathbf{q})$. The FBID resolution in this method is carried out in two steps. The first step is the computation of $\boldsymbol{\lambda}_c$ using (3.51). The second step is using the obtained $\boldsymbol{\lambda}_c$ to compute $\boldsymbol{\tau}_{ID}$ via (3.52). In case of SSP, $(\mathbf{S}_{fb}\mathbf{J}_c(q)^\intercal)^{-1} = {}^{lf}\mathbf{X}_0^*$

or $(\mathbf{S}_{fb}\mathbf{J}_c(q)^\intercal)^{-1} = {}^{rf}\mathbf{X}_0^*$. Hence, $\boldsymbol{\lambda}_c$ and $\boldsymbol{\tau}_{ID}$ are readily obtained without any matrix inversion requirement. For DSP, $(\mathbf{S}_{fb}\mathbf{J}_c(q)^\intercal)^\dagger = [{}^0\mathbf{X}_{rf}^* \; {}^0\mathbf{X}_{lf}^*]^\dagger$ as there is no direct inversion due to redundancy. Equation (3.51) can be solved directly using pseudo inversion to obtain the minimal possible solution of $\boldsymbol{\lambda}_c$ or by employing an optimization algorithm to compute the optimal $\boldsymbol{\lambda}_c^*$ concerning a set pre-specified constraints, e.g. friction cone, zmp, unilateral constraint, minimal jerk.

3.6 Experimental Verification

A set of locomotion activity experiments are conducted using a Co-Ex setup. In the first version of the experiments, a simple PD controller was used for the joint level.

3.6.1 Squat

The first activity is doing a squatting motion. The robot is initialized in a balanced non-singular posture. The robot then, performs a squatting motion with a trajectory given as:

$$z_{fb} = 0.0175cos(0.4\pi t) \tag{3.53}$$

where, z_{fb} is the floating base vertical displacement. The peak-to-peak value is 3.5 *cm* and the frequency is 0.2 *Hz*. The vertical displacement response, namely z_{fb}^{act} is shown in a solid black line in Fig. 33 as compared to the referential signal, which is denoted as z_b^{ref} and shown in a solid red line. Fig. 33 shows a good position tracking performance. The two signals are in phase, with a peak error $e_{peak} = 2.61$ *mm*.

The computation effort is evaluated based on the time consumed by the CPU to develop the control signal per sample. Fig. 34 shows the computation time t_c. The CPU could preserve a 0.5 *ms* as an average value over the whole experiment's run. Fig. 35 shows the ZMP responses in x and y direction in the blue solid line and the support polygon boundaries SP_b in solid red lines. Fig. 35 proves the balance of the proposed solution as the ZMP variations are displaced away from SP_b.

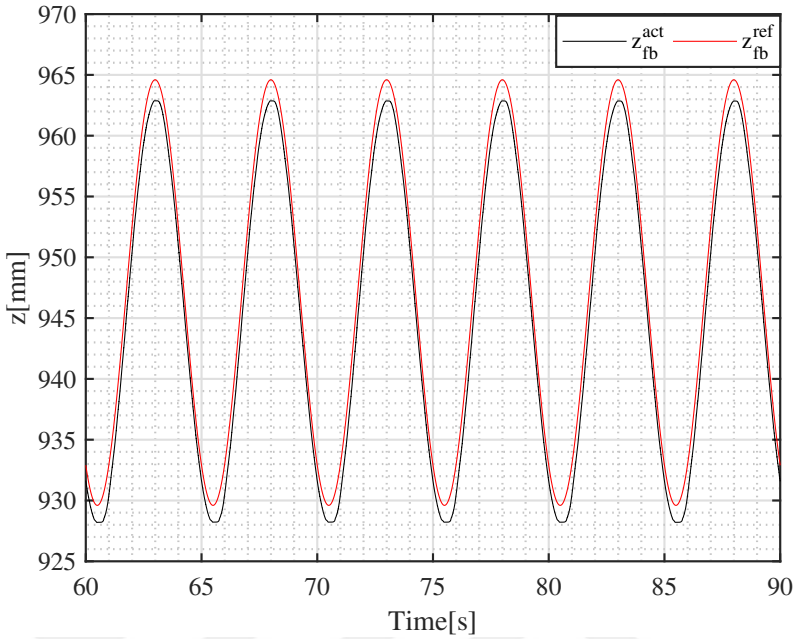

Figure 33: Floating base coordinate frame vertical position response z_{fb}^{act} and the reference signal z_{fb}^{ref}.

Ankle Motor Mechanism Evaluation: The squatting experiment is used to evaluate two mechanisms' performance based on the demanded torque by the knee joint. The two leg mechanisms are shown in Fig. 36. The first mechanism M_1 is shown in Fig. 36.a in which the A DP/F motor is installed on the thigh near to hip joint and connected to the ankle joint via a double four bar mechanism. The basic motivation for M_1 is to collect the masses around the floating base, hence allowing improved maneuverability for the swinging leg. The main disadvantage of M_1 is the mechanical coupling between the ankle and knee motor, which lead to nonlinear kinematic relation between the robot ankle joint and the knee and ankle motors. The reader is referred to appendix D for further details about the kinematic resolution of M_1.

As a result of kinematic coupling and the robot motion in a knee-bent form to avoid singularity, the knee joint demanded torque is increased. An alternative approach is to attach the ankle motor to the shank, as shown in Fig. 36.b. The motor is attached near the knee joint and transmits its motion via a four bar mechanism. The second leg mechanism

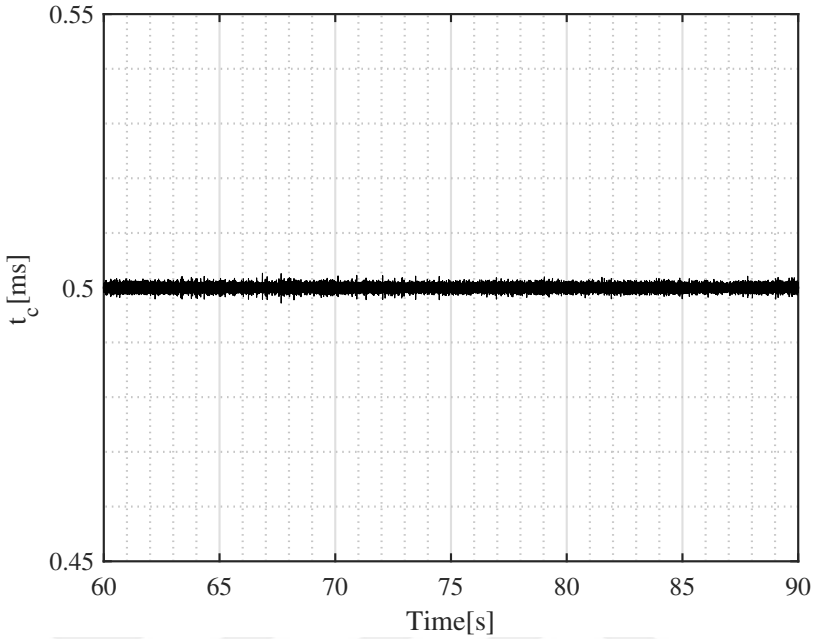

Figure 34: CPU computation time per sample.

is denoted by M_2. A comparative result concerning the knee motor torque τ_k is shown in Fig. 37. Mechanism M_1 torque is shown in a solid red line, while M_2 is shown in a solid black line. The result shows a significant torque reduction in favor of M_2, hence a hardware modification is carried out, and M_2 is used to transmit ankle motor motion.

3.6.2 Sway

A swaying motion is developed using the ZMP algorithm discussed in section 3.4.1. The motion is generated by solely triggering the lateral motion. Neither the foot clearance nor the forward velocity was excited. Fig. 38 shows the actual lateral floating base displacement y_{fb}^{act} in a solid black line and the corresponding reference signal y_{fb}^{ref} in a solid red line. Fig. 38 proves good tracking performance, as the response is in phase with the reference, moreover, the peak error is $e_{peak} = 15.5337\ mm$.

Due to the proposed joint configuration, it is impossible to perform either floating base or CoM lateral displacement without having a torso roll $\theta_{x,fb}^{act}$ motion. Fig. 39 shows the torso roll variation while performing the swaying motion. Despite the underactuation and

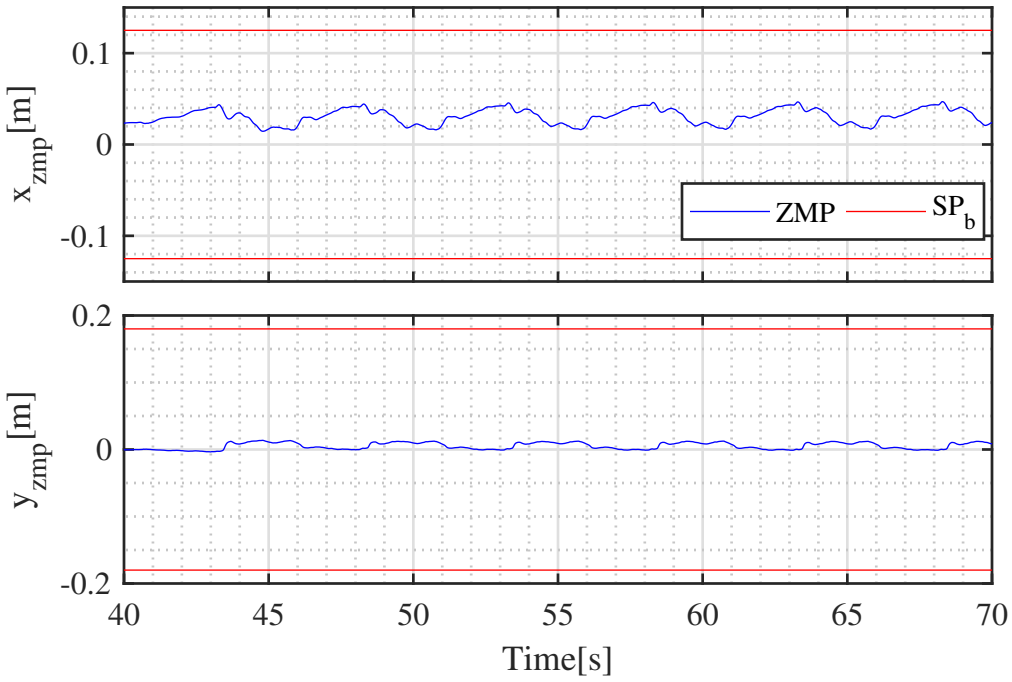

Figure 35: The ZMP response is shown in a solid blue line and the support polygon boundaries (SP_b) are shown in a solid red line.

the relatively high robot inertial parameters, the robot was able to perform the lateral motion with good Cartesian trajectory tracking.

Fig. 40 shows the ZMP variations in both x and y directions in solid blue lines, and the support polygon boundaries in solid red lines.

The Fig. 40 shows the locomotion balance while performing the swaying motion. The Cartesian trajectory generation algorithm and the inverse kinematics solution were resolved using a secondary thread, while the joint level and four bar mechanism kinematics were resolved using the primary thread. The computation time for each thread is shown in Fig. 41. The joint level controller was assigned to be executed in 0.5 *ms* while the secondary thread consumed no more than 7 *ms*, consequently, the secondary thread sampling time is adjusted to 10 *ms*. Fig. 41 reflects the real-time applicability of the proposed algorithms using parallel programming.

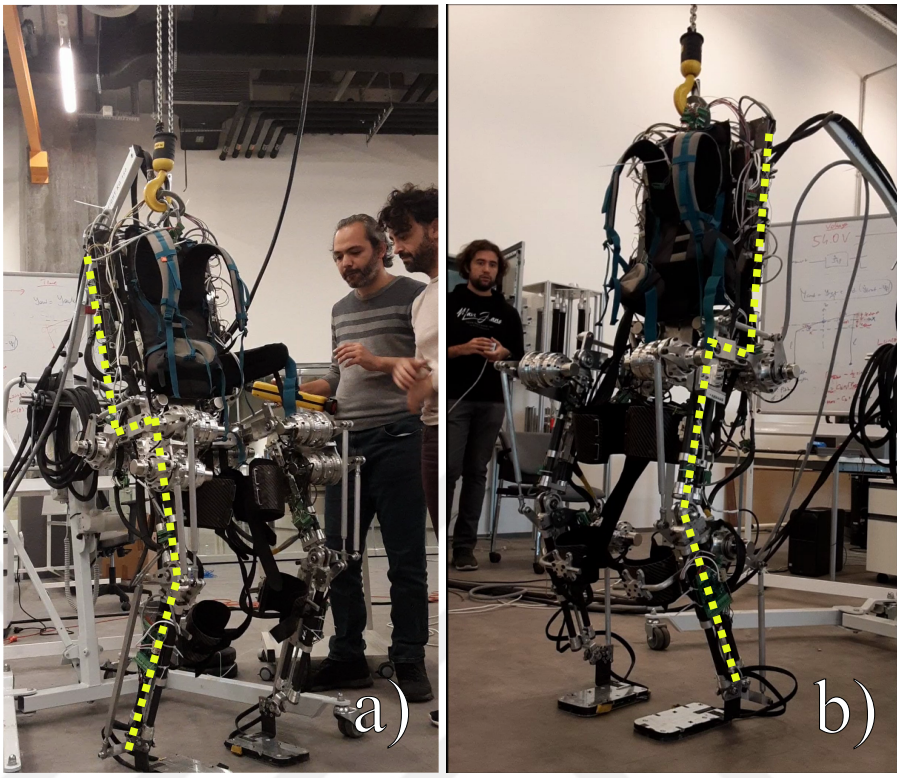

Figure 36: Squatting experiments with 2 distinct leg mechanisms: a) Mechanism $M1$ with ankle D/PF motor is located near the hip joint. b) Mechanism $M2$ with ankle D/PF motor is located on the shank.

3.6.3 Sagittal Walking

Due to hardware limitations, the robot solely performs the walking motion in the sagittal plane. The robot performs the sagittal walking motion in two modes. The first mode is the **Dynamic walk**. It uses subsection 3.4.1 to design the CoM trajectory. The SSP and DSP periods for this mode are 0.8 s and 0.2 s, respectively. The maximum foot clearance is 6 cm, and the average forward velocity is 4 cm/s.

The second mode is the **quasi-static walk**. In this mode, the CoM trajectory is designed to keep its projection on the ground within the support polygon. The mean forward velocity is adjusted to be sufficiently small to avoid excitation of high-order differential terms. The forward velocity is 2 cm/s, and the SSP and DSP periods for this mode are 1 s and 1 s, respectively. The maximum foot clearance is 6 cm. The response of the right leg knee angular displacement is shown in Fig. 42 with a maximum error value of 5^o. The zmp

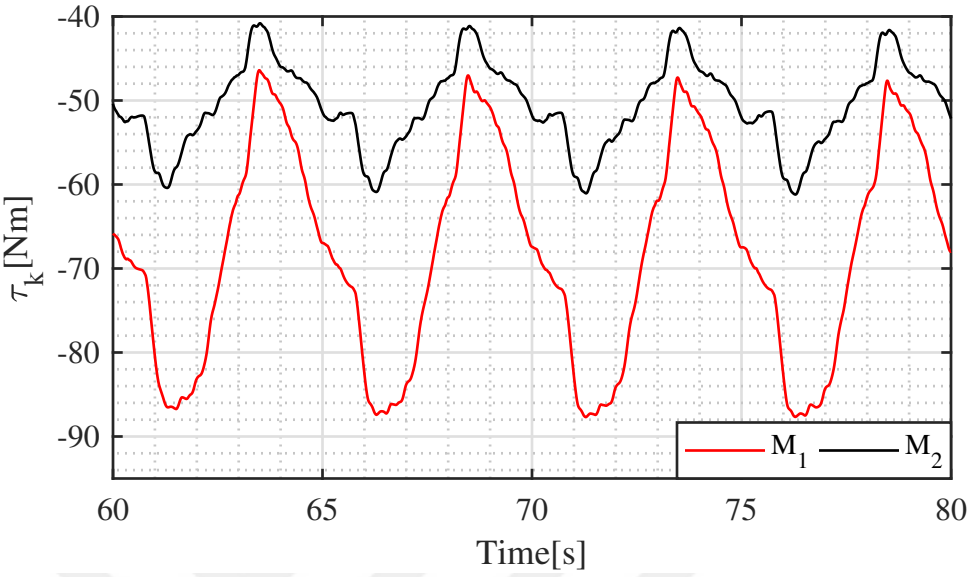

Figure 37: Knee motor torque for $M1$ and M_2.

response is shown in Fig. 43. The results demonstrate the ability of the developed robot to generate feasible walking motion in the sagittal plane.

3.6.4 Push Recovery

A set of perturbations is applied to the Co-Ex robot using the force stick shown in Fig. 44. The directions of the perturbations are described in Fig. 45. To improve push recovery, a simplified orientation control algorithm is developed. Firstly, the floating base orientation velocity error is computed as follows:

$$\boldsymbol{\omega}_{fb}^{error} = \boldsymbol{\omega}_{fb}^{ref} - \boldsymbol{\omega}_{fb}^{act}, \tag{3.54}$$

where $\boldsymbol{\omega}_{fb}^{ref} \in \mathbb{R}^3$ and $\boldsymbol{\omega}_{fb}^{act} \in \mathbb{R}^3$ are the floating base orientation velocity reference and actual values, respectively. $\boldsymbol{\omega}_{fb}^{act}$ is obtained using IMU attached to the robot floating base. Secondly, the floating base velocity error is projected onto the contact frame as follows:

$$\boldsymbol{\lambda}_c^{or} = (\mathbf{S}_{fb}\mathbf{J}_c(q)^\intercal)^\dagger \mathbf{K}_{or}\boldsymbol{\omega}_{fb}^{error}. \tag{3.55}$$

The parameter $\boldsymbol{\lambda}_c^{or}$ represents the projection of $\mathbf{K}_{or}\omega_{fb}^{error}$ onto the contact frame. The symbol $\mathbf{K}_{or} \in \mathbb{R}^{3\times3}$ is a diagonal positive semi-definite gain matrix. Thirdly, the orientation

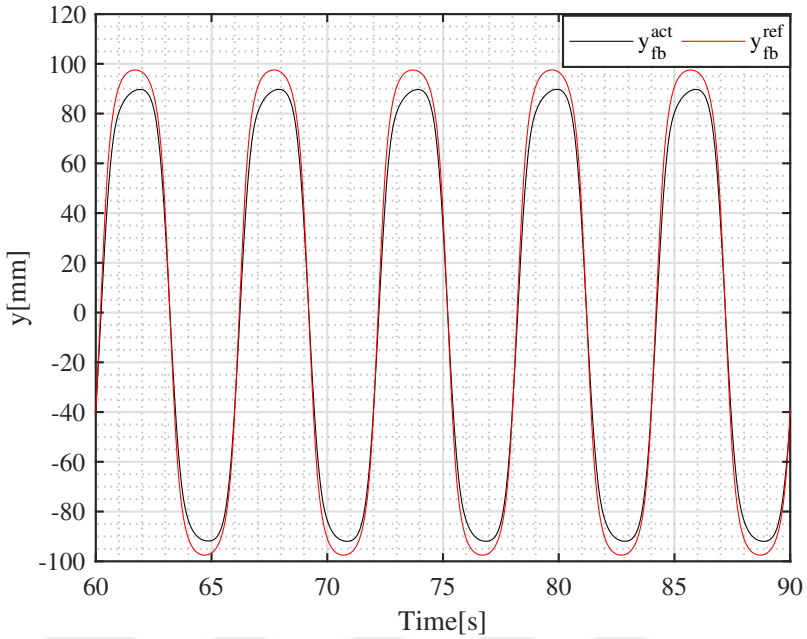

Figure 38: Floating base coordinate frame lateral position response y_{fb}^{act} and the reference signal y_{fb}^{ref}.

error is propagated to the joint torques using the following equation:

$$\boldsymbol{\tau}^{or} = \mathbf{S}\mathbf{J}_c(q)^{\intercal}\boldsymbol{\lambda}_c^{or}. \tag{3.56}$$

Eventually, the torque command signal is updated with $\boldsymbol{\tau}^{or}$ as follows:

$$\boldsymbol{\tau}_{cmd} = \boldsymbol{\tau}^{or} + \boldsymbol{\tau}_{ID} + \boldsymbol{\tau}_s^{ref}. \tag{3.57}$$

The impact of the orientation control algorithm is compared to the basic PD controller under the effect of perturbations in the forward direction. Fig. 46 shows the variation in the floating base pitch angular velocity under the effect of both the basic PD and orientation controller. Fig. 47 shows the zmp response for both the basic PD and orientation controller. The basic PD response is shown with a solid blue line, while the orientation controller response is shown with a solid red line.

Fig. 46.a and 47.a show the behavior of both controllers under an impact force of 164.7 N and impulse momentum of 64.7 Ns for the basic PD controller, and 163.1 N

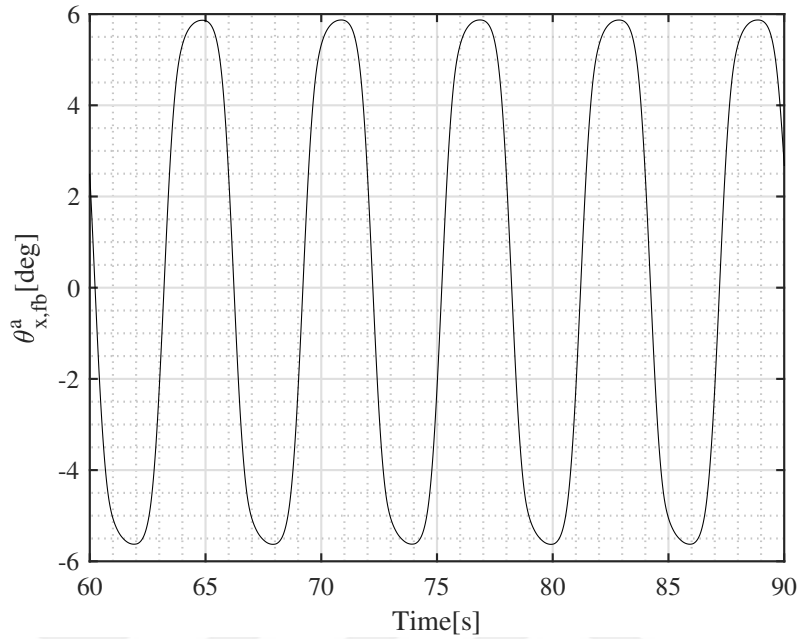

Figure 39: Floating base roll $\theta^{act}_{x,fb}$.

and 74.9 *Ns* for the orientation controller. Fig. 46.b and 47.b show the behavior of both controllers under an impact force of 192.6 *N* and impulse momentum of 110.6 *Ns* for the basic PD controller, and 196.3 *N* and 111.2 *Ns* for the orientation controller. The orientation controller provides improved performance in terms of damping the base pitch velocity orientation and reducing zmp variation.

A series of successive pushes were applied to verify the response of the orientation control performance. Fig. 48 shows the angular displacement and torque responses for the knee joint corresponding to the forward push. The angular displacement converges to the reference value, as well as the torque. Moreover, Fig. 48 shows how the command torque responds to the perturbations. The variation in the floating base pitch axis torso angular velocity is depicted in Fig. 49. The controller was able to recover from the forward pushes with an acceptable convergence rate.

The orientation controller is also verified in the lateral direction. A series of successive lateral pushes were applied to verify the response of the orientation control performance.

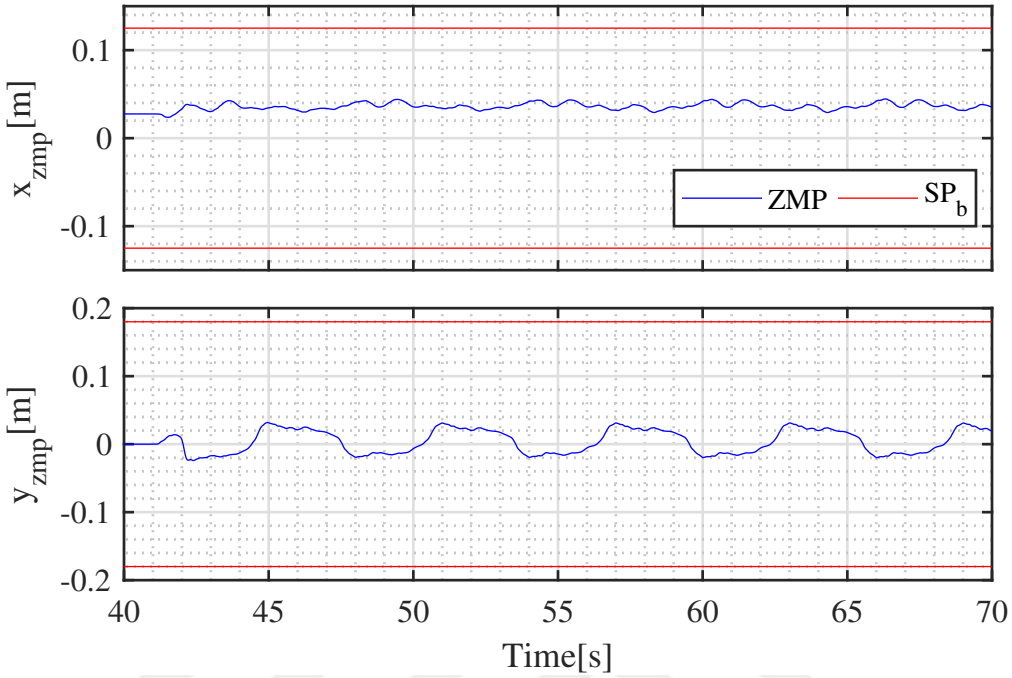

Figure 40: The ZMP response is shown in a solid blue line, and the support polygon boundaries (SP_b) are shown in a solid red line.

Fig. 50 shows the angular displacement and torque responses for the knee joint corresponding to the lateral push. The angular displacement converges to the reference value, as well as the torque. Moreover, Fig. 50 shows how the command torque responds to the perturbations. The variation in the floating base roll axis torso angular velocity is depicted in Fig. 51. The controller was able to recover from the lateral pushes with an acceptable convergence rate.

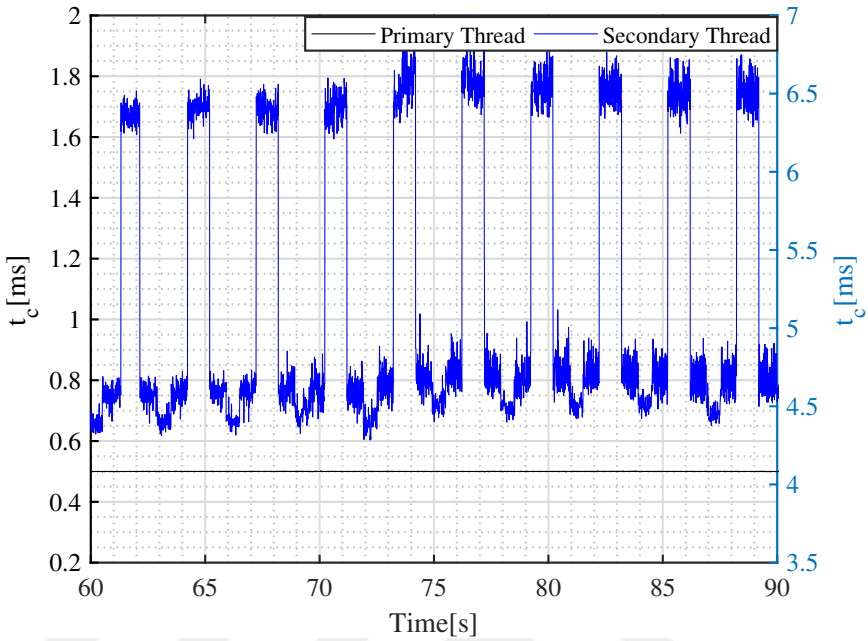

Figure 41: The computation time of the primary and secondary threads per sample.

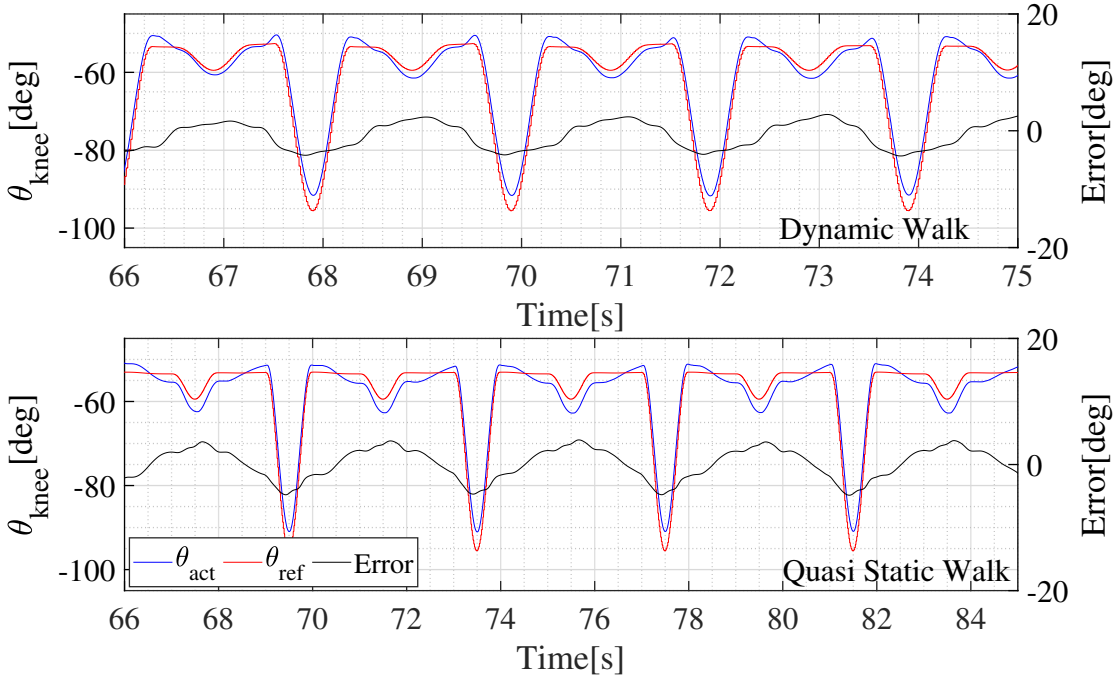

Figure 42: Knee joint angular displacement tracking for both dynamic and quasi-static walk. The subscripts ref and act denotes the reference and actual values, respectively.

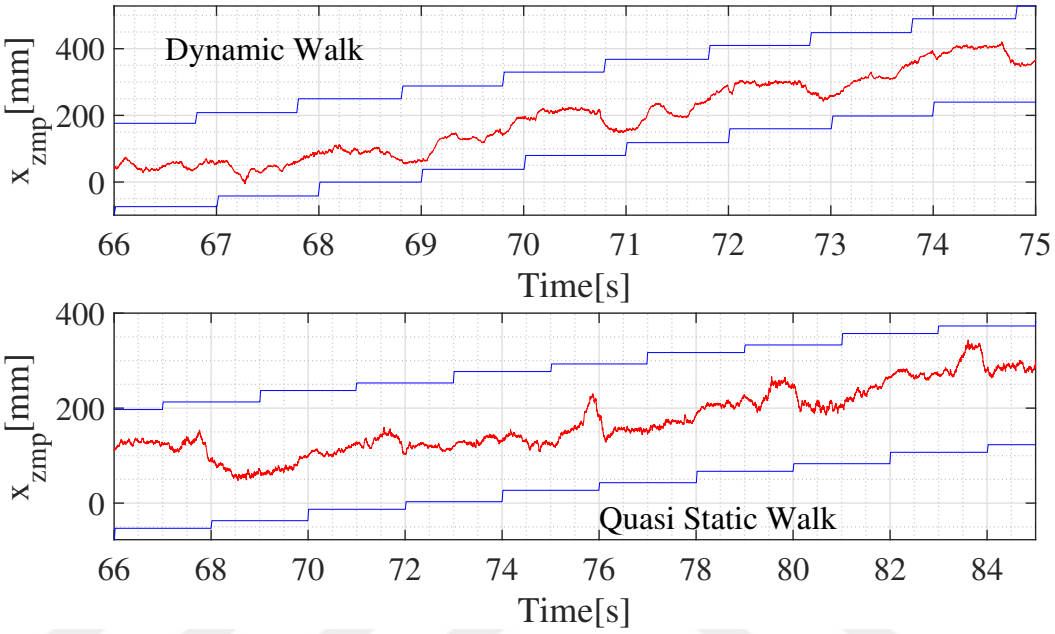

Figure 43: ZMP response for both dynamic and quasi-static walk.

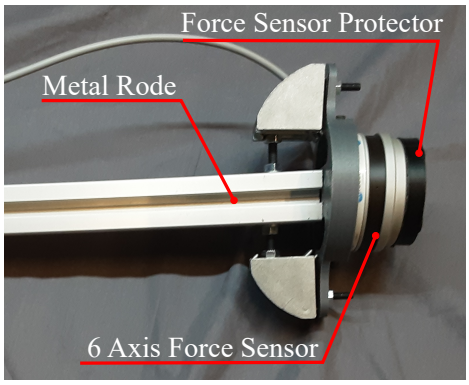

Figure 44: Force stick used to apply perturbations.

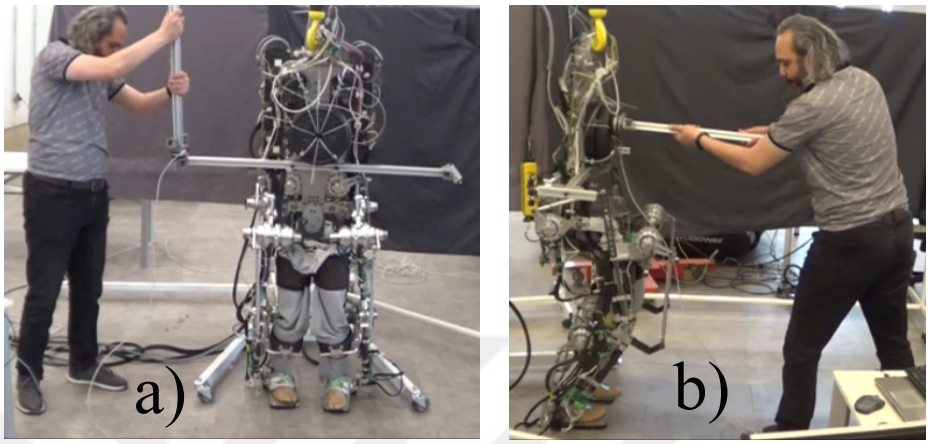

Figure 45: Perturbations direction: a)Lateral, and b)Forward.

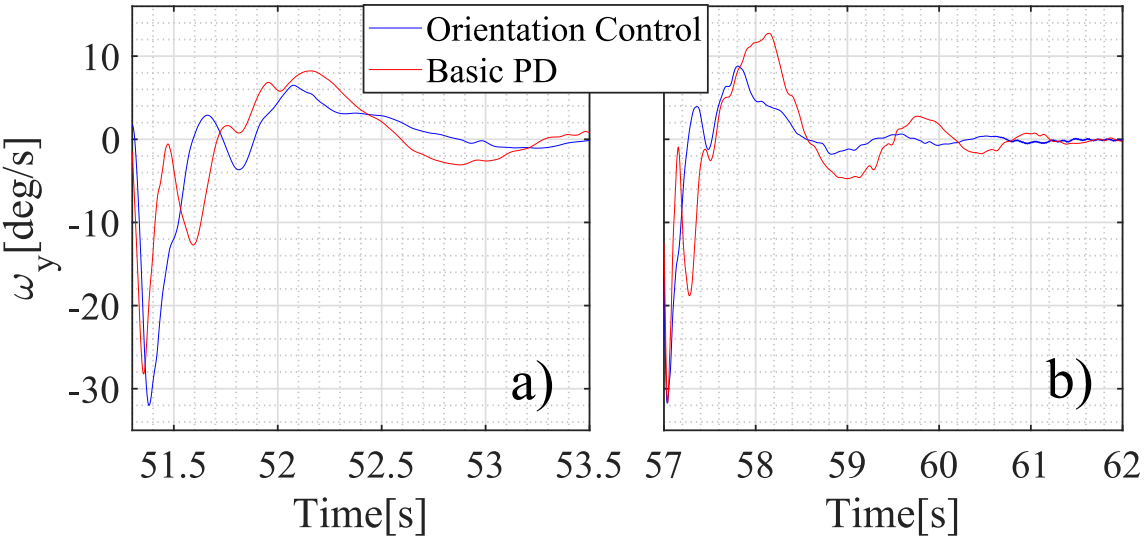

Figure 46: Pitch axis torso angular velocity under the effect of basic PD and Orientation controller.

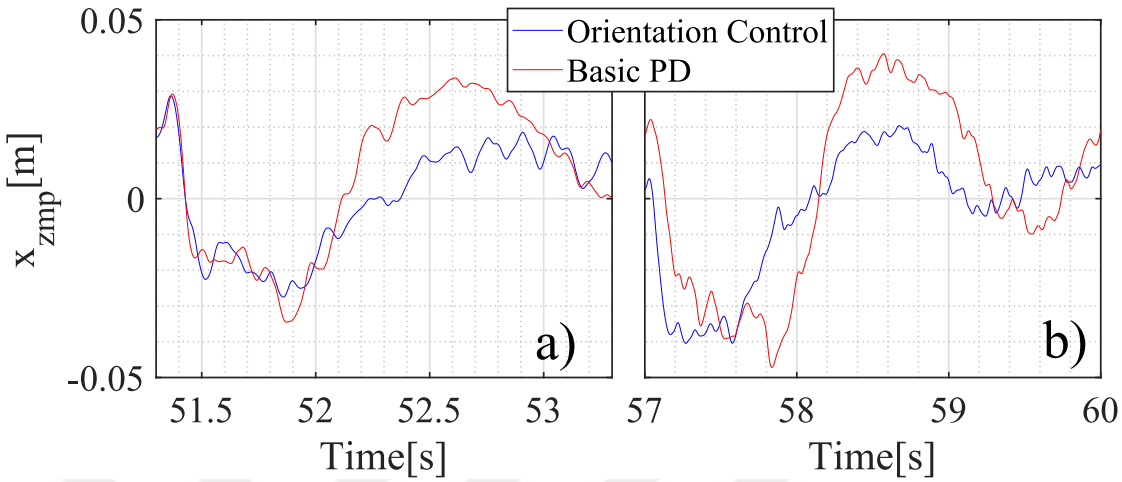

Figure 47: ZMP in *x*-direction under the effect of basic PD and Orientation controller.

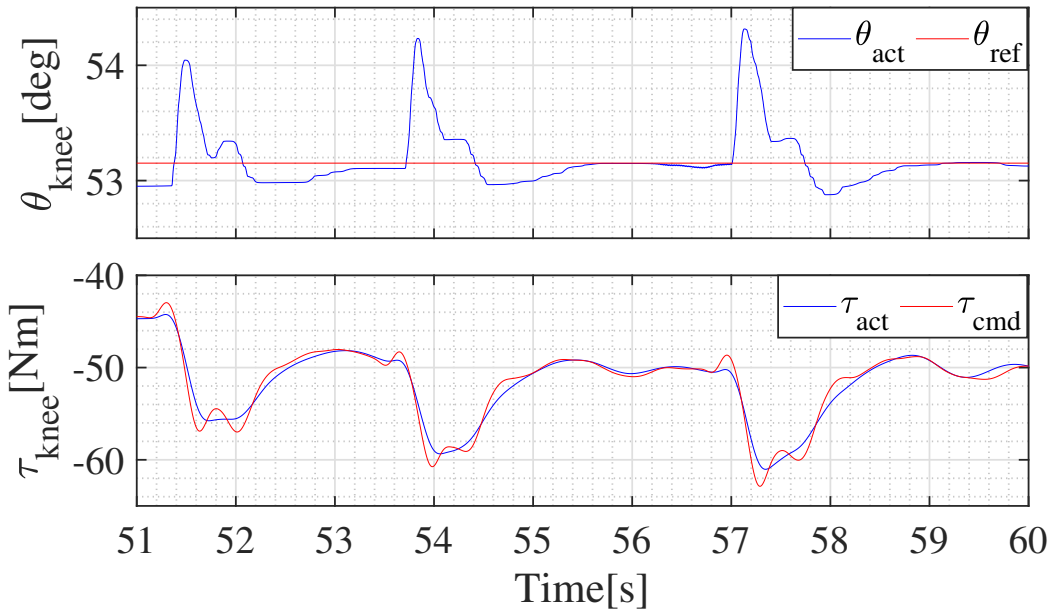

Figure 48: Knee joint angular displacement and torque response corresponding to forward pushes. The subscripts ref, act, and cmd denotes the reference, actual and command values, respectively.

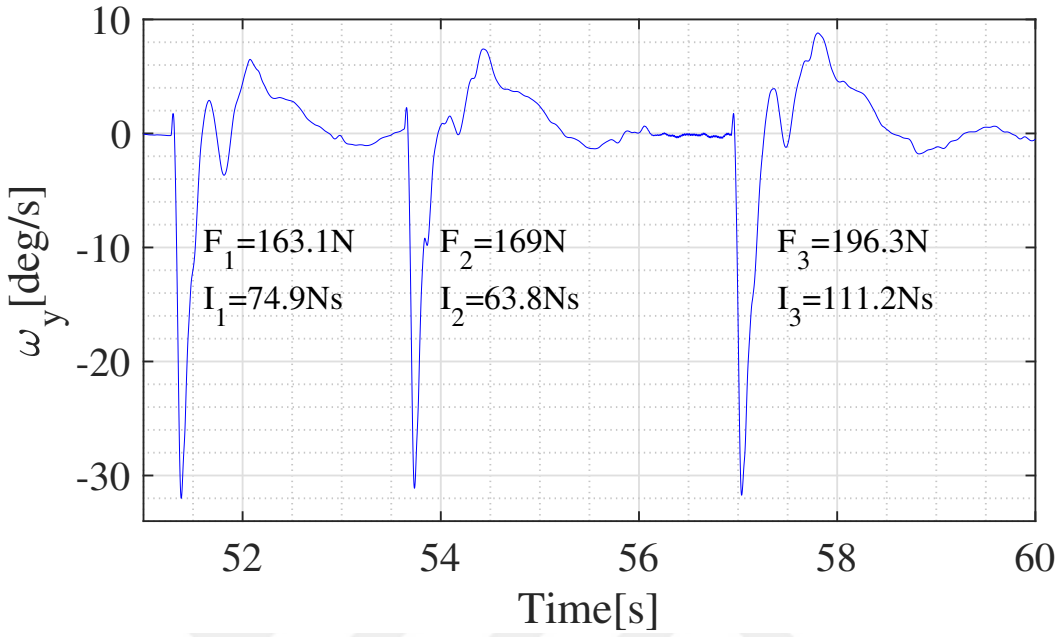

Figure 49: Pitch axis torso angular velocity under the effect of Orientation controller. The perturbation force and impulse momentum values are denoted by F_i and I_i, respectively.

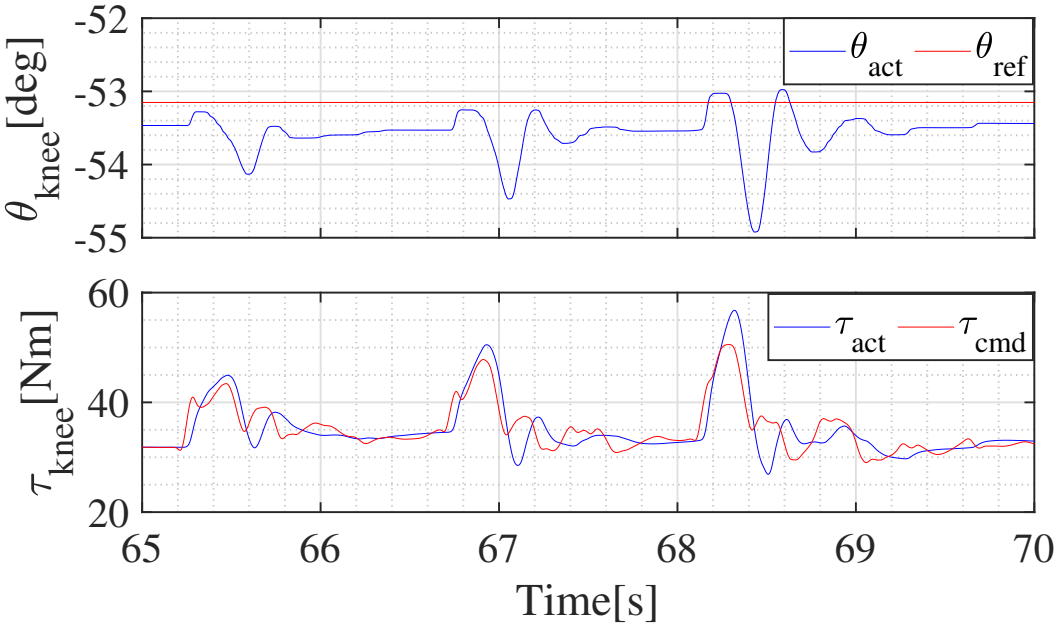

Figure 50: Knee joint angular displacement and torque response corresponding to lateral pushes. The subscripts ref, act, and cmd denotes the reference, actual and command values, respectively.

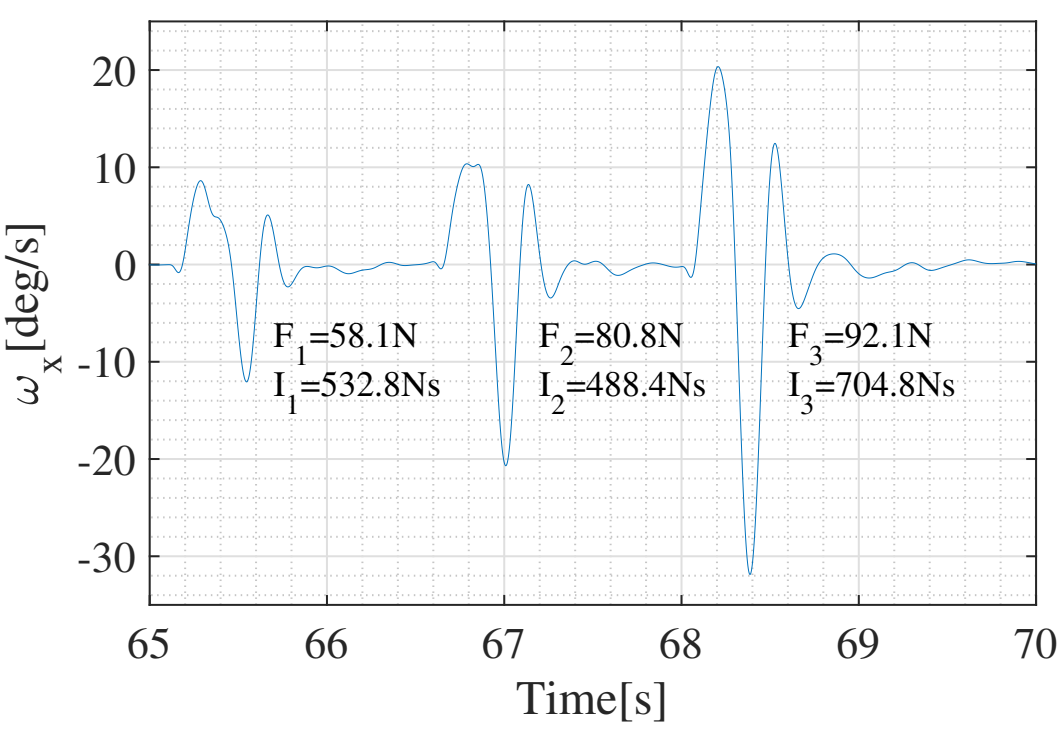

Figure 51: Roll axis torso angular velocity under the effect of Orientation controller. The perturbation force and impulse momentum values are denoted by F_i and I_i, respectively.

CHAPTER IV

LOCOMOTION CONTROL

A set of locomotion controllers are added to the framework for improving its robustness and regulation behavior against uncertainties and underactuation effects. The general control block diagram is displayed in Fig. 52 illustrating the configuration of 3 distinct controllers, a proposed novel controller namely ZIF (ZMP Impedance Feedback, red blocks), CMC (Centroidal Momentum Control, blue blocks), and BAC (Basic Admittance Control, green blocks). The common ground in applying the controller is the previously explained PST (Prioritized Stack of Tasks) optimization block, which can be configured in centroidal momentum mode or CoM trajectory mode; see section 3.3.2.

4.1 Basic Admittance Controller

In this controller, a virtual spring-damper model is introduced between the contact foot and ground to impose a compliance term that acts actively against any unmodeled contact force [46]. In this control, method, the contact force error is computed as follows: $\boldsymbol{\lambda}_{error} = {}^{ref}\boldsymbol{\lambda}_c - {}^{act}\boldsymbol{\lambda}_c$. The contact force references ${}^{ref}\boldsymbol{\lambda}_c$ are computed using the target CoM trajectory. The actual contact force ${}^{act}\boldsymbol{\lambda}_c$ is measured from the system. Following this step, an admittance block is employed to generate the Cartesian velocity value that corresponds to force error:

$$\dot{\mathbf{p}}_{AC} = \mathscr{L}^{-1}\left(\mathbf{T}(\frac{s}{sb_j + k_j})\boldsymbol{\lambda}_{error}\right). \tag{4.1}$$

In (4.1), $\mathscr{L}^{-1}(\bullet)$ is the inverse of the Laplace operator, and $\mathbf{T}(\bullet) \in R^{6n_c \times 6n_c}$ is a diagonal matrix containing admittance blocks for each axis. The parameters (b_j, k_j) are the damping and stiffness coefficients concerning the j^{th} axis; $j = x, y, z$ for translational force components, and $j = mx, my, mz$ for roll, pitch, and yaw axis moment components. The yaw axis

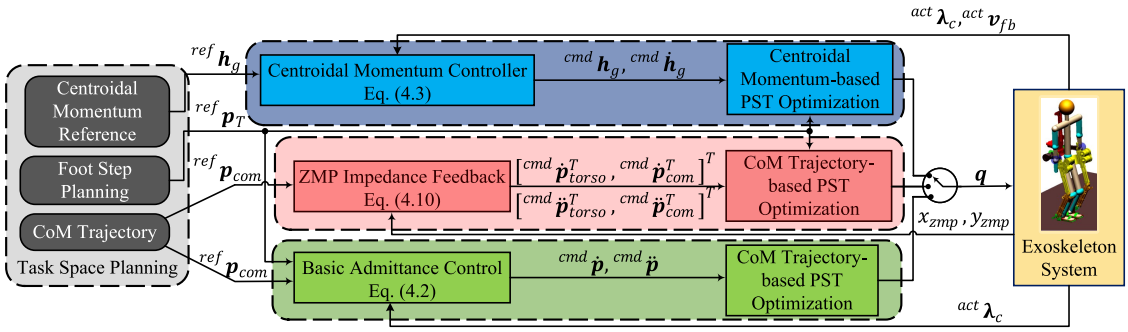

Figure 52: Block diagram of the controller implementation. All the controllers, including the proposed ZIF method, use the proposed PST optimization block to handle underactuation in a dynamically feasible manner. The variable ${}^{ref}\mathbf{p}_T$ denotes the foot trajectory reference.

moment is omitted. The eq. (4.1) should be considered for each contact body. The variable s represents the Laplace operator. The command signal in (3.24) can be configured for this controller as in the following:

$$\begin{aligned} {}^{cmd}\dot{\mathbf{p}} &= {}^{ref}\dot{\mathbf{p}} + \dot{\mathbf{p}}_{AC} \\ {}^{cmd}\ddot{\mathbf{p}} &= {}^{ref}\ddot{\mathbf{p}} + \mathbf{K}_p \boldsymbol{\lambda}_{error} \end{aligned} \tag{4.2}$$

where $({}^{ref}\dot{\mathbf{p}}, {}^{ref}\ddot{\mathbf{p}})$ denote reference velocity and acceleration in Cartesian space. $\mathbf{K}_p$ is a diagonal matrix that stores proportional force feedback gains.

4.2 Centroidal Momentum Controller

To implement centroidal momentum feedback [42], the command signal to the PST optimization block expressed in (3.25) is designed as below:

$$\begin{aligned} {}^{cmd}\mathbf{h}_g &= {}^{ref}\mathbf{h}_g + \mathbf{M}_{p,1}\,{}^{error}\mathbf{h}_g + \mathbf{M}_d\,{}^{error}\dot{\mathbf{h}}_g \\ {}^{cmd}\dot{\mathbf{h}}_g &= {}^{ref}\dot{\mathbf{h}}_g + \mathbf{M}_{p,2}\,{}^{error}\dot{\mathbf{h}}_g \end{aligned} \tag{4.3}$$

where $(\mathbf{M}_{p,1}, \mathbf{M}_{p,2}, \mathbf{M}_d)$ are positive semidefinite diagonal gain matrices. The reference and error values of momentum signals are symbolized with ${}^{ref}\mathbf{h}_g$ and ${}^{error}\mathbf{h}_g$. The centroidal momentum error is obtained by projecting the spatial floating base velocity error onto the CoM

$${}^{error}\mathbf{h}_g = {}^{com}\mathbf{X}_0^* \mathbf{I}_{fb}({}^{ref}\boldsymbol{\nu}_{fb} - {}^{act}\boldsymbol{\nu}_{fb}). \tag{4.4}$$

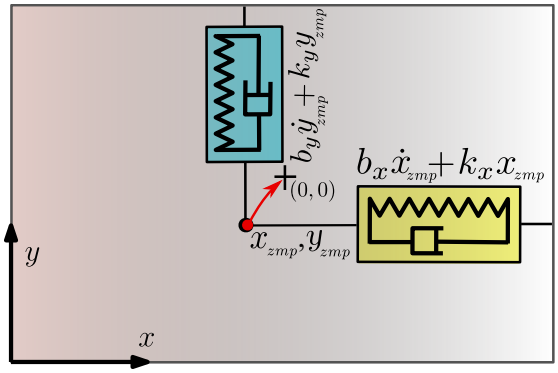

Figure 53: ZMP impedance is described using a 2-DoF virtual spring-damper system.

To avoid numeric differentiation for ${}^{error}\dot{\mathbf{h}}_g$, contact force error is projected onto the CoM, in which ${}^{ref}\boldsymbol{\lambda}_{com}$ is extracted from the desired CoM trajectory.

$$ {}^{error}\dot{\mathbf{h}}_g = {}^{ref}\boldsymbol{\lambda}_{com} - {}^{com}\mathbf{X}_c^{*\,act}\boldsymbol{\lambda}_c. \tag{4.5} $$

4.3 ZMP Impedance Feedback (ZIF) Controller

The ZMP feedback is provided utilizing a virtual spring damper system, i.e., ZMP impedance; see Fig. 53. In classical ZMP feedback methods, the controller is based on a simple pendulum model, and it usually enforces the actual ZMP to follow the reference ZMP that is arbitrarily defined within the support polygon [68]. In the proposed method, two spring-damper couples are defined between the support polygon boundaries and the measured ZMP, whose natural length resides within the polygon. In doing so, the measured ZMP is always pushed back into the support polygon by the force λ_{sx} which is defined as below:

$$ \lambda_{sx} = b_x\dot{x}_{zmp} + k_x x_{zmp} \tag{4.6} $$

where (k_x, b_x) are spring-damper coefficients; see Fig. 53. Using (4.6) and a scaling factor of K_x, the force λ_{sx} can be approximately mapped to multibody x-axis ZMP which is the ratio between $m_{c,y}$ and $\lambda_{c,z}$:

$$ \frac{m_{c,y}}{\lambda_{c,z}} = \frac{(b_x\dot{x}_{zmp} + k_x x_{zmp})}{K_x} \tag{4.7} $$

Recall that the contact moments and forces $(m_{c,y}, \lambda_{c,z})$ can be yielded via the floating dynamics eqs. (3.14)-(3.15). With this in mind, (4.7) can be processed to compute the base acceleration that corresponds to the x-axis ZMP correction action ($\ddot{\mathbf{p}}_{zif,x}$):

$$\ddot{\mathbf{p}}_{zif,x} = -{}^{com}\mathbf{X}_0(\boldsymbol{\xi}\mathbf{I}_{fb})^{\dagger}\boldsymbol{\xi}(\overline{\mathbf{F}}\ddot{\mathbf{q}}_r + \mathbf{C}_{\mathbf{fb}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}_{\mathbf{fb}}(\mathbf{q})) \tag{4.8}$$

$$\boldsymbol{\iota} = (\mathbf{s}_{my} + \frac{(b_x\dot{x}_{zmp} + k_x x_{zmp})}{K_x}\mathbf{s}_z){}^{c}\mathbf{X}_0^{*}. \tag{4.9}$$

where, $\ddot{\mathbf{q}}_r$ is the actuated joint acceleration vector, and the term $\boldsymbol{\iota}$ is used for simplification. A similar approach is carried out for the y-axis ZMP correction action ($\ddot{\mathbf{p}}_{zif,y}$). Using (4.8) and (4.9), the command signals are calculated as follows:

$$\begin{aligned}
[{}^{cmd}\ddot{\mathbf{p}}_{torso}^{\intercal}\ {}^{cmd}\ddot{\mathbf{p}}_{com}^{\intercal}]^{\intercal} &= [\mathbf{0}_{1\times 3}\ {}^{ref}\ddot{\mathbf{p}}_{com}^{\intercal}]^{\intercal} + \ddot{\mathbf{p}}_{zif,x}^{\intercal}\mathbf{K}_{\mathbf{1,x}} + \ddot{\mathbf{p}}_{zif,y}^{\intercal}\mathbf{K}_{\mathbf{1,y}} \\
[{}^{cmd}\dot{\mathbf{p}}_{torso}^{\intercal}\ {}^{cmd}\dot{\mathbf{p}}_{com}^{\intercal}]^{\intercal} &= [\mathbf{0}_{1\times 3}\ {}^{ref}\dot{\mathbf{p}}_{com}^{\intercal}]^{\intercal} + \dot{\mathbf{p}}_{zif,x}^{\intercal}\mathbf{K}_{\mathbf{2,x}} + \dot{\mathbf{p}}_{zif,y}^{\intercal}\mathbf{K}_{\mathbf{2,y}}
\end{aligned} \tag{4.10}$$

In (4.10), $(\mathbf{K}_{\mathbf{1,x}}, \mathbf{K}_{\mathbf{1,y}}, \mathbf{K}_{\mathbf{2,x}}, \mathbf{K}_{\mathbf{2,y}})$ are positive definite diagonal gain matrices. To obtain $(\dot{\mathbf{p}}_{zif,x}, \dot{\mathbf{p}}_{zif,y})$, $(\ddot{\mathbf{p}}_{zif,x}, \ddot{\mathbf{p}}_{zif,y})$ are integrated numerically. By designing the command signal via (4.8)-(4.10), the ZMP correction signal could be mapped to the CoM position and torso orientation by employing the multibody dynamics of the system.

4.4 Locomotion Controllers Simulation

To check the feasibility of the proposed locomotion controllers framework, a series of dynamic walking simulations were conducted using the 3D dynamic simulator MSC ADAM-S. The 40 kg robot model was coupled with 12 distinct anthropomorphic user models ($66 \sim 102$ kg and $170 \sim 190$ cm); see section 2.1.1.

To present a comparative view and examine individual contributions of the identified controllers, three controllers were employed: i) Basic Admittance Control (BAC) that embodies contact force feedback [46], ii) Centroidal Momentum Control (CMC) [42], iii) the proposed ZMP Impedance Feedback (ZIF) control. The controller gains were heuristically tuned in accordance with the non-excitation of unmodeled dynamics, and they are tabulated in Table 4 for CMC, ZIF, and BAC, respectively.

Furthermore, two different scenarios were simulated to investigate the robustness characteristics of the controllers: 1) Dynamic walking (SC#1). 2) Perturbed dynamic walking (SC#2). In both simulation scenarios, the walking parameters were kept the same; the single and double support periods were 0.4 and 0.2 s, respectively, and the maximum foot clearance was 4 cm. The forward velocities were 0.3 m/s for SC#1 and 0.2 m/s for SC#2, and thus, the stride lengths were adjusted accordingly: 38.5 cm and 25.7 cm, respectively. The controllers' performances were evaluated based on ZMP variation in x and y directions (x_{zmp}, y_{zmp}) and pitch axis torso angular velocity ω_y. Each controller was simulated for both scenarios and repeated 12 times for each anthropomorphic subject.

Table 4: Gain values for each locomotion controller

	Parameter	Value
CMC	$\mathbf{M}_{p,1}$	$diag(0,0,0,5.1,7.2,1)10^{-2}$
	$\mathbf{M}_{p,2}$	$diag(8,6,0,30,6,10)10^{-3}$
	$\mathbf{M}_d$	$diag(0,0,0,0.17,2,1)$
ZIF	$[K_x, k_x, K_y, k_y]$	$(1,8,7,4)N/m$
	$[b_x, b_y]$	$(0.1,0.02)N.s/m$
	$[\mathbf{K_{1,x}}, \mathbf{K_{1,y}}]$	$(0.5\mathbf{1}_6,\ 0.22\mathbf{1}_6)$
	$[\mathbf{K_{2,x}}, \mathbf{K_{2,y}}]$	$(0.45\mathbf{1}_6,\ 0.2\mathbf{1}_6)$
BAC	$[k_x, k_y, k_z, k_{mx}, k_{my}]$	$(3,50,70,1500,22000)N/m$
	$[b_x, b_y, b_z, b_{mx}, b_{my}]$	$(2,70,180,2000,26000)N.s/m$
	$\mathbf{K}_p$	$diag(17,10,7,5,0.25)10^{-5}$

Table 5: Statistical significance among the controller performances.

Scenario	Controllers		$\Delta\mu = \overline{C}_I - \overline{C}_{II}$	σ^2_Δ	*pValue*	*tValue* $t(11)$	95% CI	
	C_I	C_{II}					Lower bound	Upper bound
SC#1	BAC	CMC	-0.278	15e-3	$p < 0.001$	30.083	-0.304	-0.252
	BAC	ZIF	0.018	5e-4	$p = 0.006$	3.438	0.003	0.033
	CMC	ZIF	0.297	17e-3	$p < 0.001$	30.621	0.269	0.324
SC#2	BAC	CMC	-0.202	21e-3	$p < 0.001$	18.601	-0.232	-0.171
	BAC	ZIF	0.021	2e-4	$p < 0.001$	6.897	0.012	0.029
	CMC	ZIF	0.222	18e-3	$p < 0.001$	22.381	0.194	0.20

4.4.1 Scenario 1 (SC#1): Dynamic Walking

The results concerning this scenario are displayed in Fig. 54. Pink, green, and blue regions indicate the corresponding variations for ZIF, BAC, and CMC, respectively. Processing the data collected from 12 anthropomorphic subjects, mean±SD regions concerning $x_{zmp}, y_{zmp}, \omega_y$ variations were displayed. Mean variations were denoted via solid lines.
As a result, all controllers provided a dynamically balanced locomotion behavior. Therefore, it is observed that they are feasible options when coupled with the proposed PST optimization block for the exoskeleton-aided walking support task. Among these controllers, the ZMP variations were comparatively limited in the case of the ZIF. In particular, peak-to-peak mean ZMP variations were observed to be farther from the support polygon boundaries generally. When it comes to the suppression of torso angle, the CMC was outperformed by the other controllers. This may imply that the CMC is more sensitive to parameter uncertainties.

4.4.2 Scenario 2 (SC#2): Perturbed Dynamic Walking

The results concerning this scenario are displayed in Fig 55. In this scenario, an external impulsive force of 200 N was applied at the hip of the system, first along the x direction when $t = 10.6\ s$, then along the y direction when $t = 12.4\ s$. These periods were indicated via gray shaded regions in the figure. Similarly, pink, green, and blue regions indicate the corresponding variations for ZIF, BAC, and CMC, respectively. After processing the data collected from 12 anthropomorphic subjects, mean±SD regions concerning $x_{zmp}, y_{zmp}, \omega_y$ variations were displayed. Mean variations were denoted via solid lines.

It is observed that all 3 controllers enabled the human-robot coupled system to exhibit dynamically balanced walking behavior, despite the applied perturbations. The measured ZMP variations were always within the support polygon. Differing from the results of SC#1, the standard deviation regions are comparatively enlarged for BAC and CMC controllers, particularly after the perturbations were applied. Qualitatively speaking, the

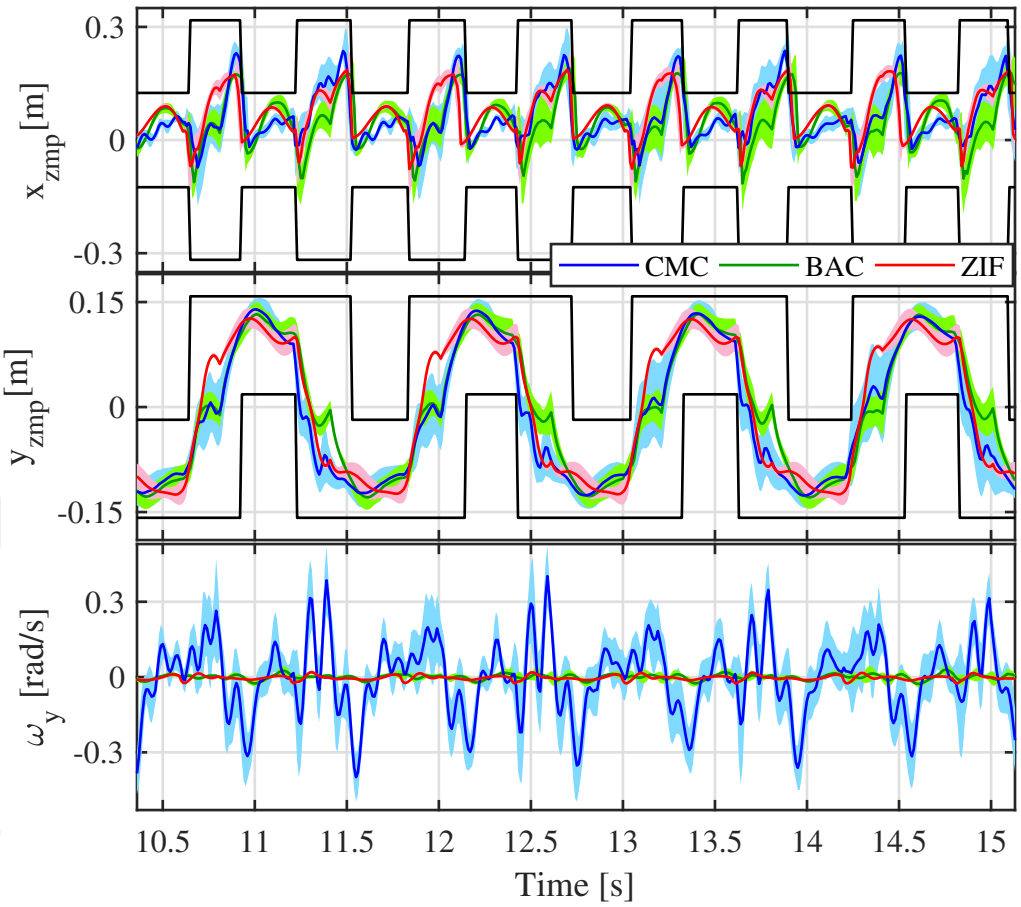

Figure 54: x-axis and y-axis ZMP variations concerning the local frame, and pitch axis torso angular velocity for the case of disturbance-free walking, scenario 1.

proposed controller (ZIF) outperformed the other controllers in terms of ZMP variations. Moreover, pitch axis torso angular velocity was not contained when CMC is used, suggesting a similar result obtained in the case of SC#1.

4.4.3 Statistical Analysis

To determine the most favorable controller option, the results presented in the previous subsections were statistically analyzed. To define metrics, the peak-to-peak values of the x-axis ZMP, y-axis ZMP, and pitch axis torso angular velocity for each step were measured. Taking the average over these peak values makes a representative metric for the analysis. The peak value of ZMP in each step is quite definitive regarding the balance condition, and it may vary in each step. Therefore, an average of such peak values gives a clearer

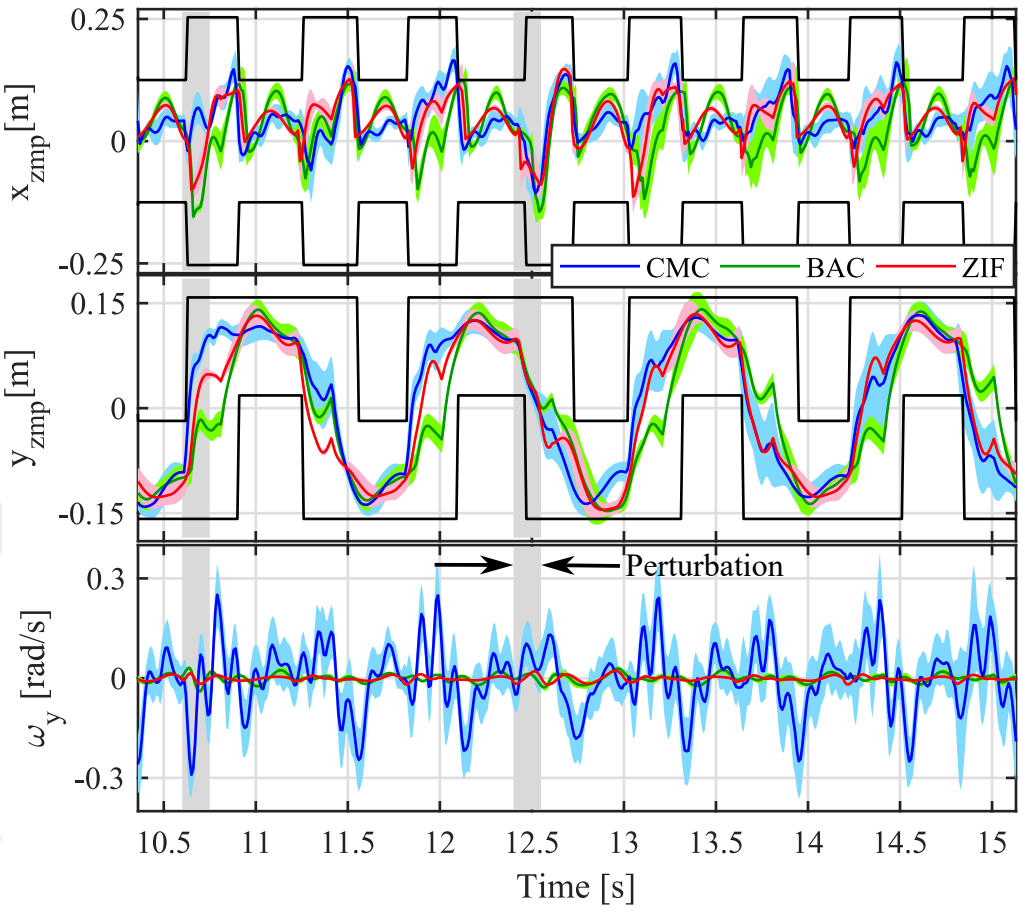

Figure 55: x-axis and y-axis ZMP variations concerning the local frame, and pitch axis torso angular velocity for the case of perturbed walking, scenario 2. The gray regions represent the perturbation instants.

view of the performance assessment. The aforementioned metrics are denoted as x_{zmp}^{pp}, y_{zmp}^{pp}, and ω_y^{pp} concerning x-axis ZMP, y-axis ZMP, and pitch axis torso angular velocity, respectively.

An analysis of variance was performed using Repeated Measures ANOVA to test the significant difference between controllers concerning both walking scenarios. The MATLAB function *ranova(•)* was used, at the significance level of 0.05. There was a significant difference between controllers in SC#1, $F(2,22) = 798.1819, p < 0.001$, and SC#2, $F(2,22) = 401.8652, p < 0.001$.

Afterward, a post-hoc test was conducted using Bonferroni with an adjusted significance level of 0.0167 per test (0.05/3). MATLAB function *multcompare(•)* was used and

the results were displayed in Table 5. The analysis revealed a significant reduction in average peak-to-peak values when ZIF $(M = 0.2113, SD = 0.0152)$ was used, compared to BAC $(M = 0.2295, SD = 0.0306)$, and CMC $(M = 0.5078, SD = 0.0584)$ for the case of SC#1. When it comes to SC#2, the peak-to-peak averages are significantly less when ZIF $(M = 0.1855, SD = 0.0179)$ was used, compared to BAC $(M = 0.2062, SD = 0.0159)$, and CMC $(M = 0.4078, SD = 0.0635)$. The *tValue* and *pValue* parameters for each pairwise comparison are reported in Table 5, columns 6 and 7.

Columns 2 and 3 indicate the first and second controllers being compared, which are represented by C_I and C_{II} respectively. The fourth and fifth columns show the mean difference, $\Delta\mu$, and mean squared error, σ_Δ^2, of the two controllers in comparison. The last two columns display the 95% confidence interval confidence interval (CI) of the two controllers' mean difference.

The results indicate that the ZIF controller is statistically more favorable when compared to CMC and BAC, concerning both scenarios. Furthermore, Fig. 56 illustrates the variation between the three controllers based on the averaged values of the three metrics, namely, $x_{zmp}^{pp}, y_{zmp}^{pp}, \omega_y^{pp}$ both for SC#1 and SC#2. Similarly, the controller ZIF provided less variation when compared to CMC and BAC, indicating its robustness against parameter uncertainties and perturbations.

4.5 Modified ZMP Impedance Feedback

Although the proposed ZIF controller showed a robust performance in the conducted simulation experiments, the presence of differential term $\dot{x}_{zmp}$ in (4.9) cripples its practical implementation. To improve its applicability, a Modified ZMP impedance feedback (MZIF) controller is introduced.

The reaction force due to the spring damper is as follows:

$$\delta\boldsymbol{\lambda}_{zif} = \mathbf{K}_{zif}\bar{\mathbf{x}}_{zmp} + \mathbf{B}_{zif}\dot{\bar{\mathbf{x}}}_{zmp}. \tag{4.11}$$

The symbols $\delta\boldsymbol{\lambda}_{zif} \in \mathbb{R}^2$, and $\bar{\mathbf{x}}_{zmp} \in \mathbb{R}^2$ represent the reaction forces and measured ZMP

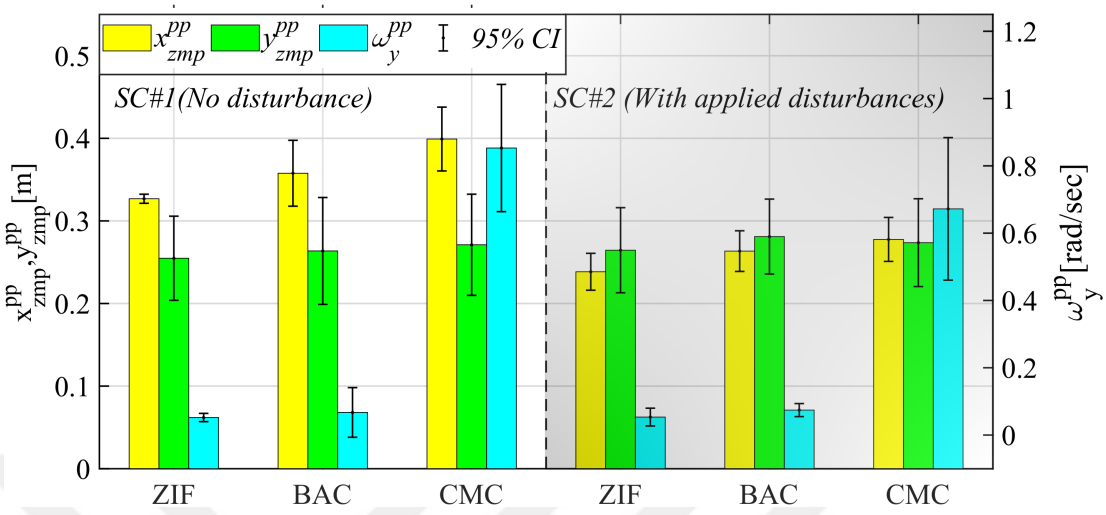

Figure 56: Average values of x^{pp}_{zmp}, y^{pp}_{zmp}, and ω^{pp}_{y} for 12 anthropomorphic subjects.

variations in x and y directions, respectively. The matrices $\mathbf{K}_{zif} \in \mathbb{R}^{2\times2}$, and $\mathbf{B}_{zif} \in \mathbb{R}^{2\times2}$ are two diagonal stiffness and damping matrices, respectively.

It is known that the ZMP represents the ratio between the horizontal contact moments and the vertical force. Hence, the reaction force can be approximately regarded as follows:

$$\delta\boldsymbol{\lambda}_{zif} \propto \frac{\delta\boldsymbol{m}_{x,y}}{m_r g}, \tag{4.12}$$

where, $\delta\boldsymbol{m}_{x,y} \in \mathbb{R}^2$ is a vector representing the contact moments in x, and y directions.The symbols m_r, and g denote the robot aggregated mass and the gravitational acceleration, respectively. Using (4.11), equation (4.12) can be solved in terms of contact horizontal moments as follows:

$$\mathbf{B}^{-1}_{zif} e^{-\mathbf{L}_{zif}t} \int_0^t e^{\mathbf{L}_{zif}t}\,\delta\boldsymbol{m}_{x,y}dt = \overline{\mathbf{x}}_{zmp}, \tag{4.13}$$

where, $\mathbf{L}_{zif} = \mathbf{B}^{-1}_{zif}\mathbf{K}_{zif}$. Using integration by parts, the following equation can be obtained:

$$\mathbf{B}^{-1}_{zif}\delta\mathbf{h}_{x,y} - \mathbf{B}^{-1}_{zif}\mathbf{L}_{zif} e^{-\mathbf{L}_{zif}t} \int_0^t e^{\mathbf{L}_{zif}t}\,\delta\mathbf{h}_{x,y}dt = \overline{\mathbf{x}}_{zmp}, \tag{4.14}$$

where, $\delta\mathbf{h}_{x,y} \in \mathbb{R}^2$ is the contact momentum vector whose components are corresponding to $\delta\boldsymbol{m}_{x,y}$. Through the discretization of (4.14), and applying Heun's method for numerical integration [69], the solution of (4.14) can be yielded as follows:

$$\delta\mathbf{h}^k_{x,y} = (\mathbf{1}_2 - 0.5\mathbf{L}_{zif}t_s)^{-1}(\mathbf{L}_{zif}\mathbf{I}^{k-1}_{x,y} + 0.5\mathbf{L}_{zif}\delta\mathbf{h}^{k-1}_{x,y}t_s + \mathbf{B}_{zif}\overline{\mathbf{x}}_{zmp}), \tag{4.15}$$

where, $\mathbf{I}_{x,y}^{k}$ is an integral term that can be approximated as:

$$\mathbf{I}_{x,y}^{k} \approx \mathbf{I}_{x,y}^{k-1} + t_s \frac{\delta \mathbf{h}_{x,y}^{k} + \delta \mathbf{h}_{x,y}^{k-1}}{2}, \tag{4.16}$$

and t_s is the sampling time. It is worth mentioning that the value of $\mathbf{I}_{x,y}^{0}$ is assigned to zero at the beginning of each step, to eliminate the residual effect induced by the previous steps. To map the obtained momentum in (4.15) to the joint space, the contact momentum is firstly projected onto centroidal dynamics, then CMM is used to map it to the joint velocity variables such that:

$$\delta \dot{\mathbf{q}}_{zif} = \mathbf{A}_{g}^{\dagger 0} \mathbf{X}_{c}^{*} \delta \mathbf{h}_{x,y}^{k}. \tag{4.17}$$

Consequently, ZIF control action $\delta \dot{\mathbf{q}}_{zif}$ is added to the referential trajectories $\dot{\mathbf{q}}_{ref}$ giving the command signal

$$\dot{\mathbf{q}}_{cmd} = \delta \dot{\mathbf{q}}_{zif} + \dot{\mathbf{q}}_{ref}. \tag{4.18}$$

CHAPTER V

INERTIAL PARAMETER IDENTIFICATION AND ADAPTIVE CONTROL

5.1 Adaptive Locomotion Controller

The aforementioned locomotion controllers directly depend on the dynamic model. This is obvious from equation (4.8) for ZIF controller, and (4.17) for its modified version. Consequently, the variation of the inertial parameters may adversely affect the controllers' performance. Hence, an estimated dynamic model based on estimated inertial parameters could have a major improvement on the locomotion controller's performance. For example, the ZIF control action given by (4.8) can be rewritten in terms of the estimated dynamic model parameters as follows:

$$^{ad}\ddot{\mathbf{p}}_{zif,x} = -^{com}\mathbf{X}_0(\boldsymbol{\xi}\hat{\mathbf{I}}_{fb})^{\dagger}\boldsymbol{\xi}(\hat{\bar{\mathbf{F}}}\ddot{\mathbf{q}}_r + \hat{\mathbf{C}}_{\mathbf{fb}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \hat{\mathbf{G}}_{\mathbf{fb}}(\mathbf{q})), \tag{5.1}$$

where $^{ad}\ddot{\mathbf{p}}_{zif,x}$ is adaptive ZIF control action. The accent $\wedge$ refers to the estimated variables. A direct connection between the estimated variables and the estimated dynamic model parameters is figured as follows:

$$\hat{\mathbf{H}}(\mathbf{q}) = \begin{bmatrix} \hat{\mathbf{I}}_{fb} & \hat{\bar{\mathbf{F}}} \\ \hat{\bar{\mathbf{F}}}^{\mathsf{T}} & \hat{\mathbf{H}}_r \end{bmatrix}, \ \hat{\mathbf{C}}_{\mathbf{fb}}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{S}_{\mathbf{fb}}\hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}}) \ \textit{and} \ \hat{\mathbf{G}}_{\mathbf{fb}}(\mathbf{q}) = \mathbf{S}_{\mathbf{fb}}\hat{\mathbf{G}}(\mathbf{q}), \tag{5.2}$$

where $\hat{\mathbf{H}}(\mathbf{q})$ is the estimated inertia matrix, $\hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})$ is the estimated Coriolis and centrifugal matrix, and $\hat{\mathbf{G}}(\mathbf{q})$ is the estimated gravitational vector.

In the case of modified ZIF, its control action exploits the inertial parameters estimation procedure using equation (4.17) as follows:

$$^{ad}\delta\dot{\mathbf{q}}_{zif} = \hat{\mathbf{A}}_g^{\dagger 0}\mathbf{X}_c^*\delta\mathbf{h}_{x,y}^k, \tag{5.3}$$

where ${}^{ad}\delta\dot{\mathbf{q}}_{zif}$ is the adaptive modified ZIF control action and $\hat{\mathbf{A}}_g$ is the estimated CMM. The estimated $\hat{\mathbf{A}}_g^k$ can be directly obtained from the estimated inertia matrix as follows:

$$\hat{\mathbf{A}}_{\mathbf{g}} = {}^{com}\mathbf{X}_0^* \mathbf{S}_{\mathbf{fb}} \hat{\mathbf{H}}(\mathbf{q}). \tag{5.4}$$

To this end, an online real-time applicable inertial parameters identification algorithm could be employed to estimate inertia matrix $\hat{\mathbf{H}}(\mathbf{q})$, Corilos centrifugal matrix $\hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})$, and gravitational matrix $\hat{\mathbf{G}}(\mathbf{q})$. Direct relations between the inertial parameters and the dynamic model matrices are given as follows [70]:

$$\hat{\mathbf{H}}(\mathbf{q}) = \overline{\mathbf{J}}^{\mathsf{T}} \hat{\overline{\overline{\mathbf{I}}}} \overline{\mathbf{J}}, \tag{5.5}$$

$$\hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}}) = \overline{\mathbf{J}}^{\mathsf{T}} \hat{\mathbf{B}} \overline{\mathbf{J}} + \overline{\mathbf{J}}^{\mathsf{T}} \hat{\overline{\overline{\mathbf{I}}}} \dot{\overline{\mathbf{J}}}, \tag{5.6}$$

$$\hat{\mathbf{G}}(\mathbf{q}) = -\overline{\mathbf{J}}^{\mathsf{T}} \hat{\overline{\overline{\mathbf{I}}}} \overline{\mathbf{X}} \mathbf{a}_g^0. \tag{5.7}$$

The matrix $\overline{\mathbf{J}} = [\mathbf{J}_1^{\mathsf{T}}\ \mathbf{J}_2^{\mathsf{T}}\ ...\ \mathbf{J}_{n_b}^{\mathsf{T}}]^{\mathsf{T}}$ is a stack of Jacobian matrices that are described for each link fixed frame, and $\hat{\overline{\overline{\mathbf{I}}}} = diag([\hat{\mathbf{I}}_{s,1}\ \hat{\mathbf{I}}_{s,2}\ ...\ \hat{\mathbf{I}}_{s,n_b}])$ is a block diagonal matrix as the matrices $\hat{\mathbf{I}}_{s,i} \in \mathbb{R}^{6\times 6}$ represent the spatial inertia matrix of the estimated inertial parameters for each link i which are projected onto the axis of rotation [1]. The spatial inertia matrix is described as:

$$\hat{\mathbf{I}}_{s,i} = \begin{bmatrix} \hat{\mathbf{I}}_i & \overline{S}(\hat{\mathbf{h}}_i) \\ \overline{S}(\hat{\mathbf{h}}_i)^{\mathsf{T}} & \hat{m}_i \mathbf{1}_3 \end{bmatrix}, \tag{5.8}$$

where, $\hat{\mathbf{I}}_i \in \mathbb{R}^{3\times 3}$ is the estimated link i inertia tensor, $\hat{\mathbf{h}}_i$ is the estimated link i first order moment of inertia, and $\hat{m}_i$ is the estimated link i mass.The function $\overline{S}(\bullet)$ is a skew symmetric matrix. The vector $\mathbf{a}_g^0$ is a spatial vector representing the gravitational acceleration as projected onto the floating base. The matrix $\hat{\mathbf{B}} = diag([\hat{\mathbf{O}}_1\ \hat{\mathbf{O}}_2\ ...\ \hat{\mathbf{O}}_{n_b}])$ is a block diagonal matrix, where

$$\hat{\mathbf{O}}_i = \boldsymbol{\nu}_i \times^* \hat{\mathbf{I}}_{s,i} + \hat{\mathbf{I}}_{s,i} \boldsymbol{\nu}_i \times, \tag{5.9}$$

$$\boldsymbol{\nu}_i \times = \begin{bmatrix} \overline{S}(\boldsymbol{\omega}_i) & \mathbf{0} \\ \overline{S}(\mathbf{v}_i) & \overline{S}(\boldsymbol{\omega}_i) \end{bmatrix}, \tag{5.10}$$

$$\boldsymbol{\nu}_i \times^* = -\boldsymbol{\nu}_i \times, \tag{5.11}$$

$\boldsymbol{\omega}_i$, and $\mathbf{v}_i$ are the angular and linear velocities for each link fixed frame concerning the global frame respectively. Matrix $\overline{\mathbf{X}} = [{}^1\mathbf{X}_0^\intercal \ {}^2\mathbf{X}_0^\intercal \ ... \ {}^{n_b}\mathbf{X}_0^\intercal]^\intercal$ is a stack of the spatial transformation matrix for each link. Spatial transformation is developed as:

$$ {}^B\mathbf{X}_A = \begin{bmatrix} {}^B\mathbf{E}_A & \mathbf{0} \\ -\overline{S}({}^B\mathbf{r}_A){}^B\mathbf{E}_A & {}^B\mathbf{E}_A \end{bmatrix}, \tag{5.12} $$

where, ${}^B\mathbf{E}_A \in \mathbb{SO}^{3\times3}$ is the rotation matrix of frame A relative to frame B, and vector ${}^B\mathbf{r}_A \in \mathbb{R}^3$ is the displacement vector of frame A relative to frame B. By developing an algorithm to estimate the inertial parameters $\hat{\overline{\mathbf{I}}}$ the dynamic model parameters (5.5-5.7) can be estimated.

5.2 Inertial Parameter Identification

Modern Inertial Parameter Identification (IPID) approaches stem from a seminal work in [71] that recast the manipulator dynamic model to linearly relate the measured joint forces to the inertial parameters via acceleration-dependent coefficients. Afterward, the inertial parameters, such as mass, the center of mass, and inertia tensors, are fitted to the observed data via a least square approach. The majority of the classical approaches solely focused on fixed base robots [72–75]. On the contrary, the floating base structure of legged systems complicates the matter due to the presence of unactuated virtual joints, force sensing requirements at all joints and contact points, and contact phase variations. These lead to variations regarding the identifiable parameters. Ayusawa et al. showed that full-body inertial parameters can be estimated only from contact forces and the free-flying motion [76]. Although this method allows the exclusion of joint friction and actuator dynamics from the identification process, issues associated with force sensors would cripple its application.Generally speaking, two main challenging subjects would cripple the identification procedure.

The first subject is the limitation of the excitable parameters, due to robot joint movement constraints [77] and the identification trajectories [78, 79]. A set of classical approaches were used to determine the base parameters, (the minimum set of inertial parameters that can be used to determine the dynamic model), either symbolically [80–83] or numerically using SVD or QR decomposition [71, 84].

The second subject is the complete physical consistency condition noticed by Traversaro et al. [53]. The condition stated that the inertia tensors are not only positive definite but also their eigenvalues verify the so-called triangle constraint. In [85], Wensing et al. treated the triangle constraints in the form of Linear Matrix Inequalities (LMI) using a 4×4 pseudo inertia matrix. A Semidefinite Programming (SDP) framework can be applied to impose the pseudo inertia matrix positivity on the identification algorithm. The current problem in the literature is that the majority of proposed approaches are implemented offline, rendering their use in real-time is infeasible. Researchers used to solve the problem in an online form, but they did not consider the parameters identifiability, or triangle constraints [86, 87].

To tackle this problem, the inertial parameter identification is converted to a recursive procedure with real-time applicability. The proposed identification algorithm in this work represents a complementary step toward an adaptive controller for the exoskeleton. The algorithm uses three main folds to develop a feasible identification approach: 1) A numerical algorithm is adopted to develop a Recursive Identifiability Projection Matrix (RIPM), that is used to select the excitable parameters in the current sample and project them onto the space of the identifiable parameters developed from the previously sampled data. 2) To guarantee the complete physical consistency of the inertial parameters, an SDP optimization manifold is developed based on a customized Interior Point Method (IPM) [54, 88, 89] that can perform recursively. 3) To mitigate the effect of noise, and contact transitions, and to reduce the measurement requirement, the concept of filtered regressor is exploited. Using this concept enables data filtration without losing any of the identifiable parameters information [55].

5.3 Preliminaries

5.3.1 Rigid-Body Dynamics

A rigid body system can be represented as

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \bar{\mathbf{h}}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{f}, \tag{5.13}$$

where, $\mathbf{H}(\mathbf{q}) \in \mathbb{R}^{n_{dof} \times n_{dof}}$ is the inertia matrix, n_{dof} is the number of degrees of freedom, $\bar{\mathbf{h}}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n_{dof}}$ is a vector represents Coriolis and centrifugal, and gravitational terms. A generalized force vector $\mathbf{f} \in \mathbb{R}^{n_{dof}}$ is defined as

$$\mathbf{f} = \mathbf{S}\boldsymbol{\tau} + \mathbf{J_c}(\mathbf{q})^\intercal \boldsymbol{\lambda}_\mathbf{c}, \tag{5.14}$$

representing the resultant of the joint torques $\boldsymbol{\tau} \in \mathbb{R}^{n_r}$ and the external contact wrenches vector $\boldsymbol{\lambda}_c \in \mathbb{R}^{6n_c}$. The two matrices $\mathbf{S} \in \mathbb{R}^{n_{dof} \times n_r}$, and $\mathbf{J_c}(\mathbf{q}) \in \mathbb{R}^{6n_c \times n_{dof}}$ represents the actuated joints selection matrix and contact forces Jacobian matrix, respectively.

The dynamic model (5.13) can be represented linearly concerning the inertial parameter vector $\boldsymbol{\pi} \in \mathbb{R}^{10n_b}$ [71]. For a set of samples equation (5.13) can be represented as:

$$\mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\boldsymbol{\pi} = \mathbf{f}_s, \tag{5.15}$$

where, n_b is the number of rigid bodies, and $\mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) \in \mathbb{R}^{n_s n_{dof} \times 10n_b}$ is the regressor matrix and n_s represents the number of acquired samples. The vector $\boldsymbol{\pi} = [\boldsymbol{\pi}_1^\intercal, ..., \boldsymbol{\pi}_{n_b}^\intercal]^\intercal$, stores the inertia parameters vector of each body, given by $\boldsymbol{\pi}_i \in \mathbb{R}^{10}$. The symbol $\mathbf{f}_s \in \mathbb{R}^{n_s n_{dof}}$ is a vector that stores the generalized force vectors $\mathbf{f}$ over different samples. Each body i is represented by 10 inertial parameters as projected on the axis of rotation given as follows:

$$\pi_i = [I_{i,xx}\ I_{i,yy}\ I_{i,zz}\ I_{i,xy}\ I_{i,xz}\ I_{i,yz}\ h_{i,x}\ h_{i,y}\ h_{i,z}\ m_i]^\intercal, \tag{5.16}$$

where, m_i represents the link i mass, $\boldsymbol{h}_i = [h_{i,x}\ h_{i,y}\ h_{i,z}]^\intercal$ is link i first order mass moment, and $I_{i,xx}, I_{i,yy}, I_{i,zz}, I_{i,xy}, I_{i,xz}, I_{i,yz}$ represents the inertia tensor $\boldsymbol{I}_i$ components:

$$\mathbf{I}_i = \begin{bmatrix} I_{i,xx} & I_{i,xy} & I_{i,xz} \\ I_{i,xy} & I_{i,yy} & I_{i,yz} \\ I_{i,xz} & I_{i,yz} & I_{i,zz} \end{bmatrix} \tag{5.17}$$

The inertial parameters are functions in the density distribution $\rho(\mathbf{r}) : \mathbb{R}^3 \rightarrow \mathbb{R}^{++}$:

$$m_i = \int_{\mathbb{R}^3} \rho_i(\mathbf{r}_i) d\mathbf{r}_i, \tag{5.18}$$

$$\mathbf{h}_i = \int_{\mathbb{R}^3} \mathbf{r}_i \rho_i(\mathbf{r}_i) d\mathbf{r}_i, \tag{5.19}$$

$$\mathbf{I}_i = \int_{\mathbb{R}^3} \overline{S}(\mathbf{r}_i)\overline{S}(\mathbf{r}_i)^\intercal \rho_i(\mathbf{r}_i) d\mathbf{r}_i, \tag{5.20}$$

where, $\mathbf{r}_i \in \mathbb{R}^3$ represents the location of any arbitrary point within link i boundaries with respect to its axis of rotation. To impose physical consistency, the positivity of (5.18) and (5.20) should be respected while solving (5.15).

$$\begin{aligned} \hat{\pi} \quad &= \quad \arg \min_{\forall \boldsymbol{\pi} \in \mathbb{R}^{10 n_b}} ||\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\pi} - \mathbf{f}_s||^2 \\ &s/t \quad m_i > 0, \mathbf{I}_i(\pi) > 0, i \in \{1, 2, ..., n_b\} \end{aligned} \tag{5.21}$$

5.3.2 Complete Physical Consistency

In [53] Traversaro et al. reported that not only does the inertia tensor (5.20) should be positive definite, but also its eigenvalues must obey the triangle constraint. Due to the symmetrical structure of link i inertia tensor around its center of mass (CoM) $\mathbf{I}_i^c$, so it can be diagonalized via orthogonal matrix $\overline{\mathbf{O}}_i$ as follows:

$$\mathbf{I}_i^c = \overline{\mathbf{O}}_i \boldsymbol{\Lambda}_i \overline{\mathbf{O}}_i^\intercal, \tag{5.22}$$

where, $\boldsymbol{\Lambda}_i \in \mathbb{R}^{3\times 3}$ is a diagonal matrix, whose elements represent the eigenvalues of $\mathbf{I}_i^c$. Using (5.20), and (5.22), the three diagonal elements $(\lambda_{i,1}, \lambda_{i,2}, \lambda_{i,3})$ of $\boldsymbol{\Lambda}_i$ constitute the following inequalities:

$$\lambda_{i,1} + \lambda_{i,2} > \lambda_{i,3},\ \lambda_{i,1} + \lambda_{i,3} > \lambda_{i,2},\ \lambda_{i,3} + \lambda_{i,2} > \lambda_{i,1}. \tag{5.23}$$

The inequalities (5.23) represent the triangle constraint. Note that the relation between (5.23) and the inertial parameters (5.16) is nonlinear. To this end, Wensing et al. [85] showed that the triangle constraint (5.23) can be justified by the positive definiteness of the

pseudo inertia matrix $\mathbf{I}_{P,i}$ which can be represented linearly in (5.16). Following this proof, inequalities given in (5.23) are reformulated as following:

$$\frac{\lambda_{i,1}+\lambda_{i,2}+\lambda_{i,3}}{2} > max(\lambda_{i,1},\lambda_{i,2},\lambda_{i,3}), \tag{5.24}$$

As the sum of the eigenvalues equals the matrix trace, (5.24) can be represented as:

$$\frac{tr(\mathbf{I}_i^c)}{2} > \lambda_{max}(\mathbf{I}_i^c), \tag{5.25}$$

where, $tr(\bullet)$ is the trace function, and $\lambda_{max}(\bullet)$ returns the maximum eigenvalue. The inequality (5.25) may lead to a positive definite matrix given by:

$$\frac{tr(\mathbf{I}_i^c)}{2}\mathbf{1}_3 - \mathbf{I}_i^c > \mathbf{0}. \tag{5.26}$$

Using the parallel axis theorem, the representation of inequality (5.26) in terms of $\mathbf{I}_i$ could lead to:

$$\frac{tr(\mathbf{I}_i - m_i\overline{S}(\mathbf{r_{i,c}})\overline{S}(\mathbf{r_{i,c}})^\intercal)}{2}\mathbf{1}_3 - (\mathbf{I}_i - m_i\overline{S}(\mathbf{r_{i,c}})\overline{S}(\mathbf{r_{i,c}})^\intercal) > \mathbf{0}, \tag{5.27}$$

where $\mathbf{r}_{i,c} \in \mathbb{R}^3$ represents the distance between link i center of rotation and its CoM. Since $tr(\overline{S}(\mathbf{r_{i,c}})^\intercal\overline{S}(\mathbf{r_{i,c}})) = 2\mathbf{r_{i,c}}^\intercal\mathbf{r_{i,c}}$ and $\mathbf{r_{i,c}}\mathbf{r_{i,c}}^\intercal = \mathbf{r_{i,c}}^\intercal\mathbf{r_{i,c}}\mathbf{1}_3 - \overline{S}(\mathbf{r_{i,c}})^\intercal\overline{S}(\mathbf{r_{i,c}})$. Using (5.19) and inequality (5.27) can lead to the following inequality:

$$(\frac{tr(\mathbf{I}_i)}{2}\mathbf{1}_3 - \mathbf{I}_i) - \frac{1}{m_i}\mathbf{h}_i\mathbf{h}_i^\intercal > \mathbf{0}. \tag{5.28}$$

Using Schur complement, the following inequality can be concluded:

$$\mathbf{I}_{P,i}(\pi_i) = \begin{bmatrix} (\frac{tr(\mathbf{I}_i)}{2}\mathbf{1}_3 - \mathbf{I}_i) & \mathbf{h}_i \\ \mathbf{h}_i^\intercal & m_i \end{bmatrix} > \mathbf{0}. \tag{5.29}$$

Hence, a linear interpretation of the triangle constraints in terms of inertial parameters given by (5.16) is deduced.

5.3.3 Identifiable Inertial Parameters

It may not be possible to excite all the inertial parameters given in (5.16) due to joint constraints or the form of trajectories [90]. To provide a feasible solution, a set of identifiable parameters should be selected. The inertial parameters can be classified into three sets defined as follows: The first set denotes the base parameters. The base parameters are defined as the minimal inertial parameters needed to specify the dynamic model. The second set is the linearly identifiable inertial parameters. That set stores the linearly coupled parameters to the base parameters. The third set represents non-identifiable parameters. These parameters correspond to the zero columns in the regressor. The first and second sets constitute the set of identifiable parameters.

The identifiable parameters can be numerically determined using QR decomposition [84]. The regressor matrix given in (5.15) can be decomposed as follows:

$$\begin{bmatrix} \mathbf{Y}_b & \mathbf{Y}_l \end{bmatrix} \begin{bmatrix} \boldsymbol{\pi}_b \\ \boldsymbol{\pi}_l \end{bmatrix} = \mathbf{f}_s, \tag{5.30}$$

where, $\mathbf{Y}_b \in \mathbb{R}^{n_s n_{dof} \times n_{bp}}$ and $\mathbf{Y}_l \in \mathbb{R}^{n_s n_{dof} \times (n_i - n_{bp})}$ denote the regressor columns corresponding to the base and linearly dependent inertial parameters, respectively. The symbols n_{bp} and n_i represent the number of bases and identifiable parameters. $\boldsymbol{\pi}_b$ and $\boldsymbol{\pi}_l$ represent the base and linearly dependent inertial parameters. Using QR decomposition of the decomposed regressor matrix in (5.30) could lead to:

$$\begin{bmatrix} \mathbf{Y}_b & \mathbf{Y}_l \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_b & \mathbf{Q}_0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_b & \mathbf{R}_l \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{5.31}$$

where, $\mathbf{Q}_b \in \mathbb{R}^{n_s n_{dof} \times n_{bp}}$ is column orthonormal matrix, $\mathbf{R}_b \in \mathbb{R}^{n_{bp} \times n_{bp}}$ is upper triangle matrix, and $\mathbf{Q}_0 \in \mathbb{R}^{n_s n_{dof} \times (n_s n_{dof} - n_{bp})}$ represents the columns corresponding to zero rows in (5.31). Using (5.31), (5.30) can be represented as:

$$\begin{bmatrix} \mathbf{1}_b & \mathbf{K} \end{bmatrix} \begin{bmatrix} \boldsymbol{\pi}_b \\ \boldsymbol{\pi}_l \end{bmatrix} = \mathbf{R}_b^{-1} \mathbf{Q}_b^{\intercal} \mathbf{f}_s, \tag{5.32}$$

where, the matrix $\mathbf{K} = \mathbf{R}_b^{-1}\mathbf{R}_l$. Consequently, (5.32) can be minimized instead of (5.15). The optimization problem in (5.21), can be modified using (5.32) and (5.29) such that:

$$\begin{aligned} \hat{\overline{\boldsymbol{\pi}}} \quad &= \quad \arg\min_{\forall \overline{\boldsymbol{\pi}} \in \mathbb{R}^{n_i}} ||\overline{\boldsymbol{\Phi}}\overline{\boldsymbol{\pi}} - \bar{\mathbf{f}}||^2 \\ & s/t \quad \mathbf{I}_{P,i}(\overline{\boldsymbol{\pi}}_i) + \mathbf{I}^0_{P,i} > \mathbf{0},\ i \in \{1,2,...,n_b\}, \end{aligned} \tag{5.33}$$

where, $\overline{\boldsymbol{\Phi}}$ and $\bar{\mathbf{f}}$ are defined respectively, using (5.32) such that: $\overline{\boldsymbol{\Phi}} = \begin{bmatrix} \mathbf{1}_b & \mathbf{K} \end{bmatrix}$, and $\bar{\mathbf{f}} = \mathbf{R}_b^{-1}\mathbf{Q}_b^{\intercal}\mathbf{f}_s$. The identifiable parameters set in (5.33) is given by $\overline{\boldsymbol{\pi}} = [\boldsymbol{\pi}_b^{\intercal}\ \boldsymbol{\pi}_l^{\intercal}]^{\intercal}$. Matrix $\mathbf{I}^0_{P,i} \in \mathbb{R}^{4\times 4}$ is the pseudo inertia matrix (5.29) as a function in the initial guess of the non-identifiable parameters.

5.4 Filtered Regressor

The presence of joint accelerations $\ddot{\mathbf{q}}$ in (5.15) requires numerical differentiation of $\dot{\mathbf{q}}$, which is severely affected by noise. This can be avoided by employing a low pass filter in (5.13) [55]. Thus equation (5.13) can be expressed as:

$$\frac{\gamma}{s+\gamma}(\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{Y}_{\mathbf{h}}(\mathbf{q},\dot{\mathbf{q}})\pi) = \mathbf{f}_f, \tag{5.34}$$

where, $\mathbf{Y}_{\mathbf{h}}(\mathbf{q},\dot{\mathbf{q}})$ represents a linear mapping between, $\overline{\mathbf{h}}(\mathbf{q},\dot{\mathbf{q}})$ and the inertial parameters. The filtered force vector is $\mathbf{f}_f = \frac{\gamma}{s+\gamma}\mathbf{f}_s$. The symbol γ denotes the cutoff frequency. Considering the impulse response of the first order filter $w(t) = \gamma e^{-\gamma t}$, the first term in the left-hand side of the equation (5.34) computed through convolving

$$\frac{\gamma}{s+\gamma}(\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}) = \int_0^t w(t-\tau)\mathbf{H}(\mathbf{q}(\tau))\ddot{\mathbf{q}}(\tau)\mathbf{d}\tau. \tag{5.35}$$

Assuming zero initial velocities, integrating (5.35) by parts is given as:

$$\frac{\gamma}{s+\gamma}(\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}) = \gamma\mathbf{H}(\mathbf{q}(t))\dot{\mathbf{q}}(t)) - \frac{\gamma}{s+\gamma}(\dot{\mathbf{H}}(\mathbf{q})\dot{\mathbf{q}} + \gamma(\mathbf{H}(\mathbf{q})\dot{\mathbf{q}}). \tag{5.36}$$

In sense of Newton-Euler, the inertia matrix can be computed as [70]:

$$\mathbf{H}(\mathbf{q}) = \overline{\mathbf{J}}(\mathbf{q})^{\intercal}\overline{\mathbf{I}}\overline{\mathbf{J}}(\mathbf{q}), \tag{5.37}$$

where the matrix $\bar{\mathbf{I}} \in \mathbb{R}^{6n_b \times 6n_b}$ is a block diagonal matrix, given in terms of rigid bodies spatial inertia, such that: $\bar{\mathbf{I}} = diag[\mathbf{I}_{s,1}, \mathbf{I}_{s,2}, \ \ldots \ , \mathbf{I}_{s,nb}]$. The spatial inertia $\mathbf{I}_{s,i}$ for each rigid body element can be represented linearly in terms of the inertial parameters as follows:

$$\mathbf{I}_{s,i} = \begin{bmatrix} \mathbf{I}_i & \overline{S}(\mathbf{h}_i) \\ \overline{S}(\mathbf{h}_i)^\intercal & m\mathbf{1}_3 \end{bmatrix}. \tag{5.38}$$

Hence, the time derivative of (5.37) can be computed in terms of Jacobian matrices derivatives $\dot{\bar{\mathbf{J}}}(\mathbf{q})$, such that:

$$\dot{\mathbf{H}}(\mathbf{q}) = 2\bar{\mathbf{J}}(\mathbf{q})^\intercal \bar{\mathbf{I}} \dot{\bar{\mathbf{J}}}(\mathbf{q}), \tag{5.39}$$

where the matrix $\dot{\bar{\mathbf{J}}}(\mathbf{q})$ can be computed analytically using [70]. Thus, equations (5.39) and (5.37), can be represented linearly in the inertial parameters. In this sense, the equation (5.36) can be given as follows:

$$\frac{\gamma}{s+\gamma}(\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}) = \gamma \mathbf{Y_m}(\mathbf{q}, \dot{\mathbf{q}})\pi - \frac{\gamma}{s+\gamma}\mathbf{Y_{mf}}(\mathbf{q}, \dot{\mathbf{q}}, \gamma)\pi, \tag{5.40}$$

where, $\mathbf{Y_m}(\mathbf{q}, \dot{\mathbf{q}})$, and $\mathbf{Y_{mf}}(\mathbf{q}, \dot{\mathbf{q}}, \gamma)$ are two regressors matrices that represent a linear mapping between equation (5.36) terms and the inertial parameters. By combining equations (5.34) and (5.40), the filtered forces are linearly related to the inertial parameters as follows:

$$(\gamma \mathbf{Y_m}(\mathbf{q}, \dot{\mathbf{q}}) + \frac{\gamma}{s+\gamma}(\mathbf{Y_h}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{Y_{mf}}(\mathbf{q}, \dot{\mathbf{q}}, \gamma)))\pi = \mathbf{f}_f. \tag{5.41}$$

For convenience, a matrix $\mathbf{Y_{f0}}(\mathbf{q}, \dot{\mathbf{q}}, \gamma) = \frac{\gamma}{s+\gamma}(\mathbf{Y_h}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{Y_{mf}}(\mathbf{q}, \dot{\mathbf{q}}, \gamma))$ is defined. Thus, a discrete version of (5.41) is given as:

$$\mathbf{Y_f^k}(\mathbf{q^{k-1}}, \dot{\mathbf{q}}^{\mathbf{k-1}}, \gamma, t_s))\pi = \mathbf{f}_f^k, \tag{5.42}$$

where, k and t_s represent the sample index, and sampling time respectively. The discrete filtered regressor $\mathbf{Y_f^k}(\mathbf{q^{k-1}}, \dot{\mathbf{q}}^{\mathbf{k-1}}, \gamma, t_s))$ is obtained using (5.41) such that:

$$\mathbf{Y_f^k}(\mathbf{q^{k-1}}, \dot{\mathbf{q}}^{\mathbf{k-1}}, \gamma, t_s)) = \gamma \mathbf{Y_m^k}(\mathbf{q^{k-1}}, \dot{\mathbf{q}}^{\mathbf{k-1}}) + \mathbf{Y_{f0}^k}(\mathbf{q^{k-1}}, \dot{\mathbf{q}}^{\mathbf{k-1}}, \gamma, t_s), \tag{5.43}$$

where,

$$\mathbf{Y_{f0}^{k}}(\mathbf{q^{k-1}},\dot{\mathbf{q}}^{\mathbf{k-1}},\gamma,t_s) = (1+\gamma t_s)^{-1}(\mathbf{Y_{f0}^{k-1}}(\mathbf{q^{k-2}},\dot{\mathbf{q}}^{\mathbf{k-2}},\gamma,t_s) + \gamma t_s(\mathbf{Y_{h}^{k}}(\mathbf{q^{k-1}},\dot{\mathbf{q}}^{\mathbf{k-1}}) - \mathbf{Y_{mf}^{k}}(\mathbf{q^{k-1}},\dot{\mathbf{q}}^{\mathbf{k-1}},\gamma))). \quad (5.44)$$

From now on, the filtered regressor shown in (5.42) is going to replace the regressor shown in (5.15), for the proceeding analyses and computations.

5.5 Recursive Inertial Parameters Identification

To manage the inclusion of a new data set for a group of samples with a window size of n_w, the optimization problem in (5.33) is modified to admit the acquired samples as follows:

$$\begin{aligned} \hat{\tilde{\boldsymbol{\pi}}}^{W} &= \arg\min_{\forall \tilde{\boldsymbol{\pi}}^{W} \in \mathbb{R}^{m_{i,W}}} \frac{1}{2}||\tilde{\boldsymbol{\Phi}}_W \tilde{\boldsymbol{\pi}}^{W} - \tilde{\boldsymbol{f}}_W||^2 \\ s/t \quad & \mathbf{I}_{P,i}(\tilde{\boldsymbol{\pi}}_i^{W}) + \mathbf{I}_{P,i}^{0} > 0,\ i \in \{1,2,...,n_b\}, \end{aligned} \quad (5.45)$$

where

$$\tilde{\boldsymbol{\Phi}}_W = \begin{bmatrix} \overline{\boldsymbol{\Phi}}_1 \overline{\boldsymbol{P}}_1 \\ \overline{\boldsymbol{\Phi}}_2 \overline{\boldsymbol{P}}_2 \\ \vdots \\ \overline{\boldsymbol{\Phi}}_W \overline{\boldsymbol{P}}_W \end{bmatrix}, \ \tilde{\boldsymbol{f}}_W = \begin{bmatrix} \bar{\mathbf{f}}_1 \\ \bar{\mathbf{f}}_2 \\ \vdots \\ \bar{\mathbf{f}}_W \end{bmatrix}. \quad (5.46)$$

Given a window index $wi \in 1,2,...,W-1,W$ and the total number of included windows W, the matrix $\overline{\boldsymbol{\Phi}}wi \in \mathbb{R}^{nbp,wi \times n_{i,wi}}$ and the vector $\bar{\mathbf{f}}wi \in \mathbb{R}^{nbp,wi}$ represent the regressor and the generalized force vector for any arbitrary window wi. The matrices $\overline{\boldsymbol{\Phi}}wi$ and vectors $\bar{\mathbf{f}}wi$ are obtained separately for each window. Consequently, the linear relation described in equation (5.32) can be reconstructed for any arbitrary window wi as follows:

$$\overline{\boldsymbol{\Phi}}_{wi} \overline{\boldsymbol{\pi}}_{wi} = \bar{\mathbf{f}}_{wi}. \quad (5.47)$$

The symbols $n_{bp,wi}$ and $n_{i,wi}$ represent the size of the base and identifiable parameter sets for window wi, respectively.

Due to phase switching and online implementation, the optimal excitation for each window is never guaranteed. Consequently, the identifiable parameter vector $\overline{\boldsymbol{\pi}}wi$ is different for each window. Hence, the matrix $\overline{\boldsymbol{P}}wi \in \mathbb{R}^{n_{i,wi} \times m_{i,W}}$ is developed to project the regressors of different windows onto a single inertial parameters vector $\tilde{\boldsymbol{\pi}}^W \in \mathbb{R}^{m_{i,W}}$. The vector $\tilde{\boldsymbol{\pi}}^W$ represents the identifiable parameters, augmented by the regressor $\tilde{\boldsymbol{\Phi}}W \in \mathbb{R}^{mb,W \times m_{i,W}}$. The symbol $m_{i,W}$ refers to the size of the identifiable parameter sets using the augmented regressor $\tilde{\boldsymbol{\Phi}}W$. The parameter $mb,W = \sum_{wi=1}^{W} n_{bp,wi}$ represents the sum of the base parameter set sizes for all included windows. The projection matrix $\overline{\boldsymbol{P}}_{wi}$ can be used to map equation (5.47) to $\tilde{\boldsymbol{\pi}}^W$ as follows:

$$\overline{\boldsymbol{\Phi}}_{wi}\overline{\boldsymbol{P}}_{wi}\tilde{\boldsymbol{\pi}}^W = \bar{\mathbf{f}}_{wi}. \tag{5.48}$$

The regressor $\overline{\boldsymbol{\Phi}}_{wi}$ combined with the projection matrix $\overline{\boldsymbol{P}}_{wi}$ and force vector $\bar{\mathbf{f}}_{wi}$ for the included windows constitutes the augmented regressor $\tilde{\boldsymbol{\Phi}}_W \in \mathbb{R}^{m_{b,W} \times m_{i,W}}$ and augmented force vector $\tilde{\boldsymbol{f}}_W \in \mathbb{R}^{m_{b,W}}$ as shown in (5.46).

Using (5.46), the optimization problem given in (5.45) can be rewritten in the following quadratic form:

$$\begin{aligned} \hat{\tilde{\boldsymbol{\pi}}}^W \quad &= \quad \arg \min_{\forall \tilde{\boldsymbol{\pi}}^W \in \mathbb{R}^{m_{i,W}}} \frac{1}{2}\tilde{\boldsymbol{\pi}}^{W\intercal}\overline{\mathbf{Q}}_W\tilde{\boldsymbol{\pi}}^W + \bar{\mathbf{c}}_W^{\intercal}\tilde{\boldsymbol{\pi}}^W \\ &\quad s/t \quad \mathbf{I}_{P,i}(\tilde{\boldsymbol{\pi}}_i^W) + \mathbf{I}_{P,i}^0 > \mathbf{0},\ i \in \{1,2,...,n_b\}, \end{aligned} \tag{5.49}$$

where, the matrices $\overline{\mathbf{Q}}_W \in \mathbb{R}^{m_{i,W} \times m_{i,W}}$, and $\bar{\mathbf{c}}_W \in \mathbb{R}^{m_{i,W}}$, are updated recursively as follows:

$$\overline{\mathbf{Q}}_W = \overline{\mathbf{Q}}_{W-1} + (\overline{\boldsymbol{\Phi}}_W\overline{\boldsymbol{P}}_W)^{\intercal}\overline{\boldsymbol{\Phi}}_W\overline{\boldsymbol{P}}_W \tag{5.50}$$

$$\bar{\mathbf{c}}_W = \bar{\mathbf{c}}_{W-1} - \bar{\mathbf{f}}_W^{\intercal}\overline{\boldsymbol{\Phi}}_W\overline{\boldsymbol{P}}_W, \tag{5.51}$$

in case of no change in the identifiable parameters between samples. For the first window

$$\overline{\mathbf{Q}}_1 = (\overline{\boldsymbol{\Phi}}_1\overline{\boldsymbol{P}}_1)^{\intercal}\overline{\boldsymbol{\Phi}}_1\overline{\boldsymbol{P}}_1 + \varepsilon\mathbf{1}_{m_{i,1}}. \tag{5.52}$$

The parameter ε is a relatively small positive scalar value. The last term in (5.52) is added to avoid the singularity. If the number of the excited parameters increases, the matrix $\overline{\mathbf{Q}}_W$

is updated as follows:

$$\overline{\mathbf{Q}}_W = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix} + \varepsilon \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{\delta m} \end{bmatrix} \tag{5.53}$$

$$\mathbf{Q}_{11} = \overline{\mathbf{Q}}_{W-1} + ((\overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)^\intercal \overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)[1 : m_{i,W-1}][1 : m_{i,W-1}] \tag{5.54}$$

$$\mathbf{Q}_{12} = ((\overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)^\intercal \overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)[1 : m_{i,W-1}][m_{i,W-1} + 1 : m_{i,W}] \tag{5.55}$$

$$\mathbf{Q}_{21} = ((\overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)^\intercal \overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)[m_{i,W-1} + 1 : m_{i,W}][1 : m_{i,W-1}] \tag{5.56}$$

$$\mathbf{Q}_{22} = ((\overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)^\intercal \overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)[m_{i,W-1} + 1 : m_{i,W}][m_{i,W-1} + 1 : m_{i,W}] \tag{5.57}$$

where $\delta m = m_{i,W} - m_{i,W-1}$. The square brackets, in (5.54-5.57), denote the matrix rows and columns, e.g. $\mathbf{A}[r_1 : r_n][c_1 : c_n]$ corresponds to matrix $\mathbf{A}$ from row r_1 to r_n and from column c_1 to c_n. To enable the update criterion given by (5.53), the matrix $\overline{\boldsymbol{P}}_{wi}$ is designed so that it permutes the recently excited parameters equivalent columns as the last columns of $\tilde{\mathbf{\Phi}}_W$.

The recursive structure of $\overline{\mathbf{Q}}_W$ can be exploited to recursively compute $\overline{\mathbf{Q}}_W^{-1}$ as function of $\overline{\mathbf{Q}}_{W-1}^{-1}$. The same algorithms represented in [89] were used, which are based on the Sherman-Morrison-Woodbury formula to recursively compute the inverse of $\overline{\mathbf{Q}}_W$. In case of no change in the matrix order, algorithm 1 in Appendix B is used such that:

$$\overline{\mathbf{Q}}_W^{-1} = recI(\overline{\mathbf{Q}}_{W-1}^{-1}, (\overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)). \tag{5.58}$$

In the case of matrix order increase algorithm 2 in Appendix B is used as follows:

$$\begin{aligned} \overline{\mathbf{Q}}_W^{-1} &= Add(\overline{\mathbf{Q}}_{11}^{-1}, \overline{\mathbf{Q}}_W), \\ \overline{\mathbf{Q}}_{11}^{-1} &= recI(\overline{\mathbf{Q}}_{W-1}^{-1}, (\overline{\mathbf{\Phi}}_W \overline{\boldsymbol{P}}_W)). \end{aligned} \tag{5.59}$$

5.6 Recursive Identifiability Projection Matrix (RIPM)

Similarly to what is shown in subsection 5.3.3, the window *wi* regressor $\mathbf{Y}_{wi} \in \mathbb{R}^{n_w n_{dof} \times 10 n_b}$ can be decomposed into two separate matrices. One with columns corresponding to the

base inertial parameters denoted as $\mathbf{Y}_b^{wi} \in \mathbb{R}^{n_w n_{dof} \times n_{bp,wi}}$, and second with columns corresponding to the linearly coupled inertial parameters, namely $\mathbf{Y}_l^{wi} \in \mathbb{R}^{n_w n_{dof} \times (n_{i,wi} - n_{bp,wi})}$, such that:

$$\mathbf{Y}_{wi}\boldsymbol{P}_{wi} = [\mathbf{Y}_b^{wi}\ \mathbf{Y}_l^{wi}]. \tag{5.60}$$

The matrix $\boldsymbol{P}_{wi} \in \mathbb{R}^{10n_b \times n_{i,wi}}$ is a binary matrix that can be decomposed into two matrices such that:

$$\boldsymbol{P}_{wi} = \mathbf{V}_{wi}\mathbf{P}_{\mathbf{r},wi}, \tag{5.61}$$

where, $\mathbf{V}_{wi} \in \mathbb{R}^{10n_b \times n_{i,wi}}$ is a matrix that stores the standard basis vectors of the non-zero columns of $\mathbf{Y}_{wi}$, and $\mathbf{P}_{\mathbf{r},wi} \in \mathbb{R}^{n_{i,wi} \times n_{i,wi}}$ is a permutation matrix that is used to rearrange the columns corresponding to basis and linearly coupled inertial parameters; see (5.60). Due to the structure of $\boldsymbol{P}_{wi}$ given in (5.61), its outer product is expressed as:

$$\boldsymbol{P}_{wi}\boldsymbol{P}_{wi}^{\mathsf{T}} = \mathbf{V}_{wi}\mathbf{V}_{wi}^{\mathsf{T}}, \tag{5.62}$$

which generates a binary matrix, with a set of ones located only on the diagonal elements. The ones columns indexes correspond to the identifiable parameters indexes in $\boldsymbol{\pi}$. Therefore, $\boldsymbol{P}_{wi}^{\mathsf{T}}$ can be used in two different ways. Firstly, it can be used to invert (5.60)

$$[\mathbf{Y}_b^{wi}\ \mathbf{Y}_l^{wi}]\boldsymbol{P}_{wi}^{\mathsf{T}} = \mathbf{Y}_{wi}. \tag{5.63}$$

Secondly, it can be used to project the identifiable subspace $\overline{\boldsymbol{\pi}}^{wi} \in \mathbb{R}^{n_{i,wi}}$ by $\overline{\boldsymbol{\Phi}}_{wi}$ onto the inertial parameters vector $\boldsymbol{\pi}$ as follows:

$$\boldsymbol{\pi}_{i,wi}^{\perp} = \boldsymbol{P}_{wi}\overline{\boldsymbol{\pi}}^{wi}, \tag{5.64}$$

where $\boldsymbol{\pi}_{i,wi}^{\perp}$ is the projection of $\overline{\boldsymbol{\pi}}^{wi}$ onto $\boldsymbol{\pi}$. From equation (5.64) and (5.47) the following equation can be concluded:

$$\overline{\boldsymbol{\Phi}}_{wi}\boldsymbol{P}_{wi}^{\mathsf{T}}\boldsymbol{\pi} = \bar{\mathbf{f}}_{wi}. \tag{5.65}$$

The set of identifiable parameters by $\overline{\boldsymbol{\Phi}}_{wi}$ is defined as follows:

$$\bar{\boldsymbol{I}}_{wi} := \{\overline{\boldsymbol{\pi}}^{wi} \in \mathbb{R}^{n_{i,wi}} \mid n_{i,wi} \leq m_{i,W} \leq 10n_b\}. \tag{5.66}$$

For the last two consecutive windows W and $W-1$, the identifiable parameters set $\bar{\boldsymbol{I}}_W = \bar{\boldsymbol{I}}_{W-1}$ are assumed to be equal. A projection matrix can be developed recursively using $\boldsymbol{P}_{W-1}$ to project the columns of $\overline{\boldsymbol{\Phi}}_W$ onto $\overline{\boldsymbol{\pi}}^{W-1}$, via (5.65) and (5.64) such that:

$$\overline{\boldsymbol{\Phi}}_W \boldsymbol{P}_W^\intercal \boldsymbol{P}_{W-1} \overline{\boldsymbol{\pi}}^{W-1} = \bar{\mathbf{f}}_W. \tag{5.67}$$

Using (5.47) and (5.48), $\overline{\boldsymbol{\pi}}^{W-1}$ can be substituted by $\tilde{\boldsymbol{\pi}}^{W-1}$, as follows:

$$\overline{\boldsymbol{\Phi}}_W \boldsymbol{P}_W^\intercal \boldsymbol{P}_{W-1} \overline{\boldsymbol{P}}_{W-1} \tilde{\boldsymbol{\pi}}^{W-1} = \bar{\mathbf{f}}_W. \tag{5.68}$$

By defining a set of identifiable parameters by $\tilde{\boldsymbol{\Phi}}_W$ as follow:

$$\tilde{\boldsymbol{I}}_W := \{\tilde{\boldsymbol{\pi}}^W \in \mathbb{R}^{m_{i,W}} \mid n_{i,wi} \leq m_{i,W} \leq 10n_b\} \tag{5.69}$$

it can be concluded that the use of equation (5.68) may not suffice to select all the identifiable parameters in case of $\bar{\boldsymbol{I}}_W \nsubseteq \tilde{\boldsymbol{I}}_{W-1}$. In such a case, one can define a projection matrix per window given as follow:

$${}^{\cap}\boldsymbol{P}_W = \boldsymbol{P}_W^\intercal \boldsymbol{P}_{W-1} \overline{\boldsymbol{P}}_{W-1} \tag{5.70}$$

where, matrix ${}^{\cap}\boldsymbol{P}_W \in \mathbb{R}^{n_{i,W} \times n_{i,W-1}}$ projects the inertial parameters located within $\bar{\boldsymbol{I}}_W \cap \tilde{\boldsymbol{I}}_{W-1}$ onto $\tilde{\boldsymbol{\pi}}^{W-1}$. In this case, a set of zeros rows do exist in ${}^{\cap}\boldsymbol{P}_W$. There zero rows are corresponding to $\bar{\boldsymbol{I}}_W \backslash \tilde{\boldsymbol{I}}_{W-1}$. Backslash $\backslash$ denotes exclusion math operator, e.g. $I_1 \backslash I_2$ returns the elements included in set I_1 and excluded from set I_2. A projection matrix $\overline{\boldsymbol{P}}_W$ that accounts for $\bar{\boldsymbol{I}}_W \backslash \tilde{\boldsymbol{I}}_{W-1}$ can be developed by perturbing the zero rows at $\overline{\boldsymbol{P}}_W$ last columns as follow:

$$\overline{\boldsymbol{P}}_W = [{}^{\cap}\boldsymbol{P}_W \;\; \mathrm{C}(\mathrm{N}({}^{\cap}\boldsymbol{P}_W {}^{\cap}\boldsymbol{P}_W^\intercal))]. \tag{5.71}$$

For the first window ${}^{\cap}\boldsymbol{P}_1$ does not exist and $\overline{\boldsymbol{P}}_1 = \mathbf{1}_{n_{i,1}}$ is set to identity matrix. In (5.71), the function $\mathrm{N}({}^{\cap}\boldsymbol{P}_W {}^{\cap}\boldsymbol{P}_W^\intercal) = \mathbf{1}_{n_{i,W}} - {}^{\cap}\boldsymbol{P}_W {}^{\cap}\boldsymbol{P}_{wi}^\intercal$ returns the null space matrix of ${}^{\cap}\boldsymbol{P}_W$, and $\mathrm{C}(\bullet)$

returns the non-zero columns of the input matrix. In (5.70) and (5.71), matrix $\overline{\boldsymbol{P}}_W$ acts as a column permutation for $\overline{\boldsymbol{\Phi}}_W$, where columns are rearranged into two categories. The first category represents the identifiable parameters located within $\bar{\boldsymbol{I}}_W \cap \tilde{\boldsymbol{I}}_{W-1}$, while the second category represents the identifiable parameters located within $\bar{\boldsymbol{I}}_W \backslash \tilde{\boldsymbol{I}}_{W-1}$.

A secondary use of the permutation matrix obtained in (5.71) is projecting $\tilde{\boldsymbol{\pi}}^W$ onto the inertial parameters space $\boldsymbol{\pi}$. Using (5.45), (5.46), and (5.64), the projection is given as:

$$\boldsymbol{\pi}_{i,W}^{\perp} = \boldsymbol{P}_W \overline{\boldsymbol{P}}_W \tilde{\boldsymbol{\pi}}^W, \tag{5.72}$$

where $\boldsymbol{\pi}_{i,W}^{\perp}$ is the projection of $\tilde{\boldsymbol{\pi}}^W$ onto $\boldsymbol{\pi}$. The projection given in (5.72) may not be sufficient to project $\tilde{\boldsymbol{\pi}}^W$ onto $\boldsymbol{\pi}$ for the case of $\tilde{\boldsymbol{I}}_{W-1} \backslash \bar{\boldsymbol{I}}_W \neq \phi$. In such a case, a set of zero columns exist in ${}^{\cap}\boldsymbol{P}_W$. That columns corresponding to $\tilde{\boldsymbol{I}}_{W-1} \backslash \bar{\boldsymbol{I}}_W$. The projection shown by (5.72) is modified to account for $\tilde{\boldsymbol{I}}_{W-1} \backslash \bar{\boldsymbol{I}}_W$ by perturbing these columns for the new projection matrix $\tilde{\boldsymbol{P}}_W$ this can be achieved as follows:

$$\tilde{\boldsymbol{P}}_W = [\boldsymbol{P}_W \;\; \mathrm{C}(\boldsymbol{P}_W^-)] \begin{bmatrix} {}^{\cap}\boldsymbol{P}_W & \mathrm{C}(\mathrm{N}({}^{\cap}\boldsymbol{P}_W {}^{\cap}\boldsymbol{P}_W^{\mathsf{T}})) \\ \mathrm{R}(\mathrm{N}({}^{\cap}\boldsymbol{P}_W^{\mathsf{T}} {}^{\cap}\boldsymbol{P}_W)) & \boldsymbol{0} \end{bmatrix} \tag{5.73}$$

where the function $\mathrm{R}(\bullet)$ returns the non-zero rows of the input matrix. The Function $\mathrm{N}({}^{\cap}\boldsymbol{P}_W^{\mathsf{T}} {}^{\cap}\boldsymbol{P}_W) = \boldsymbol{1}_{n_{i,W-1}} - {}^{\cap}\boldsymbol{P}_W^{\mathsf{T}} {}^{\cap}\boldsymbol{P}_W$ returns the null space matrix of ${}^{\cap}\boldsymbol{P}_W^{\mathsf{T}}$. The matrix $\boldsymbol{P}_W^-$ represents the space of $\tilde{\boldsymbol{I}}_{W-1} \backslash \bar{\boldsymbol{I}}_W$. Exploiting the binary form of the projector matrices, $\boldsymbol{P}_W^-$ is computed as:

$$\boldsymbol{P}_W^- = (\boldsymbol{P}_1 \boldsymbol{P}_1^{\mathsf{T}} \oplus \boldsymbol{P}_2 \boldsymbol{P}_2^{\mathsf{T}} \; ... \; \oplus \boldsymbol{P}_W \boldsymbol{P}_W^{\mathsf{T}}) - \boldsymbol{P}_W \boldsymbol{P}_W^{\mathsf{T}}, \tag{5.74}$$

where the symbol $\oplus$ represents, the logical OR operator, for which the terms within brackets in (5.74) can be easily obtained recursively. The term $\mathrm{C}(\boldsymbol{P}_W^-)$ is added to project $\tilde{\boldsymbol{I}}_{W-1} \backslash \bar{\boldsymbol{I}}_W$ onto the space of $\boldsymbol{\pi}$. Using, (5.73), a projection can be obtained as:

$$\boldsymbol{\pi}_{i,W}^{\perp} = \tilde{\boldsymbol{P}}_W \tilde{\boldsymbol{\pi}}^W. \tag{5.75}$$

To include all the identifiable parameters by $\tilde{\boldsymbol{\Phi}}_{W-1}$ equation (5.70) is modified as:

$${}^{\cap}\boldsymbol{P}_W = \boldsymbol{P}_W^{\mathsf{T}} \tilde{\boldsymbol{P}}_{W-1}. \tag{5.76}$$

5.7 Recursive SDP Interior Point Method (SDP-IPM)

5.7.1 Interior Point Method

To make use of the recursive procedure obtained in sections 5.5 and 5.6, a semidefinite programming framework based on the interior point method is proposed to solve (5.49); see [54, 88]. The inequality constraint given in (5.49) can be replaced by the block diagonal matrix: $\mathbf{I}_P^D(\tilde{\boldsymbol{\pi}}^W) = diag(\mathbf{I}_{P,1}(\tilde{\boldsymbol{\pi}}_1^W), \mathbf{I}_{P,2}(\tilde{\boldsymbol{\pi}}_2^W), ..., \mathbf{I}_{P,n_b}(\tilde{\boldsymbol{\pi}}_{n_b}^W))$, where $\mathbf{I}_P^D(\tilde{\boldsymbol{\pi}}^W) \in \mathbb{R}^{4n_b \times 4n_b}$. Exploiting the LMI form, the matrix $\mathbf{I}_P^D(\tilde{\boldsymbol{\pi}}^W)$ can be represented linearly with respect to $\tilde{\pi}_l^W$ as follows:

$$\mathbf{I}_P^D(\tilde{\boldsymbol{\pi}}^W) = \sum_{l=1}^{m_{i,W}} \tilde{\pi}_l^W \mathbf{G}_l \tag{5.77}$$

where, $\tilde{\pi}_l^W$ represents the l^{th} component in $\tilde{\boldsymbol{\pi}}^W$, and $\mathbf{G}_l \in \mathbb{R}^{4n_b \times 4n_b}$ $l = 1, ..., m_{i,W}$, are linearly independent symmetric matrices.

Using (5.77) the optimization problem (5.49) can be rewritten as:

$$\hat{\tilde{\boldsymbol{\pi}}}^W = \arg \min_{\forall \tilde{\boldsymbol{\pi}}^W \in \mathbb{R}^{m_{i,W}}} \frac{1}{2} \tilde{\boldsymbol{\pi}}^{W\intercal} \overline{\mathbf{Q}}_W \tilde{\boldsymbol{\pi}}^W + \bar{\mathbf{c}}_W^\intercal \tilde{\boldsymbol{\pi}}^W \tag{5.78}$$

$$s/t \quad \sum_{l=1}^{n_{i,W}} \tilde{\pi}_l^W \mathbf{G}_l + \mathbf{I}_0^D = \mathbf{Z} \tag{5.79}$$

$$\mathbf{Z} > \mathbf{0}, \tag{5.80}$$

$$\mathbf{A}_{eq} \tilde{\boldsymbol{\pi}}^W = \mathbf{0} \tag{5.81}$$

where, $\mathbf{Z} \in \mathbb{R}^{4n_b \times 4n_b}$ represents a slack variable matrix while $\mathbf{I}_0^D \in \mathbb{R}^{4n_b \times 4n_b}$ is a block diagonal matrix given as: $\mathbf{I}_0^D = diag(\mathbf{I}_{P,1}^0, \mathbf{I}_{P,2}^0, ..., \mathbf{I}_{P,n_b}^0)$. Linear equality constraints, with full row rank $\mathbf{A}_{eq} \in \mathbb{R}^{m \times m_{i,W}}$ matrix $\forall m < m_{i,W}$, are included to impose the symmetrical structure of the bipedal exoskeleton inertial parameters. The associated Lagrangian with (5.78) is given as:

$$\underline{\mathscr{L}} = \frac{1}{2} \tilde{\boldsymbol{\pi}}^{W\intercal} \overline{\mathbf{Q}}_W \tilde{\boldsymbol{\pi}}^W + \bar{\mathbf{c}}_W^\intercal \tilde{\boldsymbol{\pi}}^W - \langle \sum_{l=1}^{m_{i,W}} \tilde{\pi}_l^W \mathbf{G}_l + \mathbf{I}_0^D - \mathbf{Z}, \mathbf{X} \rangle$$

$$- \mu log\,(det(\mathbf{Z})) - \mathbf{y}^\intercal \mathbf{A}_{eq} \tilde{\boldsymbol{\pi}}^W, \tag{5.82}$$

where a logarithmic barrier function $log(\bullet)$ is used to impose positive definiteness on $\mathbf{Z}$. The function $det(\mathbf{Z})$ is determinant of $\mathbf{Z}$. The function $\langle \mathbf{A}, \mathbf{B} \rangle$ is the trace function of the product of any two arbitrary matrices $\mathbf{A}$ and $\mathbf{B}$. The vector $\mathbf{y} \in \mathbb{R}^m$ is Lagrangian multiplier, and $\mathbf{X} \in \mathbb{R}^{4n_b \times 4n_b}$ is a positive definite symmetric matrix.

The parameter μ is a positive scalar that is used to include the barrier function within the objective function. It is initiated with a large value to give priority to the barrier function resolution, preventing $\mathbf{Z}$ from approaching its boundary. Then gradually, over the optimization iterations, the parameter μ is reduced to allow the algorithm to minimize the original objective function. Using (5.82), the first-order optimality conditions are obtained as follows:

$$\begin{aligned} \nabla_{\pi} \underline{\mathscr{L}} =& \overline{\mathbf{Q}}_W \tilde{\boldsymbol{\pi}}^W + \bar{\mathbf{c}}_W^{\intercal} - \left[\langle \mathbf{G}_1, \mathbf{X} \rangle \ \langle \mathbf{G}_2, \mathbf{X} \rangle \ ... \ \langle \mathbf{G}_{n_{i,W}}, \mathbf{X} \rangle \right]^{\intercal} \\ & - \mathbf{A}_{eq}^{\intercal} \mathbf{y} = \mathbf{r}_{\pi} \end{aligned} \tag{5.83}$$

$$\nabla_X \underline{\mathscr{L}} = -(\sum_{l=1}^{n_{i,W}} \tilde{\pi}_l^W \mathbf{G}_l + \mathbf{I}_0^D) + \mathbf{Z} = \mathbf{R}_X \tag{5.84}$$

$$\nabla_Z \underline{\mathscr{L}} = -\mu \mathbf{Z}^{-1} + \mathbf{X} = \mathbf{R}_Z \tag{5.85}$$

$$\nabla_y \underline{\mathscr{L}} = -\mathbf{A}_{eq} \tilde{\boldsymbol{\pi}}^{wi} = \mathbf{r}_y \tag{5.86}$$

where, $\mathbf{r}_{\pi} \in \mathbb{R}^{m_{i,W}}$ and $\mathbf{r}_y \in \mathbb{R}^m$ are two vectors that represent the residuals of the first and fourth optimality conditions. The matrices $\mathbf{R}_X \in \mathbb{R}^{4n_b \times 4n_b}$ and $\mathbf{R}_Z \in \mathbb{R}^{4n_b \times 4n_b}$ are the residuals of the second and third optimality conditions. In literature, the condition (5.85) is denoted as the complementary condition, and the way of handling it determines the optimization method. To allow the optimization variables to follow a central path that satisfies the complementary condition, the matrices $\mathbf{X}$ and $\mathbf{Z}$ have to justify the centering condition such that:

$$\mathbf{XZ} = \mu \mathbf{1}_{4n_b}, \tag{5.87}$$

where the matrices $\mathbf{X}$ and $\mathbf{Z}$ are commute. In this work, XZ+ZX method is used for direct Newton method application [54]. In this method, the complementary condition given by

(5.85) is replaced by the following equality:

$$\mathbf{R}_Z = \mathbf{XZ} + \mathbf{ZX} - 2\mu\mathbf{1}_{4n_b} \tag{5.88}$$

which satisfies the centering condition given in (5.87). To obtain a KKT system, the function $\mathbf{a} = svec(\mathbf{A})$ is introduced which maps an arbitrary symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ to a vector $\mathbf{a} \in \mathbb{R}^{\frac{n(n+1)}{2}}$. $\mathbf{A} = smat(\mathbf{a})$ is the inverse of $svec(\bullet)$; see Appendix B. A Newton step that satisfies the first-order optimality condition for the XZ+ZX method is obtained by solving the following KKT system:

$$\begin{bmatrix} \overline{\mathbf{Q}}_W & -\overline{\mathbf{G}} & \mathbf{0} & -\mathbf{A}_{eq}^{\mathsf{T}} \\ -\overline{\mathbf{G}}^{\mathsf{T}} & \mathbf{0} & \mathbf{1}_{n_{sym}} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_S & \mathbf{F}_S & \mathbf{0} \\ -\mathbf{A}_{eq} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \triangle\tilde{\boldsymbol{\pi}}^W \\ \triangle\mathbf{x} \\ \triangle\mathbf{z} \\ \triangle\mathbf{y} \end{bmatrix} = \begin{bmatrix} -\mathbf{r}_\pi \\ -\mathbf{r}_x \\ -\mathbf{r}_z \\ -\mathbf{r}_y \end{bmatrix} \tag{5.89}$$

where the matrix $\overline{\mathbf{G}} \in \mathbb{R}^{m_{i,W}\times n_{sym}}$ is obtained as follows:

$\overline{\mathbf{G}} = [svec(\mathbf{G}_1)\ svec(\mathbf{G}_2)\ ...\ svec(\mathbf{G}_{m_{i,W}})]^{\mathsf{T}}$. The matrices $\mathbf{E}_S \in \mathbb{R}^{n_{sym}\times n_{sym}}$ and $\mathbf{F}_S \in \mathbb{R}^{n_{sym}\times n_{sym}}$ are obtained using a symmetric Kronecker product math operation $\otimes_s$ such that:

$$\mathbf{E}_S = \mathbf{Z} \otimes_s \mathbf{1}_{n_{sym}}, \quad \mathbf{F}_S = \mathbf{X} \otimes_s \mathbf{1}_{n_{sym}}. \tag{5.90}$$

See Appendix B for the $\otimes_s$ operator details. Note that $n_{sym} = 2n_b(4n_b+1)$. The residuals $\mathbf{r}_x$ and $\mathbf{r}_z$ and the Newton steps $\triangle\mathbf{x}$ and $\triangle\mathbf{z}$ are transformed to equivalent matrix forms as follows:

$$\mathbf{r}_x = svec(\mathbf{R}_X), \quad \mathbf{r}_z = svec(\mathbf{R}_Z) \tag{5.91}$$

$$\triangle\mathbf{X} = smat(\triangle\mathbf{x}), \quad \triangle\mathbf{Z} = smat(\triangle\mathbf{z}) \tag{5.92}$$

An interior point method (IPM) based optimization algorithm is developed to solve (5.78) using the first order optimality conditions (5.83), (5.84), (5.86), and (5.88) and KKT system (5.89), is given in table 6:

Table 6: Semi-Definite Programming Interior Point Method (SDP-IPM)

Initialize SDP-IPM

1-Feasibly initialize $\tilde{\boldsymbol{\pi}}_0^W,\ \mathbf{y}_0,\ \mathbf{X}_0 > \mathbf{0},\ \mathbf{Z}_0 > \mathbf{0}$

2-Compute the equivalent vector form

$$\mathbf{x}_0 = svec(\mathbf{X}_0),\ \mathbf{z}_0 = svec(\mathbf{Z}_0)$$

3-Obtain the initial residuals $(\mathbf{r}_\pi^0,\ \mathbf{r}_x^0,\ \mathbf{r}_y^0)$ using (5.83, 5.84, 5.86, and 5.91)

4- Choose the step size factor $0 < \alpha < 1$, and barrier reduction parameter $0 < \sigma < 1$

5- Select sufficiently small positive scalers tolerance parameters $(\varepsilon_\pi, \varepsilon_x, \varepsilon_y, \varepsilon_o)$

6-Initialize iteration index $h = 0$

SDP-IPM Main Loop

$while(\frac{||\mathbf{r}_\pi^h||}{1+||\bar{\mathbf{c}}_W||} \leq \varepsilon_\pi, \frac{||\mathbf{r}_x^h||}{1+||\mathbf{z}_h||} \leq \varepsilon_x, ||\mathbf{r}_y|| \leq \varepsilon_y,$
$\frac{\langle \mathbf{X}_h, \mathbf{Z}_h \rangle / 4n_b}{1+|\frac{1}{2}\tilde{\boldsymbol{\pi}}_h^{W\mathsf{T}} \bar{\mathbf{Q}}_W \tilde{\boldsymbol{\pi}}_h^W + \bar{\mathbf{c}}_W^\mathsf{T} \tilde{\boldsymbol{\pi}}_h^W|} \leq \varepsilon_o)$

1-Update the barrier parameter $\mu_h = \sigma \langle \mathbf{X}_h, \mathbf{Z}_h \rangle / 4n_b$

2-Solve the KKT system (5.89) to obtain Newton step $(\triangle\tilde{\boldsymbol{\pi}}^W, \triangle\mathbf{x}, \triangle\mathbf{z}, \triangle\mathbf{y})$

3-Update the optimization variables

$\tilde{\boldsymbol{\pi}}_{h+1}^W = \tilde{\boldsymbol{\pi}}_h^W + \alpha\triangle\tilde{\boldsymbol{\pi}}^W$

$\mathbf{x}_{h+1} = \mathbf{x}_h + \alpha\triangle\mathbf{x}$

$\mathbf{z}_{h+1} = \mathbf{z}_h + \alpha\triangle\mathbf{z}$

$\mathbf{y}_{h+1} = \mathbf{y}_h + \alpha\triangle\mathbf{y}$

4-Compute the equivalent matrices form

$$\mathbf{X}_{h+1} = smat(\mathbf{x}_{h+1}),\ \mathbf{Z}_{h+1} = smat(\mathbf{z}_{h+1})$$

5–Obtain new residuals values $(\mathbf{r}_\pi^h,\ \mathbf{r}_x^h,\ \mathbf{r}_y^h,\ \mathbf{r}_z^h)$ using (5.83, 5.84, 5.86, 5.88, and 5.91)

6-Update iteration index $h = h + 1$

end

5.7.2 KKT system Resolution

In most recursive least squares algorithms, it is required to compute the matrix inversions recursively. Hence, the solution of (5.89) is determined in terms of $\overline{\mathbf{Q}}_{wi}^{-1}$, which can be obtained recursively using (5.58), (5.59).

Given the KKT system, in (5.89), the KKT matrix can be reduced to two by two block

matrix by eliminating the second and third rows as follows:

$$\begin{bmatrix} \overline{\mathbf{H}} & -\mathbf{A}_{eq}^{\intercal} \\ -\mathbf{A}_{eq} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \triangle\tilde{\boldsymbol{\pi}}^{W} \\ \triangle\mathbf{y} \end{bmatrix} = \begin{bmatrix} -\overline{\mathbf{G}}\mathbf{E}_{S}^{-1}(\mathbf{r}_z - \mathbf{F}\mathbf{r}_x) - \mathbf{r}_{\pi} \\ -\mathbf{r}_y \end{bmatrix}, \tag{5.93}$$

and

$$\triangle\mathbf{z} = \overline{\mathbf{G}}\triangle\tilde{\boldsymbol{\pi}}^{W} - \mathbf{r}_x, \tag{5.94}$$

$$\triangle\mathbf{x} = -\mathbf{E}_{S}^{-1}(\mathbf{F}_{S}\overline{\mathbf{G}}^{\intercal}\triangle\tilde{\boldsymbol{\pi}}^{W} + (\mathbf{r}_z - \mathbf{F}_{S}\mathbf{r}_x)), \tag{5.95}$$

where, the matrix $\overline{\mathbf{H}}$ is as follows: $\overline{\mathbf{H}} = \overline{\mathbf{Q}}_{wi} + \overline{\mathbf{G}}\mathbf{E}_{S}^{-1}\mathbf{F}_{S}\overline{\mathbf{G}}^{\intercal}$. Using Schur complement, the solution of (5.93) can be obtained in terms of $\overline{\mathbf{H}}^{-1}$ such that:

$$\begin{bmatrix} \triangle\tilde{\boldsymbol{\pi}}^{W} \\ \triangle\mathbf{y} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{H}}^{-1} + \overline{\mathbf{H}}^{-1}\mathbf{A}_{eq}^{\intercal}\mathbf{S}_{c}^{-1}\mathbf{A}_{eq}\overline{\mathbf{H}}^{-1} & \overline{\mathbf{H}}^{-1}\mathbf{A}_{eq}^{\intercal}\mathbf{S}_{c}^{-1} \\ \mathbf{S}_{c}^{-1}\mathbf{A}_{eq}\overline{\mathbf{H}}^{-1} & \mathbf{S}_{c}^{-1} \end{bmatrix} \begin{bmatrix} -\overline{\mathbf{G}}\mathbf{E}_{S}^{-1}(\mathbf{r}_z - \mathbf{F}_{S}\mathbf{r}_x) - \mathbf{r}_{\pi} \\ -\mathbf{r}_y \end{bmatrix}$$

where, Schur complement $\mathbf{S}_c$ is $\mathbf{S}_c = (-\mathbf{A}_{eq}\overline{\mathbf{H}}^{-1}\mathbf{A}_{eq}^{\intercal})$. The inverse of the matrix $\overline{\mathbf{H}}$ is obtained using the Kenneth-Miller inverse of the sum of matrices algorithm [91] as a function of $\overline{\mathbf{Q}}_{W}^{-1}$, such that:

$$\overline{\mathbf{H}}^{-1} = KM(\overline{\mathbf{Q}}_{W}^{-1}, \overline{\mathbf{G}}\overline{\mathbf{M}}), \tag{5.96}$$

where, the matrix $\overline{\mathbf{M}}$ is as follows: $\overline{\mathbf{M}} = \mathbf{E}_{S}^{-1}\mathbf{F}_{S}\overline{\mathbf{G}}^{\intercal}$. For further details about the Kenneth-Miller formula; see Appendix B, algorithm 4.

To compute matrix $\overline{\mathbf{M}}$, each column $\mathbf{m}_c$ is computed distinctly such that:

$$\mathbf{m}_c = svec(\mathbf{M}_c), \tag{5.97}$$

where, $c = 1, 2, ..., m_{i,W}$. Matrix $\mathbf{M}_c$ is obtained by solving the Lyapunov equation

$$\mathbf{Z}\mathbf{M}_c + \mathbf{M}_c\mathbf{Z} = \mathbf{X}\mathbf{G}_c + \mathbf{G}_c\mathbf{X}, \tag{5.98}$$

To solve $\mathbf{S}_c^{-1}$ in (5.96), **QR** decomposition of $\mathbf{A}_{eq}^{\intercal}$ is obtained such that:

$$\mathbf{Q}_{eq}\begin{bmatrix} \mathbf{R}_{eq} \\ \mathbf{0} \end{bmatrix} = qr(\mathbf{A}_{eq}^{\intercal}), \tag{5.99}$$

where $\mathbf{Q}_{eq} \in \mathbb{R}^{m_{i,W} \times m_{i,W}}$ orthogonal matrix, and $\mathbf{R}_{eq} \in \mathbb{R}^{m \times m}$ upper triangle matrix. Using (5.99), Schur complement inverse is given as:

$$\mathbf{S}_c^{-1} = \mathbf{R}_{eq}^{-1} RM(\mathbf{Q}_{eq} \overline{\mathbf{H}} \mathbf{Q}_{eq}^{\intercal}, m_{i,W} - m) \mathbf{R}_{eq}^{-T} \tag{5.100}$$

where, the function $RM(\mathbf{A}^{-1}, n)$ is used to return the upper left square submatrix of $\mathbf{A}$ inverse after removing n rows and columns from $\mathbf{A}$ in terms of $\mathbf{A}^{-1}$; see Appendix B, algorithm 3.

5.8 Inertial Parameters Identification Algorithm Simulation Results

To verify the feasibility of the proposed identification algorithm, a series of dynamic walking simulations were conducted using the 3D dynamic RaiSim simulator [57]; see chapter 2.

5.8.1 Forward Walking Pattern Generation

Twelve different human-robot coupled models are utilized to simulate forward dynamic walking motion. Each simulation was repeated 6 times with different forward velocities. In each simulation run, the walking parameters were kept the same for the single and double support periods; they are given as 0.8 and 0.2 s, respectively, and the maximum foot clearance was 3 cm. The forward velocity was set to 0.04, 0.06, 0.08, 0.1, 0.12 and 0.14 m/s.

The identification algorithm is evaluated based on the data collected every sampling period; at 1 kHz. Meanwhile, the walking trajectory generation is updated at 20 kHz. Multi-threading functions were used to execute the identification algorithm at $(4-10\,Hz)$. The plots provided in this section are shown with a sampling time of 10 ms. C language was used in addition to GNU Scientific Library to solve the linear algebra functions [62].

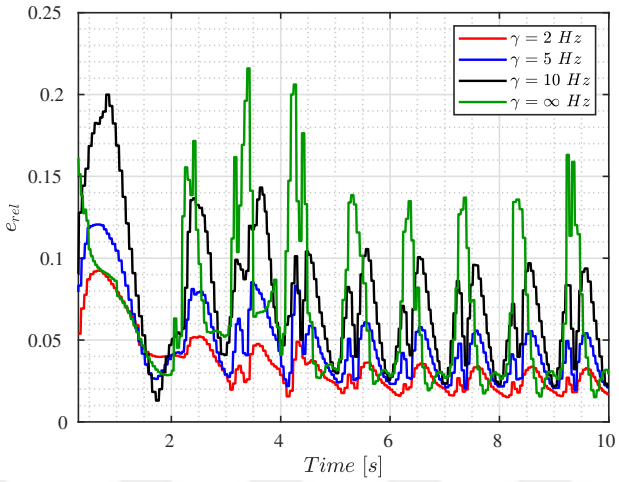

Figure 57: Relative error e_{rel} for four different simulation runs with different filter cutoff frequency γ.

5.8.2 Discussion

In this subsection, the feasibility of the proposed algorithm is discussed which are among 72 distinct simulation runs. To keep figures readable, not all the results are plotted. Observing Fig. 57, the relative error e_{rel} computed using equation (5.42) and (5.75) are expressed as below:

$$e_{rel} = \frac{||\mathbf{Y}_f^k(\mathbf{q}_{act}^{k-1}, \dot{\mathbf{q}}_{act}^{k-1}, \gamma, t_s))\tilde{\boldsymbol{P}}_W \hat{\tilde{\boldsymbol{\pi}}}^W - \mathbf{f}_f^k||}{||\mathbf{f}_f^k||}. \tag{5.101}$$

It is used as a metric to evaluate the contribution of the filtered regressor in the proposed algorithm. The subscript *act* denotes the actual measured signal.

The efficacy of the filtered regressor shows a significant improvement in the relative error due to the reduction in the filter cutoff frequency γ. As it is being reduced, the effect of the contact transitions on the identification algorithm accuracy is mitigated. Hence, in the next simulation experiments, the value of $\gamma = 2\ Hz$ is fixed.

To evaluate the online applicability the computational overhead is evaluated based on the computation time t_c to resolve the algorithm. Fig. 58 depicts t_c for four different subjects. Each subject is simulated 6 times with different walking speeds as previously stated. The acquired data are processed and displayed as, mean±SD with shaded regions.

The mean variations were denoted via solid lines. The computation time varies within $(0.1 \sim 0.25\ s)$, which can be handled online via multi-threaded programming.

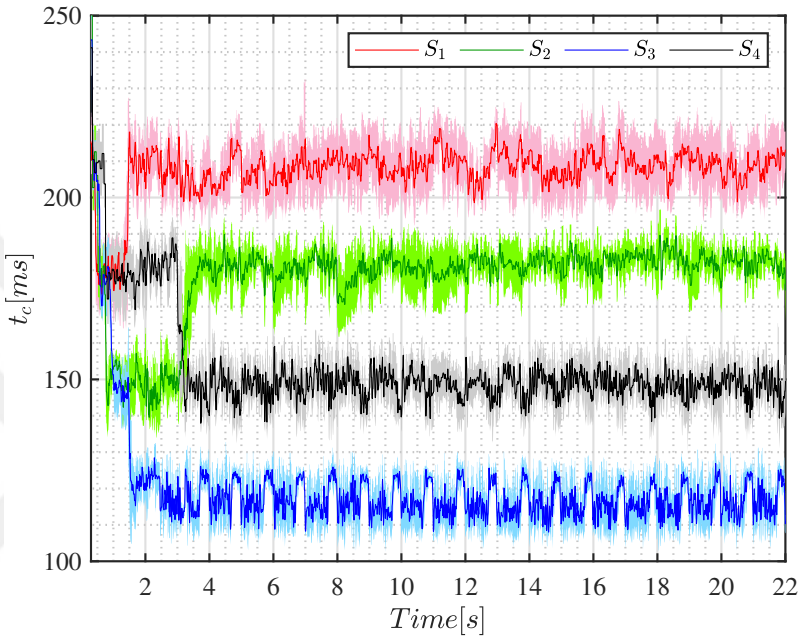

Figure 58: Computation time t_c for four different subjects walking with different velocities over different simulation runs.

The accuracy of the proposed identification algorithm is verified for four human subjects as shown in Fig. 59. The acquired relative error e_{rel} for the different subjects is processed and displayed as, mean$\pm$SD with shaded regions. Its mean variations were denoted via solid lines. The given accuracy levels are settled within acceptable values. Moreover, it converges to its minimal value within 8 seconds as the simulation proceeds.

Quantitatively speaking, errors concerning the RMS e_{rel} values regarding 12 subjects are tabulated in table 7. The data displays the ability of the algorithm to converge to achieve reliable values for different subjects with different walking patterns concerning the actual force/torque values.

To provide a qualitative assessment of the algorithm performance, Fig. 60 depicts the contact wrenches as projected on the floating base $\mathbf{f}_{fb}$. They are represented by the first six

Table 7: RMS of e_{rel} concerning 12 subjects with different walking pattern

Subjects	Relative Error (mean±SD)
S1	0.0316±0.0005
S2	0.0295±0.0011
S3	0.0249±0.0006
S4	0.0245±0.0004
S5	0.0235±0.0005
S6	0.0260±0.0007
S7	0.0229±0.0008
S8	0.0211±0.0009
S9	0.0194±0.0008
S10	0.0196±0.0006
S11	0.0190±0.0006
S12	0.0292±0.0005

components of the generalized force vector $\mathbf{f}$. The force vector $\mathbf{f}_{fb}$ is given as:

$$\mathbf{f}_{fb} = [M_x\ M_y\ M_z\ F_x\ F_y\ F_z], \tag{5.102}$$

where, (M_x, M_y, M_z) and (F_x, F_y, F_z) are the moments and forces in x, y, and z directions, respectively. The estimated and actual force vectors are given below:

$$\hat{\mathbf{f}} = \mathbf{Y}(\mathbf{q}_{ref}, \dot{\mathbf{q}}_{ref}, \ddot{\mathbf{q}}_{ref})\tilde{\boldsymbol{P}}_W \hat{\tilde{\boldsymbol{\pi}}}^W \tag{5.103}$$

and

$$\mathbf{f}^{act} = \mathbf{S}\boldsymbol{\tau}^{act} + \mathbf{J}_\mathbf{c}(\mathbf{q}_{ref})^\intercal \boldsymbol{\lambda}_c^{act}, \tag{5.104}$$

respectively. The subscript *ref* and superscript *act* denote the referential and actual signals, respectively. Fig. 60 shows the estimated force vector $\hat{\mathbf{f}}_{fb} = \mathbf{S}_{fb}\hat{\mathbf{f}}$, in solid black line. The symbol $\mathbf{S}_{fb} \in \mathbb{R}^{6\times n_{dof}}$ is the first six components of the generalized force vector selection matrix. The solid red line shows the actual $\mathbf{f}_{fb}^{act}$. Fig. 60 shows a comparison of a single simulation run between the actual and estimated floating base force vector components. It depicts a good agreement between the estimated force vector and the actual response.

Moreover, for the same simulation run, the estimated joint torques vector $\hat{\boldsymbol{\tau}}$ were obtained and compared to the actual joint torques $\boldsymbol{\tau}^{act}$. The estimated joint torques are obtained as follows:

$$\hat{\boldsymbol{\lambda}}_c = (\mathbf{S}_{fb}\mathbf{J}_c(\mathbf{q}_{ref})^\intercal)^\dagger \hat{\mathbf{f}}_{fb}, \tag{5.105}$$

$$\hat{\boldsymbol{\tau}} = \mathbf{S}^\intercal(\hat{\mathbf{f}} - \mathbf{J}_c(\mathbf{q}_{ref})^\intercal \hat{\boldsymbol{\lambda}}_c), \tag{5.106}$$

where, the product of $\mathbf{S}_{fb}\mathbf{S} = \mathbf{0}$. It corresponds to the unactuated virtual joints between the floating base and the global frame. The symbol † denotes the pseudo inversion operation, and $\hat{\boldsymbol{\lambda}}_c$ represents the estimated contact force.

Fig. 61 shows the torques of the right leg for two steps. The estimated joint torques are shown in solid red while the measured torques are shown in the solid blue line. Fig. 61 demonstrates the comparable similarity between the estimated torques and the measured torques, with only small differences. Additionally, it illustrates how the estimated torque signals remain unaffected by the actual signal chattering, thanks to the inclusion of the filtered regressor. It is argued that this chattering is caused by contact transitions.

Fig. 62 depicts that the algorithm guarantees the positivity of the inertia tensor eigenvalues and the triangle constraint over a single simulation run. The same figure also shows the eigenvalues of the floating base, hip, thigh, shank, and foot inertia tensors, which are symbolized as $\boldsymbol{\lambda}_{fb}, \boldsymbol{\lambda}_h, \boldsymbol{\lambda}_{th}, \boldsymbol{\lambda}_{sh}$, and $\boldsymbol{\lambda}_f$, respectively. Fig. 62 shows the results obtained

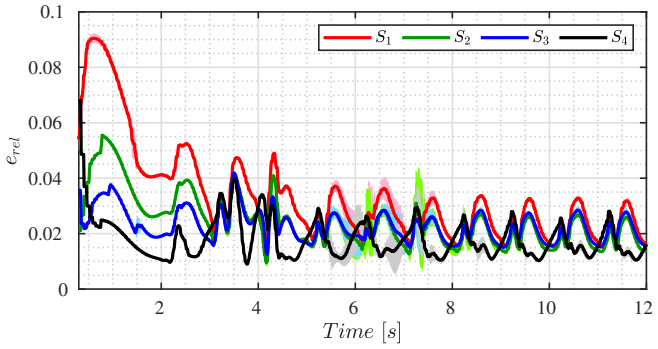

Figure 59: Relative error e_{rel} for four different subjects walking with different velocities over different simulation runs.

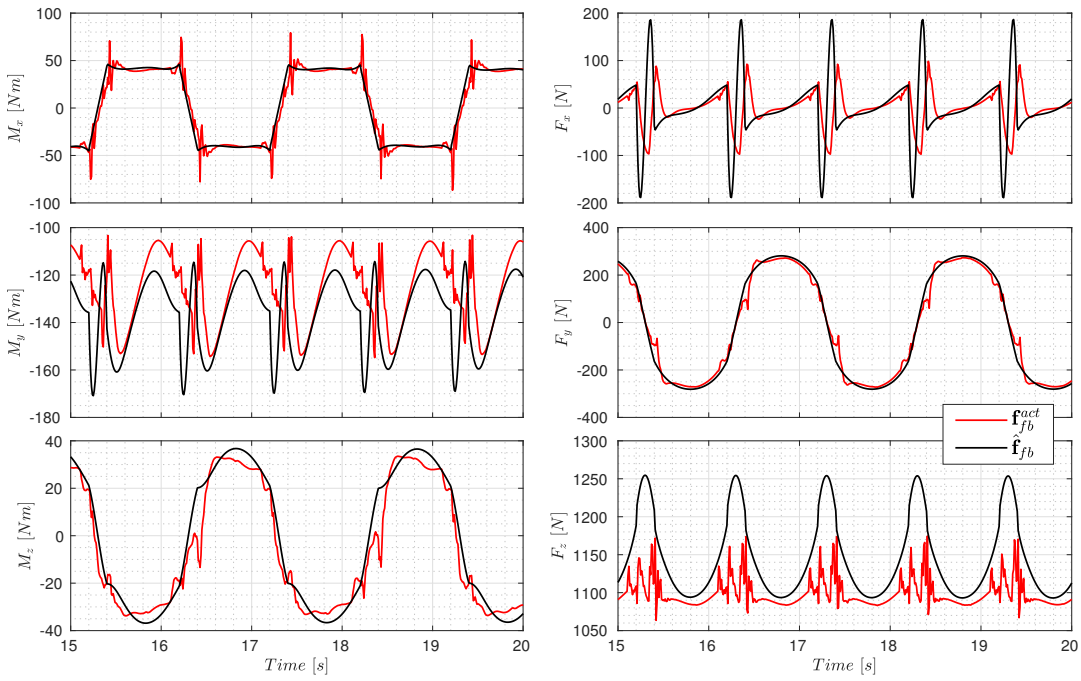

Figure 60: The estimated contact wrench components as projected on the floating base versus the actual measured value.

by the right leg only as the left leg's inertial parameters are the same. This is guaranteed via equality constraint (5.81).

The estimated masses of the robot links are shown in Fig. 63 for a single subject over six simulation runs with different walking velocities. Masses symbols are m_{fb}, m_h, m_{th}, m_{sh}, and m_f, namely, floating base, hip, thigh, shank, and foot mass, respectively. The same figure displays the mean$\pm$SD with shaded regions concerning the masses variations, while the mean value variation is denoted via solid lines. It proves the efficacy of the identifier as it shows a very small variation for the estimated masses concerning change in velocities. Moreover, the whole masses' values are settled after approximately $4s$.

5.9 Adaptive Locomotion Controller Verification

The performance of the adaptive controller is verified via a set of simulation experiments conducted using the RaiSim simulator. Free forward walking with a mean velocity of

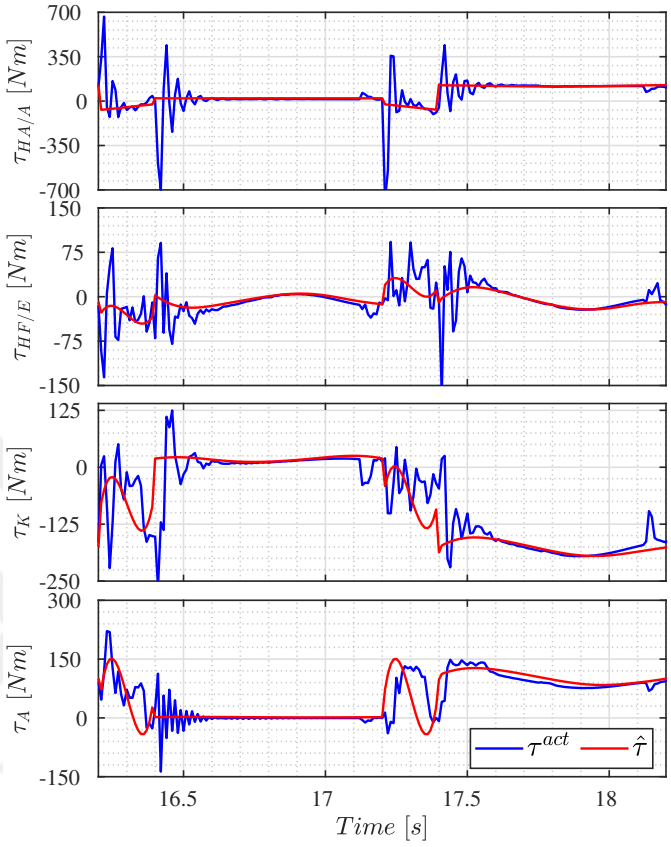

Figure 61: The estimated right leg joint torques versus the actual measured value.

250 mm/s is applied to the 40 kg robot model coupled with 12 distinct anthropomorphic user models (66 $\sim$ 102 kg and 170 $\sim$ 190 cm). The walking parameters are tuned to 0.6 and 0.2 s for single and double support periods, and the maximum foot clearance is 3 cm.

To present a comparative view and examine the individual contributions of the adaptive controller, three controllers are employed: i) Trajectory Based Control with Inverse Dynamics Algorithm (ID); see section 3.5.2, ii) Modified ZMP Impedance Feedback (MZIF); see section 4.5, and iii) the MZIF with estimated inertial parameters (MZIF+IPID); see section 5.1. The controller gains are heuristically tuned such that $\mathbf{B}zif = diag([1,\ 10])\ N.s/m$ and $\mathbf{K}zif = diag([0.1,\ 2])\ N/m$.

The ZMP variation in the x-axis and y-axis for the three controllers is shown in Fig. 64 and 65, respectively. Processing the data collected from 5 anthropomorphic subjects,

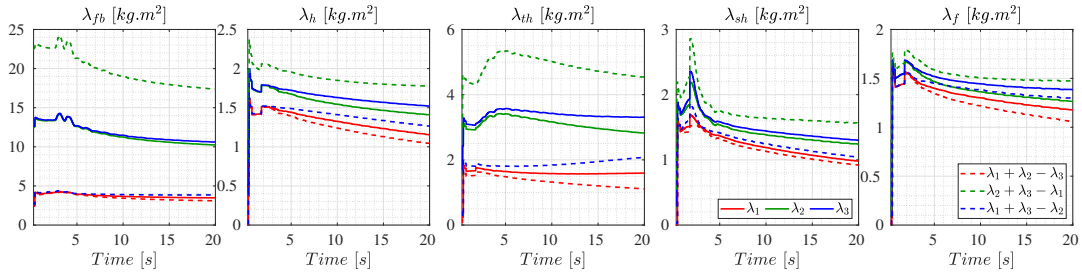

Figure 62: The variation in the inertia tensors eigenvalues and the triangle constraint over a single simulation run.

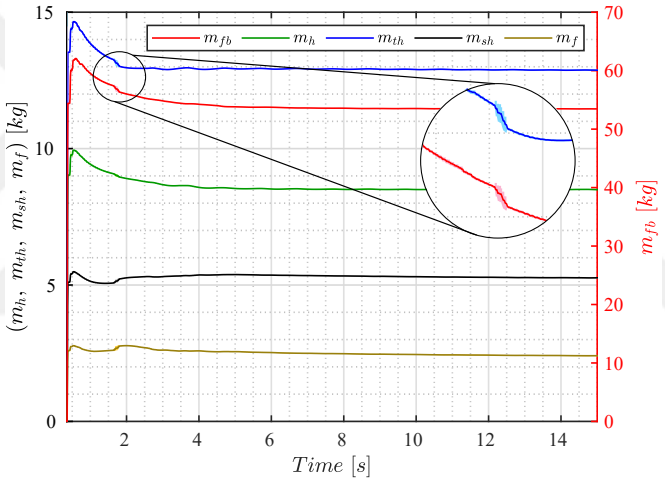

Figure 63: The variation in the estimated link masses for a single subject over six simulation runs with different velocities.

mean±SD regions concerning *xzmp* and *yzmp* variations are displayed. Mean variations are denoted via solid lines. The mean variation is computed over 10 walking steps. Among these controllers, the ZMP variations along the x-axis are comparatively well limited for the case of MZIF+IPID. In particular, peak-to-peak mean ZMP variations are observed to be farther from the support polygon boundaries generally. When it comes to ZMP variations along the y-axis, a slight increase occurs concerning its peak-to-peak value while applying MZIF+IPID. However, the variation in the y-axis ZMP is well contained within the support polygon boundaries, making the application of MZIF+IPID feasible.

Moreover, the demanded torque by the robot joints is depicted in Fig. 66. It shows the

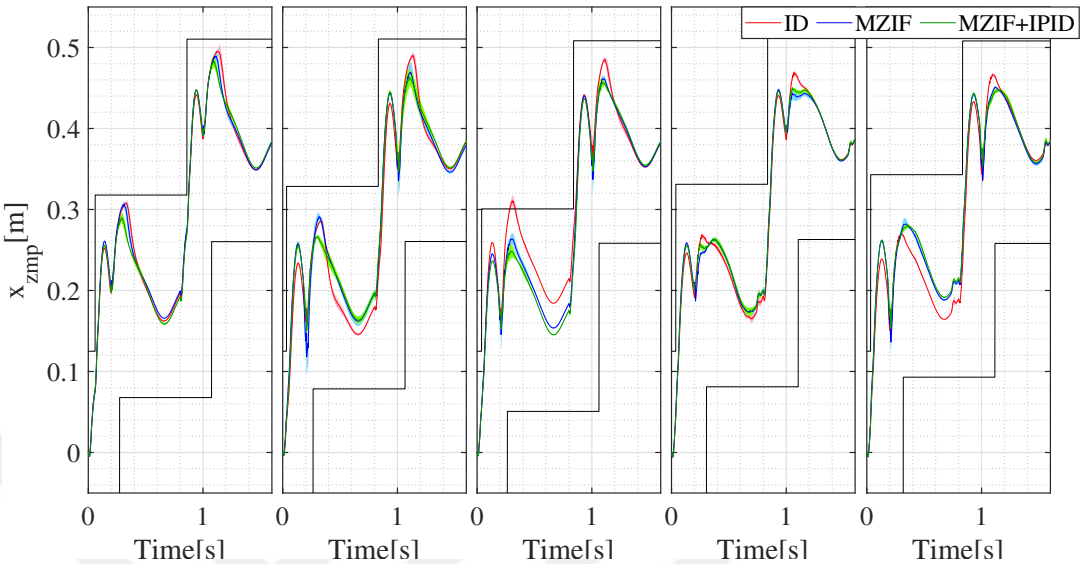

Figure 64: Average x-axis ZMP variations over 10 steps for five different subjects with respect to the global frame.

root mean square value of the torque for each subject over the whole simulation run. A consistent reduction in demanded torque is shown for MZIF+IPID. Fig. 67 illustrates the variation between the three controllers based on the averaged values of the three metrics, namely, x-axis ZMP peak-to-peak x_{zmp}^{pp}, y-axis ZMP peak-to-peak y_{zmp}^{pp}, and torque root mean square τ^{rms}. MZIF+IPID outperforms the other controllers concerning x_{zmp}^{pp} and τ^{rms}. The proposed controller MZIF+IPID is still under study to improve its impact on y_{zmp}^{pp}.

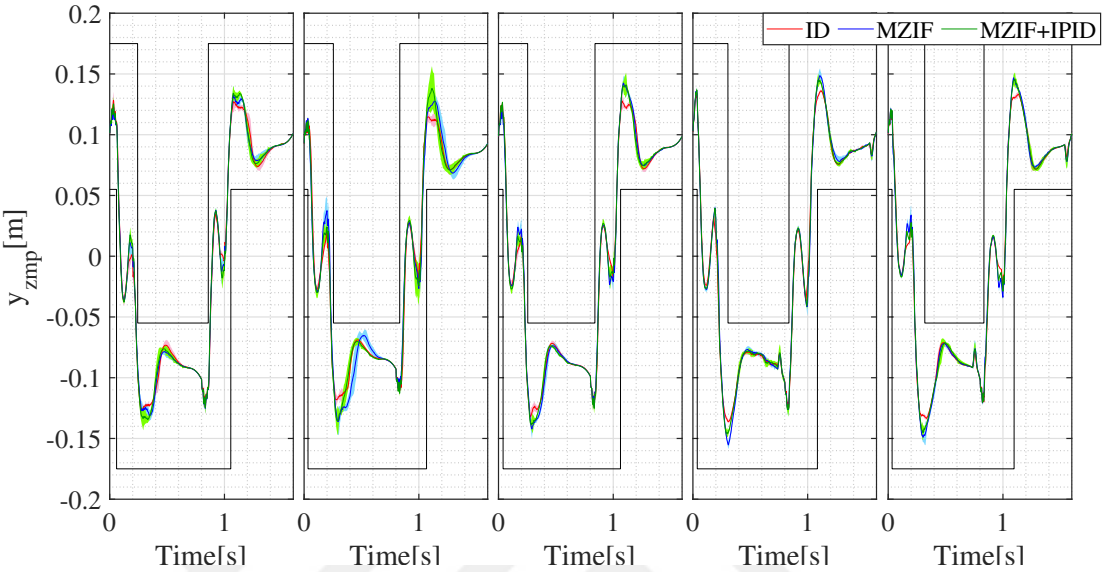

Figure 65: Average y-axis ZMP variations over 10 steps for five different subjects with respect to the global frame.

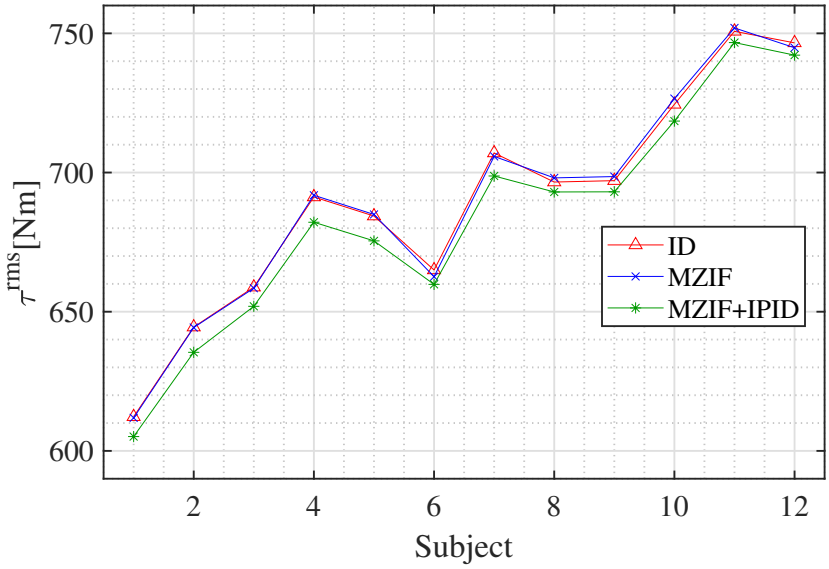

Figure 66: The robot joints demanded torque root mean square for each subject.

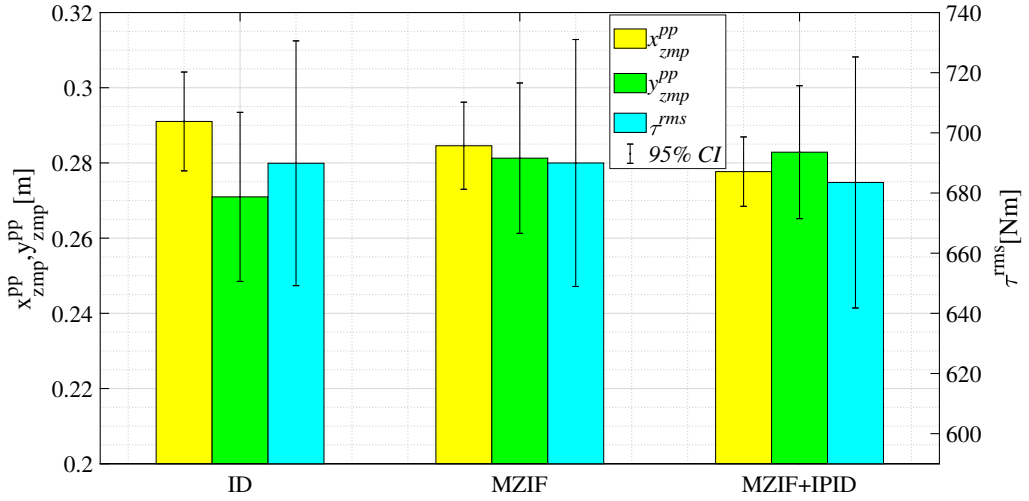

Figure 67: Average values of x_{zmp}^{pp}, y_{zmp}^{pp}, and τ^{rms} for 12 anthropomorphic subjects.

CHAPTER VI

CONCLUSION

This thesis presented a framework to address three-dimensional (3D) dynamic walking for a bipedal exoskeleton with underactuated legs. To achieve this goal, the framework is constructed via a Cartesian trajectory generator and an optimized inverse kinematics algorithm that can cope with underactuation. Differing from the case of classical bipeds, where each robot leg has at least six active joints, it is quite difficult to satisfy several constraints to ensure dynamic walking. To feasibly attain task velocities with underactuated legs, the inverse kinematics algorithm makes use of a task prioritization method via the exploitation of null space. In doing so, the tasks with lower priority, e.g., swing foot orientation, are attained as much as possible without disrupting the higher priority tasks, such as CoM trajectory. Meanwhile, the trajectory generator utilizes the ZMP concept analytically and ensures acceleration continuity throughout the whole walking period, regardless of contact and phase changes. As a result, a feasible and dynamically balanced 3-D walking motion was obtained, in which no oblique foot landing or exaggerated torso orientation variations were observed, despite the underactuated nature of the robot legs.

Moreover, a set of locomotion controller methodologies were presented concerning the robot-aided walking task for an underactuated bipedal exoskeleton with an arbitrary human user. The proposed optimization algorithm allows the synthesis of other feedback controllers, e.g., admittance control or centroidal momentum control.

The framework was thoroughly tested with 3 distinct feedback controllers in a realistic simulation environment in which a 40 kg underactuated exoskeleton model and 12 anthropomorphic ($66 \sim 102$ kg) subjects were utilized. As a result, the human-robot combined

systems exhibited dynamically balanced walking behavior robustly, despite parameter uncertainties and applied perturbations. This result adequately confirmed the validity of the proposed optimization algorithm in handling the underactuation problem. Among the feedback controllers, the proposed ZMP impedance controller outperformed other controllers with statistical significance.

As the proposed locomotion controllers are based on a robot dynamic model, a real-time applicable inertial parameter identification algorithm was presented. The algorithm is used to improve the adaptability of the locomotion controllers concerning inertial parameter variations. The identification procedure guarantees complete physical consistency. Moreover, it is enforced by the filtered regressor idea [55] to improve its immunity to the contact transitions instants and signals noise. Also, an algorithm for recursively updating the identifiable inertial parameter projection matrix to enable a recursive inversion process is developed. To guarantee complete physical consistency, a semidefinite programming algorithm based on the interior point method approach is modified to enable the recursive inversion and suit the online identification procedure.

A human bipedal exoskeleton coupled model [52] is used as a case study and simulated via RaiSim dynamic simulator to verify the identification algorithm's efficacy. For thorough results, 12 distinct anthropomorphic subjects were used, each of which walks at six different velocities for six simulation runs. The identification algorithm showed promising results as the relative error was relatively low, and the computation time $(0.1 \sim 0.25\ s)$ is real-time applicable. Moreover, the complete physical consistency of the resultant inertial parameters is verified.

To confirm the proposed algorithms experimentally, Co-Ex bipedal exoskeleton is constructed. Co-Ex is a lower-body exoskeleton with four active joints per leg. It is equipped with eight series elastic actuators for actuation purposes, sixteen force-sensitive resistors for measuring contact forces, and sixteen absolute encoders for angular displacement and SEA spring deflection sensing. Moreover, a set of Advantech PCI cards, Raspberry Pi

circuits, and an Ethernet router were used for interfacing purposes between a desktop PC and the robot sensors and actuators. The PC is equipped with Ubuntu 18.04 and Xenomai 3.1 for real-time implementation. A C language with a GNU scientific library was used to control the robot's motion and to solve the required linear algebra operations. The PC is used to provide the joint and Cartesian level controllers. Throughout the conducted experiments, the real-time applicability of the proposed algorithms was proven. Moreover, both the robot's behavior through position and the motor's torque tracking proved the applicability and feasibility of the control algorithms.

APPENDIX A

CO-EX KINEMATIC AND DYNAMIC MODELING

The appendix includes further technical details regarding the modified Newton-Euler algorithm that was used to compute the floating base dynamics of the system. Furthermore, the recursive algorithm that was used to compute Jacobian matrices is provided.

A.1 Forward Kinematics and Jacobian Computation

The bipedal model can be considered a branched structure. The floating coordinates were represented by considering 6 virtual DoFs to represent the base frame F_0 position and orientation concerning the inertial frame F_G; see Fig. 68. It is desired to control the following coordinates: i) center position and orientation of the right foot, ii) center position and orientation of the left foot, iii) position of the CoM (Center of Mass), and torso orientation. In another word, 3 end-effectors are existing.

To compute the related end effector Jacobian matrices in a computationally inexpensive way, a recursive Newton-Euler (NE) approach with forward propagation is employed. It has a complexity of $O(n_r+6)$. In this algorithm, angular and translational velocity vectors concerning the i^{th} frame are respectively yielded as below.

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + {}^0\mathbf{R}_i \mathbf{z}_g \dot{\boldsymbol{q}}_i \tag{A.1}$$

$$\boldsymbol{v}_i = \boldsymbol{v}_{i-1} + \boldsymbol{\omega}_i \times \mathbf{r}_i \tag{A.2}$$

In (A.1)-(A.2), ${}^0R_i$ is the orientation matrix of the i^{th} link with respect to frame F_0, $\mathbf{z}_g = [0\ 0\ 1]^\intercal$ is a generalized z-axis selection vector, r_i is the displacement vector between the i^{th} and $(i-1)^{th}$ links, and $\dot{q}_i$ represents the angular velocity of the i^{th} joint. Using

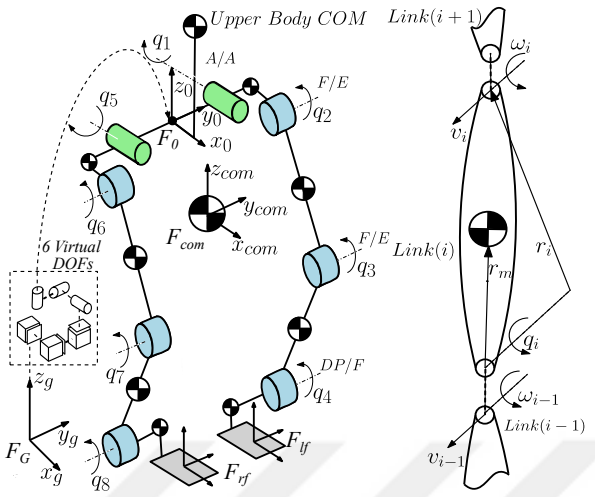

Figure 68: Left: Joint configuration of the biped model. Right: A general manipulator structure that describes the parameters used in kinematics and dynamics algorithms.

(A.1)-(A.2), angular and translational portions of the corresponding Jacobian matrices can be obtained as below:

$$\mathbf{J}^i_{\omega} = [\mathbf{J}^{i-1}_{\omega} \;\; {}^0\mathbf{R}_i\mathbf{z}_g] \tag{A.3}$$

$$\mathbf{J}^i_{v} = [\mathbf{J}^{i-1}_{v} \;\; \mathbf{0}_{3\times 1}] - \overline{S}(\mathbf{r}_i)\mathbf{J}^i_{\omega} \tag{A.4}$$

In (A.3)-(A.4), $\mathbf{0}_{3\times 1}$ is a 3×1 zero matrix, $\overline{S}(\mathbf{r}_i)$ is the skew matrix of r_i which satisfies the cross product operation $\mathbf{r}_i\times$. The time derivatives of the Jacobian matrices are recursively computed as follows:

$$\dot{\mathbf{J}}^i_{\omega} = [\dot{\mathbf{J}}^{i-1}_{\omega} \;\; \overline{S}(\boldsymbol{\omega}_i){}^0\mathbf{R}_i\mathbf{z}_g] \tag{A.5}$$

$$\dot{\mathbf{J}}^i_{v} = [\dot{\mathbf{J}}^{i-1}_{v} \;\; \mathbf{0}_{3\times 1}] - \overline{S}(\boldsymbol{\omega}_i)\overline{S}(\mathbf{r}_i)\mathbf{J}^i_{\omega} - \overline{S}(\mathbf{r}_i)\dot{\mathbf{J}}^i_{\omega}, \tag{A.6}$$

where $\overline{S}(\omega_i)$ is the skew matrix of ω_i which satisfies the cross product $\omega_i\times$. The overall CoM Jacobian can be computed as in the following:

$$\mathbf{J}_{com} = \sum_{i=1}^{n} \overline{w}_i\mathbf{J}^i_{com} \tag{A.7}$$

In (A.7), m_i is the i^{th} link mass, m_r is the total robot mass, $\overline{w}_i$ is the ratio of $\overline{w}_i = m_i/m_r$, n_b is the number of links (the actuated joints) excluding the floating base. $\mathbf{J}^i_{com}$ stands for

the CoM Jacobian concerning the i^{th} link and it can be computed recursively:

$$\mathbf{J}^i_{com} = [\mathbf{J}^{i-1}_v \ \mathbf{0}_{3\times 1}] - \overline{S}(\mathbf{r}^i_m)\mathbf{J}^i_\omega. \tag{A.8}$$

Its time derivative is obtained as below:

$$\dot{\mathbf{J}}^i_{com} = [\dot{\mathbf{J}}^{i-1}_v \ \mathbf{0}_{3\times 1}] - \overline{S}(\boldsymbol{\omega}_i)\overline{S}(\mathbf{r}^i_m)\mathbf{J}^i_\omega - \overline{S}(\mathbf{r}^i_m)\dot{\mathbf{J}}^i_\omega. \tag{A.9}$$

In (A.8) and (A.9), $\mathbf{r}^i_m$ is the local coordinates of the i^{th} link CoM. Table 8 provides the algorithm to compute the Jacobian matrix and its time derivative. The subscript B indicates the floating base. The number of joints per leg is n_l; $n_l = 4$ for the robot in question. The subscript li stands for the leg index (left: $li = 1$, right $li = 2$). The subscript b indicates the floating base virtual joints. $\mathbf{J}_{lf}, \mathbf{J}_{rf}$, and $\mathbf{J}_{torso}$ are the left, right foot, and torso Jacobian matrices, respectively.

A.2 Inverse Dynamics: The Modified-NE (MNE) Algorithm

The section presents the derivation of the Modified-NE (MNE) algorithm. The symbols used in this section are defined in Table 9. The MNE algorithm was developed to compute Coriolis centrifugal matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ recursively. Generally speaking, the conventional NE algorithm can be used to solve Inverse Dynamics (ID) function such that

$$\boldsymbol{\tau} = ID(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}, \boldsymbol{\lambda}_e). \tag{A.10}$$

The sum of Coriolis centrifugal and gravitational terms can be obtained from (A.10) as below:

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = ID(\ddot{\mathbf{q}} = \mathbf{0}, \dot{\mathbf{q}}, \mathbf{q}, \boldsymbol{\lambda}_e = \mathbf{0}). \tag{A.11}$$

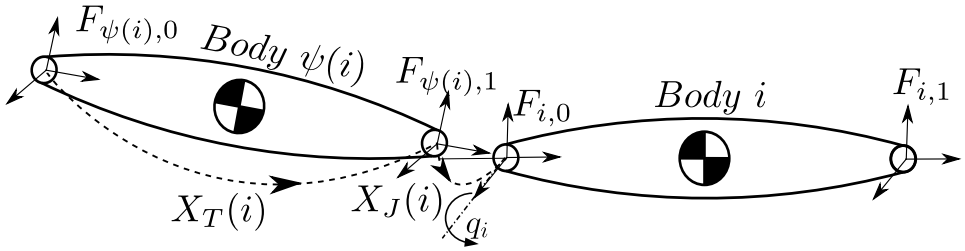

Figure 69: Coordinate frames and transformations concerning the i^{th} joint.

The eq. (A.11) can be further used to obtain the gravitational matrix by assigning joint velocities to zero. Thus $\mathbf{G}(\mathbf{q}) = ID(\ddot{\mathbf{q}} = \mathbf{0}, \dot{\mathbf{q}} = \mathbf{0}, \mathbf{q}, \boldsymbol{\lambda}_e = \mathbf{0})$. Subtracting $\mathbf{G}(\mathbf{q})$ from (A.11) yields $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$. As it is required to decouple $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ from the generalized joint velocity vector $\dot{\mathbf{q}}$, a set of modifications should be applied to the conventional floating base NE algorithm.

As provided in Table 10, the conventional NE algorithm consists of three main loops. The first loop is a forward recursion loop covered between the 3^{rd} and 10^{th} lines. It is utilized to compute frame $F_{i,0}$ velocity, and acceleration for each link. It additionally provides the forces that are required to produce such accelerations. The second loop is a backward recursion loop covered between the 12^{th} and 19^{th} lines. It is utilized to determine the forces transmitted across the joints. It also yields the inertia matrices that are projected from children's links to parent links. The last loop is a forward recursion loop covered between the 21^{st} and 26^{th} lines. It is used to obtain the floating base acceleration as projected onto each link. In line 27, the ID solution is completed by computing the torque $\boldsymbol{\tau}_i$ that is associated with the joint i. See [1] for further details.

The NE algorithm displayed in Table 10 was modified to obtain $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{G}(\mathbf{q})$ separately, and it is called the MNE (Modified NE) algorithm. It is provided in Table 11. Line 2 was modified in the NE algorithm by adding two matrices: $\mathbf{J}_0$ and $\dot{\mathbf{J}}_0$. Initially, $\mathbf{J}_0$ is defined as an identity matrix ($\mathbf{1}_{6\times6}$), and $\dot{\mathbf{J}}_0$ is defined as a zero matrix ($\mathbf{0}_{6\times6}$), respectively. Following this step, the linear mapping between the Coriolis & centrifugal force and velocity vector is obtained as $\boldsymbol{\tau}_{cr} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})[\boldsymbol{\nu}_{fb}^{\intercal} \;\; \dot{\mathbf{q}}_r^{\intercal}]^{\intercal}$. The floating base velocity is denoted by $\boldsymbol{\nu}_{fb} = \mathbf{J}_{fb}\dot{\mathbf{q}}_{fb}$, where $\dot{\mathbf{q}}_{fb}$ is 6 DOFs floating base joint velocities. The actuated joint velocity vector is $\dot{\mathbf{q}}_r$. The joint accelerations $\ddot{q}_i$ in line 8 and the external force $\boldsymbol{\lambda}_{e,i}$ in line 9 in NE are assigned as zero vectors in MNE as stated by (A.11).

Lines 7 and 8 from the NE algorithm are used to extract Jacobian, its time derivative, and gravitational acceleration for each link in the MNE algorithm; see lines 8 and 9. The extracted matrices are then combined in line 10 to show the linear mapping matrix $\mathbf{M}_{1,i}$

defined between each link's individual force (see line 9 in the NE algorithm) and joint velocities. Lines 20 and 21 in the MNE algorithm are used to transmit $\mathbf{M}_{1,i}$ and gravitational forces across the joints respectively. Line 23 in MNE is used to stack the mapping matrix $\mathbf{M}_{2,i}$ and gravitational force $\mathbf{f}_{g,i}$ for each joint into joint space matrices $\mathbf{C_r}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{G_r}(\mathbf{q})$. Lines 25 and 26 are used to augment the joint space and the floating base matrices into Coriolis & centrifugal, and gravitational matrices, namely, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and $\mathbf{G}(\mathbf{q})$.

Table 8: Jacobian computation algorithm. The superscript square brackets in lines 31 and 32 indicate the column indexes, e.g. [a:b] indicates the matrix columns from a to b.

Jacobian Computation Algorithm	
1	Initially set $\mathbf{J}^0_{\omega,b}$, $\mathbf{J}^0_{v,b}$, $\dot{\mathbf{J}}^0_{\omega,b}$, and $\dot{\mathbf{J}}^0_{v,b}$ as empty matrices
2	for $i = 1$ to 6
3	if $i \leq 3$
4	$\mathbf{J}^i_{\omega,b} = [\mathbf{J}^{i-1}_{\omega,b} \;\; \mathbf{0}_{3\times1}]$
5	$\mathbf{J}^i_{v,b} = [\mathbf{J}^{i-1}_{v,b} \;\; {}^0\mathbf{R}_i\mathbf{z}_g] - \overline{S}(\mathbf{r}_i)\mathbf{J}^i_{\omega,b}$
6	$\dot{\mathbf{J}}^i_{\omega,b} = [\dot{\mathbf{J}}^{i-1}_{\omega,b} \;\; \mathbf{0}_{3\times1}]$
7	$\dot{\mathbf{J}}^i_{v,b} = [\dot{\mathbf{J}}^{i-1}_{v,b} \;\; \overline{S}(\boldsymbol{\omega}_i){}^0\mathbf{R}_i z_g] - \overline{S}(\boldsymbol{\omega}_i)s(\mathbf{r}_i)\mathbf{J}^i_{\omega,b} - \overline{S}(\mathbf{r}_i)\dot{\mathbf{J}}^i_{\omega,b}$
8	else
9	$\mathbf{J}^i_{\omega,b} = [\mathbf{J}^{i-1}_{\omega,b} \;\; {}^0\mathbf{R}_i\mathbf{z}_g]$
10	$\mathbf{J}^i_{v,b} = [\mathbf{J}^{i-1}_{v,b} \;\; \mathbf{0}_{3\times1}] - \overline{S}(\mathbf{r}_i)J^i_{\omega,b}$
11	$\dot{\mathbf{J}}^i_{\omega,b} = [\mathbf{J}^{i-1}_{\omega,b} \;\; \overline{S}(\boldsymbol{\omega}_i){}^0\mathbf{R}_i z_g]$
12	$\dot{\mathbf{J}}^i_{v,b} = [\dot{\mathbf{J}}^{i-1}_{v,b} \;\; \mathbf{0}_{3\times1}] - \overline{S}(\boldsymbol{\omega}_i)\overline{S}(\mathbf{r}_i)\mathbf{J}^i_{\omega,b} - \overline{S}(\mathbf{r}_i)\dot{\mathbf{J}}^i_{\omega,b}$
13	end
14	end
15	$\mathbf{J}_{v,B} = \begin{bmatrix}\mathbf{0}_{3\times2n_l} \; \mathbf{J}_{v,b}\end{bmatrix}$, $\mathbf{J}_{\omega,B} = \begin{bmatrix}\mathbf{0}_{3\times2n_l} \; \mathbf{J}_{\omega,b}\end{bmatrix}$
16	$\dot{\mathbf{J}}_{v,B} = \begin{bmatrix}\mathbf{0}_{3\times2n_l} \; \dot{\mathbf{J}}_{v,b}\end{bmatrix}$, $\dot{\mathbf{J}}_{\omega,B} = \begin{bmatrix}\mathbf{0}_{3\times2n_l} \; \dot{\mathbf{J}}_{\omega,b}\end{bmatrix}$
17	for $li = 1$ to 2
18	$\mathbf{J}^0_{\omega,li} = \mathbf{J}_{\omega,b}$, $\mathbf{J}^0_{v,li} = \mathbf{J}_{v,b}$, and $\mathbf{J}_{com,li} = \mathbf{0}_{3\times1}$
19	$\dot{\mathbf{J}}^0_{\omega,li} = \dot{\mathbf{J}}_{\omega,b}$, $\dot{\mathbf{J}}^0_{v,li} = \dot{\mathbf{J}}_{v,b}$, and $\dot{\mathbf{J}}_{com,li} = \mathbf{0}_{3\times1}$
20	for $i = 1$ to n_l
21	$\mathbf{J}^i_{\omega,li} = [\mathbf{J}^{i-1}_{\omega,li} \;\; {}^0\mathbf{R}_{i,li}\mathbf{z}_g]$
22	$\dot{\mathbf{J}}^i_{\omega,li} = [\dot{\mathbf{J}}^{i-1}_{\omega,li} \;\; \overline{S}(\boldsymbol{\omega}_i){}^0\mathbf{R}_{i,li}\mathbf{z}_g]$
23	$\mathbf{J}^i_{v,li} = [\mathbf{J}^{i-1}_{v,li} \;\; \mathbf{0}_{3\times1}] - \overline{S}(\mathbf{r}_i)\mathbf{J}^i_{\omega,li}$
24	$\dot{\mathbf{J}}^i_{v,li} = [\dot{\mathbf{J}}^{i-1}_{v,li} \;\; \mathbf{0}_{3\times1}] - \overline{S}(\boldsymbol{\omega}_i)\overline{S}(\mathbf{r}_i)\mathbf{J}^i_{\omega,li} - \overline{S}(\mathbf{r}_i)\dot{\mathbf{J}}^i_{\omega,li}$
25	$\mathbf{J}^i_{com,li} = [\mathbf{J}^{i-1}_{v,li} \;\; \mathbf{0}_{3\times1}] - \overline{S}(\mathbf{r}^i_m)\mathbf{J}^i_{\omega,li}$
26	$\dot{\mathbf{J}}^i_{com,li} = [\dot{\mathbf{J}}^{i-1}_{v,li} \;\; \mathbf{0}_{3\times1}] - \overline{S}(\boldsymbol{\omega}_i)\overline{S}(\mathbf{r}^i_m)\mathbf{J}^i_{\omega,li} - \overline{S}(\mathbf{r}^i_m)\dot{\mathbf{J}}^i_{\omega,li}$
27	$\mathbf{J}_{com,li} = m_i\mathbf{J}^i_{com,li} + \mathbf{J}_{com,li}$
28	$\dot{\mathbf{J}}_{com,li} = m_i\dot{\mathbf{J}}^i_{com,li} + \dot{\mathbf{J}}_{com,li}$
29	end
30	end
31	$\mathbf{J}_{com,lf} = [\mathbf{J}^{[7:n_l]}_{com,1} \;\; \mathbf{0}_{3\times n_l} \;\; \mathbf{J}^{[1:6]}_{com,1}]$, $\dot{\mathbf{J}}_{com,lf} = [\dot{\mathbf{J}}^{[7:n_l]}_{com,1} \;\; \mathbf{0}_{3\times n_l} \;\; \dot{\mathbf{J}}^{[1:6]}_{com,1}]$
32	$\mathbf{J}_{com,rf} = [\mathbf{0}_{3\times n_l} \;\; \mathbf{J}^{[7:n_l]}_{com,2} \;\; \mathbf{J}^{[1:6]}_{com,2}]$, $\dot{\mathbf{J}}_{com,rf} = [\mathbf{0}_{3\times n_l} \;\; \dot{\mathbf{J}}^{[7:n_l]}_{com,2} \;\; \dot{\mathbf{J}}^{[1:6]}_{com,2}]$
33	$\mathbf{J}_{com} = (m_B\mathbf{J}_{v,B} + \mathbf{J}_{com,lf} + \mathbf{J}_{com,rf})/m_r$, $\dot{\mathbf{J}}_{com} = (m_B\dot{\mathbf{J}}_{v,B} + \dot{\mathbf{J}}_{com,1} + \dot{\mathbf{J}}_{com,2})/m_r$
34	$\mathbf{J}_{lf} = \begin{bmatrix}\mathbf{J}_{v,1}\\ \mathbf{J}_{\omega,1}\end{bmatrix}$, $\mathbf{J}_{rf} = \begin{bmatrix}\mathbf{J}_{v,2}\\ \mathbf{J}_{\omega,2}\end{bmatrix}$, $\mathbf{J}_{torso} = \mathbf{J}_{\omega,B}$
35	$\dot{\mathbf{J}}_{lf} = \begin{bmatrix}\dot{\mathbf{J}}_{v,1}\\ \dot{\mathbf{J}}_{\omega,1}\end{bmatrix}$, $\dot{\mathbf{J}}_{rf} = \begin{bmatrix}\dot{\mathbf{J}}_{v,2}\\ \dot{\mathbf{J}}_{\omega,2}\end{bmatrix}$, $\dot{\mathbf{J}}_{torso} = \dot{\mathbf{J}}_{\omega,B}$

Table 9: Nomenclature for NE and MNE algorithms [1].

Symbol	Definition
$\boldsymbol{\tau}$	Joint torque vector
$(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$	Joint position, velocity, and acceleration vectors
$\boldsymbol{\lambda}_{e,i}$	Spatial force vector specifying the external force acting on body i, expressed in frame $F_{i,0}$.
$\mathbf{g}_s$	Spatial vector represents the gravitational acceleration
i	Actuated joint/body index. The joint i is the joint that connects body i to its parent; see Fig. 69
$\psi(i)$	The body number of the parent of body i
N	An integer specifying the number of bodies in the mechanism
$\mathbf{X}_J(i)$	The coordinate transform from frame $F_{\psi(i),1}$ to frame $F_{i,0}$ as shown in Fig. 69
$\mathbf{X}_T(i)$	The Plücker coordinate transform from frame $F_{\psi(i),0}$ to frame $F_{\psi(i),1}$.
${}^{i}\mathbf{X}_{\psi(i)}$	The Plücker coordinate transform from frame $F_{\psi(i),0}$ to frame $F_{i,0}$ for motion vectors
${}^{i}\mathbf{X}^{*}_{\psi(i)}$	The Plücker coordinate transform from frame $F_{\psi(i),0}$ to frame $F_{i,0}$ for force vectors
$\bar{\mathbf{s}}_i$	Joint axis vector
$\bar{h}_i$	Joint type (Prismatic or Revolute)
$jcalc(\bullet)$	A function used to compute joint i transformation matrix $\mathbf{X}_J(i)$ and vector s_i
$(\boldsymbol{\nu}_i, \mathbf{a}_i)$	Body i frame $F_{i,0}$ velocity and acceleration respectively
$(q_i, \dot{q}_i, \ddot{q}_i)$	Joint i position, velocity, and acceleration respectively
$\mathbf{f}_i$	The force transmitted across joint i
$\mathbf{I}_i$	Spatial inertia matrix of body i projected on $F_{i,0}$
$\times$	Cross product in motion space
$\times^*$	Cross product in force space
$\mathbf{f}_{fb}$	The force exerted by floating base on its children
$\mathbf{I}_{fb}$	Floating base Spatial inertia matrix projected on $F_{0,0}$; see Fig. 68
$(\boldsymbol{\nu}_{fb}, \mathbf{a}_{fb})$	Floating base spatial velocity and acceleration expressed in frame $F_{0,0}$
τ_i	Corresponding torque at the joint i
$\mathbf{J}_i$	Jacobian matrix for body i frame $F_{i,0}$
$\mathbf{J}_{com}$	Jacobian matrix of the aggregated CoM at frame F_G

Table 10: The conventional Newton Euler algorithm that can be used to compute inverse dynamics of a floating base robot [1].

NE Algorithm	
1	$\boldsymbol{\nu}_0 = \boldsymbol{\nu}_{fb}$
2	$\mathbf{a}_0 = -\mathbf{g}_s$
3	$for\ i = 1\ to\ N\ do$
4	$[\mathbf{X}_J, \bar{\mathbf{s}}_i] = jcalc(\bar{h}_i, q_i)$
5	${}^{i}\mathbf{X}_{\psi(i)} = \mathbf{X}_J \mathbf{X}_T(i)$
6	${}^{\psi(i)}\mathbf{X}_i^* = {}^{i}\mathbf{X}_{\psi(i)}^T$
7	$\boldsymbol{\nu}_i = {}^{i}\mathbf{X}_{\psi(i)} \boldsymbol{\nu}_{\psi(i)} + \bar{\mathbf{s}}_i \dot{q}_i$
8	$\mathbf{a}_i = {}^{i}\mathbf{X}_{\psi(i)} \mathbf{a}_{\psi(i)} + \bar{\mathbf{s}}_i \ddot{q}_i + \boldsymbol{\nu}_i \times \bar{\mathbf{s}}_i \dot{q}_i$
9	$\mathbf{f}_i = \mathbf{I}_i \mathbf{a}_i + \boldsymbol{\nu}_i \times^* \mathbf{I}_i \boldsymbol{\nu}_i - \boldsymbol{\lambda}_{e,i}$
10	end
11	$\mathbf{f}_{fb} = \mathbf{I}_{fb} \mathbf{a}_{fb} + \boldsymbol{\nu}_{fb} \times^* \mathbf{I}_{fb} \boldsymbol{\nu}_{fb}$
12	$for\ i = N\ to\ 1\ do$
13	$If\ \psi(i) = 0\ then$
14	$\mathbf{I}_{fb} = \mathbf{I}_{fb} + {}^{\psi(i)}\mathbf{X}_i^T \mathbf{I}_i \mathbf{X}_i^{\psi(i)}$
15	$\mathbf{f}_{fb} = \mathbf{f}_{fb} + {}^{\psi(i)}\mathbf{X}_i^* \mathbf{f}_i$
16	$else$
17	$\mathbf{I}_{\psi(i)} = \mathbf{I}_{\psi(i)} + {}^{\psi(i)}\mathbf{X}_i^T \mathbf{I}_i \mathbf{X}_i^{\psi(i)}$
18	$\mathbf{f}_{\psi(i)} = \mathbf{f}_{\psi(i)} + {}^{\psi(i)}\mathbf{X}_i^* \mathbf{f}_i$
19	end
20	$\mathbf{a}_0 = \mathbf{a}_{fb}$
21	$for\ i = 1\ to\ N\ do$
22	$If\ \psi(i) = 0\ then$
23	$\mathbf{a}_i = {}^{i}\mathbf{X}_{\psi(i)} \mathbf{a}_{fb}$
24	$else$
25	$\mathbf{a}_i = {}^{i}\mathbf{X}_{\psi(i)} \mathbf{a}_{\psi(i)}$
26	end
27	$\boldsymbol{\tau}_i = \bar{\mathbf{s}}_i^T (\mathbf{I}_i \mathbf{a}_i + \mathbf{f}_i)$
28	end

Table 11: The MNE algorithm that is employed to compute Coriolis & centrifugal matrix $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ and the gravitation force vector $\mathbf{G}(\mathbf{q})$.

Modified-NE (MNE) Algorithm	
1	$\boldsymbol{\nu}_0 = \boldsymbol{\nu}_{fb}$
2	$\mathbf{g}_0 = -\mathbf{g}_s,\ \mathbf{J}_0 = \mathbf{1}_{6\times 6},\ \dot{\mathbf{J}}_0 = \mathbf{0}_{6\times 6}$
3	$for\ i = 1\ to\ N\ do$
4	$[\mathbf{X}_J, \bar{\mathbf{s}}_i] = jcalc(\bar{h}_i, q_i)$
5	${}^{i}\mathbf{X}_{\psi(i)} = \mathbf{X}_J \mathbf{X}_T(i)$
6	${}^{\psi(i)}\mathbf{X}_i^* = {}^{i}\mathbf{X}_{\psi(i)}^T$
7	$\boldsymbol{\nu}_i = {}^{i}\mathbf{X}_{\psi(i)} \boldsymbol{\nu}_{\psi(i)} + \bar{\mathbf{s}}_i \dot{q}_i$
8	$\mathbf{J}_i = [{}^{i}\mathbf{X}_{\psi(i)} \mathbf{J}_{\psi(i)} \ \ \bar{\mathbf{s}}_i]$
9	$\dot{\mathbf{J}}_i = [{}^{i}\mathbf{X}_{\psi(i)} \dot{\mathbf{J}}_{\psi}(i) \ \ \boldsymbol{\nu}_i \times \bar{\mathbf{s}}_i],\ \mathbf{g}_i = {}^{i}\mathbf{X}_{\psi(i)} \mathbf{g}_{\psi(i)}$
10	$\mathbf{M}_{1,i} = (\boldsymbol{\nu}_i \times^* \mathbf{I}_i \mathbf{J}_i) + [\mathbf{0}_{6\times 6} \ \ \mathbf{I}_i \dot{\mathbf{J}}_i],\ \mathbf{f}_{g,i} = \mathbf{I}_i \mathbf{g}_i$
11	end
12	$\mathbf{f}_{fb,g} = \mathbf{I}_{fb} \mathbf{g}_o$
13	$for\ i = N\ to\ 1\ do$
14	$If\ \psi(i) = 0\ then$
15	$\mathbf{I}_{fb} = \mathbf{I}_{fb} + {}^{\psi(i)}\mathbf{X}_i^T \mathbf{I}_i {}^{\psi(i)}\mathbf{X}_i$
16	$\mathbf{k}_{fb} = [\boldsymbol{\nu}_{fb} \times^* \mathbf{I}_{fb} \mathbf{J}_{\psi(i)} \ \mathbf{0}_{6\times N}] + {}^{\psi(i)}\mathbf{X}_i^* \mathbf{M}_{2,i}$
17	$\mathbf{f}_{fb,g} = \mathbf{f}_{fb,g} + {}^{\psi(i)}\mathbf{X}^* \mathbf{f}_{g,i}$
18	$else$
19	$\mathbf{I}_{\psi(i)} = \mathbf{I}_{\psi(i)} + {}^{\psi(i)}\mathbf{X}_i^T \mathbf{I}_i {}^{\psi(i)}\mathbf{X}_i$
20	$\mathbf{M}_{2,\psi(i)} = \mathbf{M}_{1,\psi(i)} + {}^{\psi(i)}\mathbf{X}_i^* \mathbf{M}_{2,i}$
21	$\mathbf{f}_{g,\psi(i)} = \mathbf{f}_{g,\psi(i)} + {}^{\psi(i)}\mathbf{X}_i^* \mathbf{f}_{g,i}$
22	end
23	$\mathbf{C}_{\mathbf{r},\mathbf{i}}(\mathbf{q},\dot{\mathbf{q}}) = \bar{\mathbf{s}}_i^T \mathbf{M}_{2,i},\ \mathbf{G}_{\mathbf{r},\mathbf{i}}(\mathbf{q}) = \bar{\mathbf{s}}_i^T \mathbf{f}_{g,i}$
24	end
25	$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = [\mathbf{k}_{fb}^T \ \mathbf{C}_{\mathbf{r}}(\mathbf{q},\dot{\mathbf{q}})^T]^T$
26	$\mathbf{G}(\mathbf{q}) = [\mathbf{f}_{fb,g}^T \ \mathbf{G}_{\mathbf{r}}(\mathbf{q})^T]^T$

APPENDIX B

RECURSIVE INVERSION ALGORITHMS

This appendix describes a set of recursive inversion algorithms. Two remarks regarding the explained algorithms are stated as follows:

Remark.1 The symbol % is used for comments, that are used to explain some notations without affecting the code flow.

Remark.2 The square brackets are used to denote the rows and columns for the matrices elements as follows:

- If $a = \mathbf{A}[r][c]$, then a is scalar whose value equals to the element of $\mathbf{A}$ in the row r and column c.
- If $\mathbf{a} = \mathbf{A}[r_1 : r_2][c]$, then $\mathbf{a}$ is a column vector equal to sub-vector of $\mathbf{A}$ located in the rows from r_1 to r_2 and in column c.
- If $\mathbf{a} = \mathbf{A}[r][c_1 : c_2]$, then $\mathbf{a}$ is a row vector equal to sub-vector of $\mathbf{A}$ located in the columns from c_1 to c_2 and in row r.
- If $\mathbf{B} = \mathbf{A}[r_1 : r_2][c_1 : c_2]$, then matrix $\mathbf{B}$ equal to a sub-matrix of $\mathbf{A}$ whose column indexes from c_1 to c_2 and row indexes from r_1 to r_2.

1. **Recursive Inversion:** Based on Sherman-Morrison-Woodbury formula, for an arbitrary symmetric nonsingular matrix $\mathbf{A}_k$, vector $\mathbf{b}$ of suitable dimension and scaler β, if the matrix is update is as $\mathbf{A}_{k+1} = \mathbf{A}_k + \beta \mathbf{b}\mathbf{b}^\intercal$. The inversion of the updated matrix

can be handled as follows:

$$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1} - \alpha\beta\mathbf{h}\mathbf{h}^\intercal \tag{B.1}$$

$$\mathbf{h} = \mathbf{A}_k^{-1}\mathbf{b} \tag{B.2}$$

$$\alpha = \frac{1}{1+\beta\mathbf{b}^\intercal\mathbf{h}}. \tag{B.3}$$

The same formula can be applied in the case of obtaining two symmetric matrices sum inversion. Let matrix $\mathbf{B} = \mathbf{L}\mathbf{L}^\intercal$, and $\mathbf{L}$ is an arbitrary nonsingular square matrix, the inversion of $\mathbf{A}_{k+1} = (\mathbf{A}_k + \mathbf{B})$, can be obtained recursively in terms of $\mathbf{A}_k^{-1}$, using algorithm 1.

Algorithm 1 *recI*$(\mathbf{A}_k^{-1}, \mathbf{L})$

$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1}$
for i=1:c % c is size of $\mathbf{A}_k$
  $\boldsymbol{h} = \mathbf{A}_{k+1}^{-1}\boldsymbol{l}_i$ % $\boldsymbol{l}_i$ is the column *i* in matrix $\mathbf{L}$
  $\alpha = 1/(1+\boldsymbol{l}_i^\intercal\boldsymbol{h})$
  $\mathbf{A}_{k+1}^{-1} = \mathbf{A}_{k+1}^{-1} - \alpha\boldsymbol{h}\boldsymbol{h}^\intercal$
end
return $(\mathbf{A}_{k+1}^{-1})$

2. **Recursive Order Increment:** Consider an arbitrary symmetric nonsingular matrix $\mathbf{A}_k$ with given $\mathbf{A}_k^{-1}$. Using the same formula discussed in [89], if matrix $\mathbf{A}_k$ is updated by a suitable vector $\mathbf{b}$, such that:

$$\mathbf{A}_{k+1} = \begin{bmatrix} \mathbf{A}_k & \mathbf{b} \\ \mathbf{b}^\intercal & c \end{bmatrix} \tag{B.4}$$

the inversion of the updated matrix is given by

$$\mathbf{A}_{k+1}^{-1} = \begin{bmatrix} \mathbf{A}_k^{-1} + \alpha\mathbf{h}\mathbf{h}^\intercal & -\alpha\mathbf{h} \\ -\alpha\mathbf{h}^\intercal & \alpha \end{bmatrix} \tag{B.5}$$

$$\mathbf{h} = \mathbf{A}_k^{-1}\mathbf{b} \tag{B.6}$$

$$\alpha = 1/(c - \mathbf{b}^\intercal\mathbf{h}) \tag{B.7}$$

The upper formula can be used to obtain matrix inversion recursively, in case of matrix $\mathbf{A}_k$ size changed from n to m, for $m-n>1$. In case of the updated matrix is given as:

$$\mathbf{A}_{k+1}=\begin{bmatrix}\mathbf{A}_k & \mathbf{B}[1:n][n+1:m]\\ \mathbf{B}[n+1:m][1:n] & \mathbf{B}[n+1:m][n+1:m]\end{bmatrix}, \tag{B.8}$$

for $\mathbf{B}=\mathbf{L}\mathbf{L}^{\intercal}$, and $\mathbf{L}$ is an arbitrary square nonsingular matrix with proper dimension, the inversion of the updated matrix is handled via algorithm 2.

Algorithm 2 $Add(\mathbf{A}_k^{-1},\mathbf{B})$

$\mathbf{A}_{k+1}^{-1}=\mathbf{A}_k^{-1}$
$m=dim(\mathbf{B})$, $n=dim(\mathbf{A_k})$ %Function $dim(\bullet)$ returns the order of the input matrix.
$d=m-n$
for i=1:d
  $\mathbf{b}=\mathbf{B}[1:n+i-1][n+i]$
  $\mathbf{h}=\mathbf{A}_{k+1}^{-1}\mathbf{b}$
  $c=\mathbf{B}[n+i][n+i]$
  $\alpha=1/(c-\mathbf{b}^{\intercal}\mathbf{h})$
  $\mathbf{A}_{k+1}^{-1}=\begin{bmatrix}\mathbf{A}_k^{-1}+\alpha\mathbf{h}\mathbf{h}^{\intercal} & -\alpha\mathbf{h}\\ -\alpha\mathbf{h}^{\intercal} & \alpha\end{bmatrix}$
end
return $(\mathbf{A}_{k+1}^{-1})$

3. **Recursive Order Decrement:** Consider a nonsingular square matrix $\mathbf{A}_k$ with the following form,

$$\mathbf{A}_k=\begin{bmatrix}\mathbf{A}_{k+1} & \mathbf{b}\\ \mathbf{c} & d\end{bmatrix}, \tag{B.9}$$

where, $\mathbf{b}$ and $\mathbf{c}$ are two vectors of suitable dimensions,and d is a scalar value. If the inverse of $\mathbf{A}_k$ is given by:

$$\mathbf{A}_k^{-1}=\begin{bmatrix}\mathbf{A}' & \mathbf{b}'\\ \mathbf{c}' & d'\end{bmatrix}, \tag{B.10}$$

then the inversion of $\mathbf{A}_{k+1}$ is obtained in terms of $(\mathbf{A}_k)^{-1}$ elements is done using the following proof:

$$\mathbf{A}_k\mathbf{A}_k^{-1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{B.11}$$

$$\mathbf{A}_k\mathbf{A}_k^{-1} = \begin{bmatrix} \mathbf{A}_{k+1}\mathbf{A}' + \mathbf{b}\mathbf{c}' & \mathbf{A}_{k+1}\mathbf{b}' + \mathbf{b}d' \\ \mathbf{c}\mathbf{A}' + d\mathbf{c}' & \mathbf{c}\mathbf{b}' + dd' \end{bmatrix}. \tag{B.12}$$

From the upper right equation of (B.12) the following equations can be driven:

$$\mathbf{b}\mathbf{c}' = (\mathbf{b}d')(d'^{-1}\mathbf{c}') \tag{B.13}$$

$$(\mathbf{A}_{k+1}\mathbf{b}' + \mathbf{b}d' - \mathbf{A}_{k+1}\mathbf{b}')((d')^{-1}\mathbf{c}') = -\mathbf{A}_{k+1}\mathbf{b}'d'^{-1}\mathbf{c}' \tag{B.14}$$

$$\mathbf{b}\mathbf{c}' = -\mathbf{A}_{k+1}\mathbf{b}'d'^{-1}\mathbf{c}' \tag{B.15}$$

From the upper left equation of (B.12), and (B.15), the following equations can be obtained:

$$\mathbf{A}_{k+1}^{-1} = (\mathbf{A}' - \mathbf{b}'d'^{-1}\mathbf{c}') \tag{B.16}$$

Equation (B.16) can be used in a recursive form if it is required to eliminate n rows and columns as shown in algorithm 3.

Algorithm 3 $RM(\mathbf{A}_k^{-1}, n)$

$m = dim(\mathbf{A}_k^{-1})$
for i=1:n
  $\mathbf{b}' = \mathbf{A}_k^{-1}[1 : m-i][m-i+1]$
  $\mathbf{c}' = \mathbf{A}_k^{-1}[m-i+1][1 : m-i]$
  $d' = \mathbf{A}_k^{-1}[m-i+1][m-i+1]$
  $\mathbf{A}_k^{-1} = \mathbf{A}_k^{-1}[1 : m-i][1 : m-i]$
  $\mathbf{A}_k^{-1} = \mathbf{A}_k^{-1} - (1/d')\mathbf{b}'\mathbf{c}'$
end
$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1}$
return $(\mathbf{A}_{k+1}^{-1})$

4. **Kenneth Miller Formula:** For two arbitrary square matrices **A** and **H** with the same dimensions. Let **A** be a nonsingular matrix and **H** of rank one. The inversion of the

sum of the two matrices is handled as follows:

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \mathbf{H}, \tag{B.17}$$

$$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1} - \frac{1}{1+g}\mathbf{A}_k^{-1}\mathbf{H}\mathbf{A}_k^{-1}, \tag{B.18}$$

$$g = \langle \mathbf{H}, \mathbf{A}_k \rangle, \tag{B.19}$$

[91]. In case of matrix $\mathbf{H}$ is full rank with order $n > 1$, then equation (B.18) is applied recursively n times, as shown in algorithm 4.

Algorithm 4 $KM(\mathbf{A}_k^{-1}, \mathbf{H})$

for i=1:n
  $\mathbf{h} = \mathbf{H}[1:n][i]$
  $\mathbf{a} = \mathbf{A}_k^{-1}[i][1:n]$
  $g = \mathbf{ah}$
  $\mathbf{A}_k^{-1} = \mathbf{A}_k^{-1} - \frac{1}{1+g}\mathbf{A}_k^{-1}\mathbf{ha}$
end
$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1}$
return $(\mathbf{A}_{k+1}^{-1})$

APPENDIX C

SYMMETRIC KRONECKER PRODUCT

Definition.1 For a given arbitrary symmetric matrix $\mathbf{A}_s \in \mathbb{S}^n$ and vector $\mathbf{s}_v \in \mathbb{R}^{\frac{n(n+1)}{2}}$ is defined using $svec(\bullet)$ function as follows:

$$\mathbf{s}_v = [a_{11}, \sqrt{2}a_{21}, ..., \sqrt{2}a_{n1}, s_{22}, \sqrt{2}a_{32}, ..., \sqrt{2}a_{n2}, ..., a_{nn}]^\intercal \tag{C.1}$$

where, a_{rc} indicates an element in matrix $\mathbf{A}_s$ at the r^{th} row and c^{th} column. Another defined function, $\mathbf{A}_s = smat(\mathbf{s}_v)$, can be used as an inversion of $svec(\bullet)$.

Definition. 2 Symmetric Kronecker Product can be defined as a linear operator on symmetric matrix space. For a given two square matrices $\mathbf{M}_m, \mathbf{N}_m \in \mathbb{R}^{n\times n}$ and a symmetric matrix $\mathbf{A}_s \in \mathbb{S}^n$, Symmetric Kronecker Product can be defined using following mapping:

$$(\mathbf{M}_m \otimes_s \mathbf{N}_m) svec(\mathbf{A}_s) = svec(\frac{1}{2}(\mathbf{N}_m \mathbf{A}_s \mathbf{M}_m^\intercal + \mathbf{M}_m \mathbf{A}_s \mathbf{N}_m^\intercal)). \tag{C.2}$$

APPENDIX D

DOUBLE FOUR BAR MECHANISM KINEMATIC ANALYSIS

The section describes the kinematic relationship between the ankle joint motor angular displacement θ_m, and the leg knee and ankle joints which are represented, respectively, by θ_k and θ_a. The kinematic algorithm is solved analytically, hence the mechanism link lengths could be obtained to avoid singular positions and to maximize the range of motion. Furthermore, the algorithm is used to obtain the required motor angular displacement to generate the desired ankle joint based on the previously solved floating base kinematics. The algorithm is verified using MSC.ADAMS and MATLAB/SIMULINK co-simulation; see Fig. 70-a.

The Co-Ex ankle mechanism schematic is shown in Fig. 70-b. The mechanism members are classified as follows. The straight members are represented by the blue lines, and the twisted member is represented by the green line. The red link shows the leg shank, and the dotted line expresses the thigh. The nodes denote the revolute joints, while node 4 shows two coincided revolute joints at 4 connecting links 543, 74, and 14. The link lengths are symbolized by l_i, where i is the link number.

The mechanism shows a kinematic chain connecting θ_m and θ_a. It can be divided into two kinematic loops. The first loop is represented by L_1 and nodes (12341), while the second loop is represented by L_2 and nodes (45674). L_2 can be divided into two paths. The first path is denoted by 476 and used to obtain the Cartesian coordinates of node 6 relative to 4 using θ_a and θ_k.

$$^4x_6 = l_7 cos(\theta_k) + l_6 cos(\theta_k + \theta_a) \tag{D.1}$$

$$^4y_6 = l_7 sin(\theta_k) + l_6 sin(\theta_k + \theta_a) \tag{D.2}$$

The second path in L_2 is represented by 456 and used to compute the orientation of the first

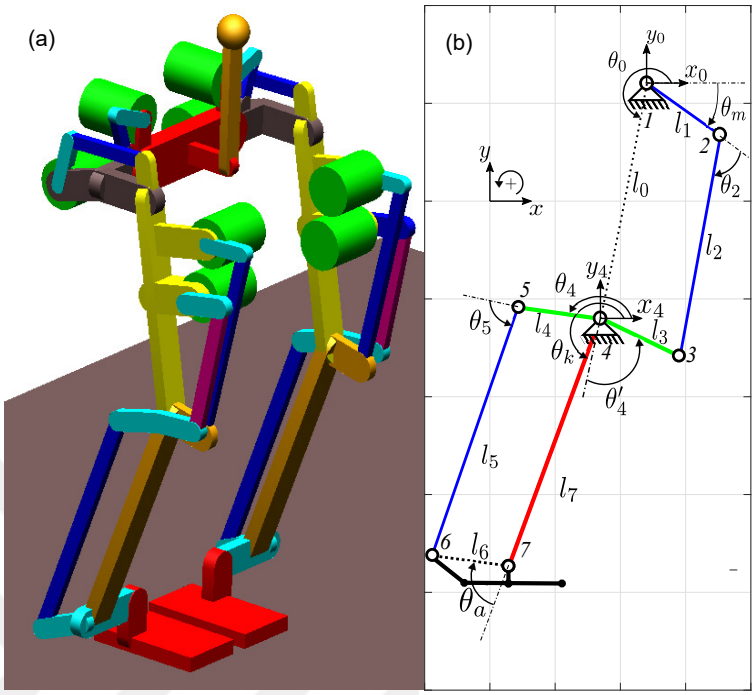

Figure 70: Co-Ex leg mechanism (a) Leg mechanism model as constructed in MSC.ADAMS. (b) Co-Ex ankle mechanism schematic model.

portion of the twisted link 543 denoted by 45. Using cosine rule, the angle between link 56 and 45 denoted by θ_5 is obtained as

$$D_5 = \frac{{}^4x_6^2 + {}^4y_6^2 - l_5^2 - l_4^2}{2l_5l_4} \tag{D.3}$$

$$\theta_5 = atan2(\sqrt{1 - D_5^2}, D_5). \tag{D.4}$$

The orientation of link 45 is given as

$$\theta_4 = atan2({}^4y_6, {}^4x_6) - atan2(l_5 sin(\theta_5), l_4 + l_5 cos(\theta_5)). \tag{D.5}$$

The orientation of the second portion of the twisted link, denoted by 43, is related to θ_4 using constant angular displacement θ_g, such that $\theta'_4 = \theta_4 + \theta_g$. The angular displacement θ_g depends on the twisted link angle and the Cartesian displacement between node 1 and 4 as shown in Fig. 70-b. Using loop L_1, the path 143 is used to compute the Cartesian

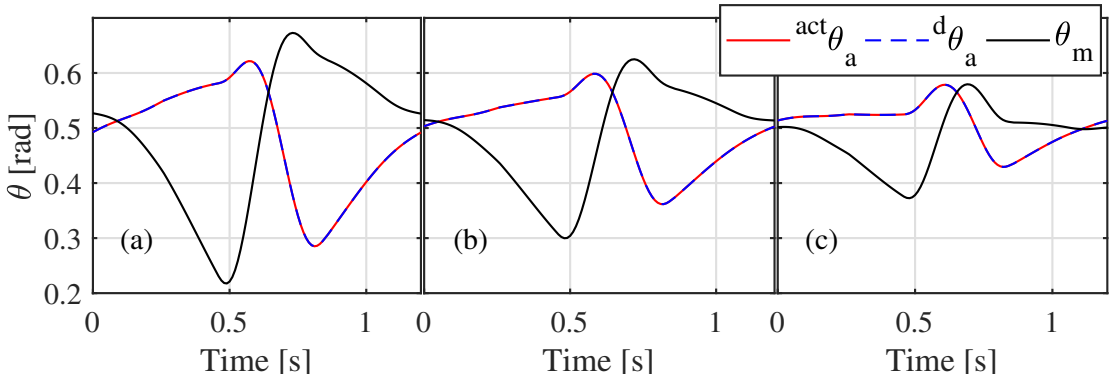

Figure 71: The angular displacements of the ankle motor θ_m, desired ankle ${}^{d}\theta_a$, and actual ankle ${}^{act}\theta_a$ for one single step. The average walking speed in figure (a) is 300 mm/s, in (b) is 200 mm/s, and in (c) is 100 mm/s.

coordinates of node 3 relative to 1.

$$ {}^{1}x_3 = l_0 cos(\theta_0) + l_3 cos(\theta_0 + \theta_4') \tag{D.6} $$

$$ {}^{1}y_3 = l_0 sin(\theta_0) + l_3 sin(\theta_0 + \theta_4') \tag{D.7} $$

From path 123 in $L1$, the angle between link 12 and 23 which is denoted by θ_2 can be obtained using the cosine rule such that:

$$ D_2 = \frac{{}^{1}x_3^2 + {}^{1}y_3^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{D.8} $$

$$ \theta_2 = atan2(\sqrt{1 - D_2^2}, D_2). \tag{D.9} $$

The ankle motor angular displacement is expressed as below:

$$ \theta_m = atan2({}^{1}y_3, {}^{1}x_3) - atan2(l_2 sin(\theta_2), l_1 + l_2 cos(\theta_2)). \tag{D.10} $$

The link lengths are chosen such that $|D_2| < 1$, $|D_5| < 1$, and to avoid interference between links and robot bodies. The kinematic algorithm is verified for various walking patterns. Fig. 71 shows the motor signal θ_m as computed by (D.10) and used to motorize the mechanism in MSC.ADAMS model, the predefined desired ankle angle ${}^{d}\theta_a$, and the ankle angular displacement ${}^{act}\theta_a$ as measured from MSC.ADAMS. Fig. 71 shows the joint signals for one single step and three different motion patterns. Fig. 71-a shows the angular displacement signals for an average walking speed of 0.3 m/s, while Fig. 71-b and 71-c shows the trajectories for 0.2 m/s, and 0.1 m/s, respectively. The average error is ${}^{d}\theta_a - {}^{act}\theta_a \approx 0$ rad. Therefore, it is concluded that the kinematic solution can handle the joints displacements mapping and links lengths sizing properly.

REFERENCES

[1] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer US, 2008.

[2] M. Wyndaele and J.-J. Wyndaele, "Incidence, prevalence and epidemiology of spinal cord injury: what learns a worldwide literature survey?," *Spinal cord*, vol. 44, no. 9, pp. 523–529, 2006.

[3] D. M. Dryden, L. D. Saunders, B. H. Rowe, L. A. May, N. Yiannakoulias, L. W. Svenson, D. P. Schopflocher, and D. C. Voaklander, "Utilization of health services following spinal cord injury: a 6-year follow-up study," *Spinal Cord*, vol. 42, pp. 513–525, July 2004.

[4] A. J. Young and D. P. Ferris, "State of the art and future directions for lower limb robotic exoskeletons," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, pp. 171–182, Feb. 2017.

[5] B. Ugurlu, H. Oshima, and T. Narikiyo, "Lower body exoskeleton-supported compliant bipedal walking for paraplegics: How to reduce upper body effort?," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1354–1360, 2014.

[6] L. Brenner, "Exploring the psychosocial impact of ekso bionics technology," *Archives of Physical Medicine and Rehabilitation*, vol. 97, no. 10, p. e113, 2016.

[7] A. E. Palermo, J. L. Maher, C. B. Baunsgaard, and M. S. Nash, "Clinician-focused overview of bionic exoskeleton use after spinal cord injury," *Topics in spinal cord injury rehabilitation*, vol. 23, no. 3, pp. 234–244, 2017.

[8] B. I, H. K, T. D, and van Middendorp J. J., "Lower-limb exoskeletons for individuals with chronic spinal cord injury: findings from a feasibility study," in *Clinical Rehabilitation*, vol. 30, pp. 73–84, 2016.

[9] M. Nikkhah, H. Ashrafiuon, and F. Fahimi, "Robust control of underactuated bipeds using sliding modes," *Robotica*, vol. 25, pp. 367–374, May 2007.

[10] C. Chevallereau, J. Grizzle, and C.-L. Shih, "Asymptotically stable walking of a five-link underactuated 3-d bipedal robot," *IEEE Transactions on Robotics*, vol. 25, pp. 37–50, Feb. 2009.

[11] K. A. Hamed and J. W. Grizzle, "Event-based stabilization of periodic orbits for underactuated 3-d bipedal robots with left-right symmetry," *IEEE Transactions on Robotics*, vol. 30, pp. 365–381, Apr. 2014.

[12] M. Berkowitz, P. O'Leary, D. Kruse, and C. Harvey, *Spinal cord injury: an analysis of medical and social costs*. Demos Medical Publishing, 1998.

[13] D. E. Garland, C. A. Stewart, R. H. Adkins, S. S. Hu, C. Rosen, F. J. Liotta, and D. A. Weinstein, "Osteoporosis after spinal cord injury," *Journal of orthopaedic research*, vol. 10, no. 3, pp. 371–378, 1992.

[14] M. J. Castro, D. F. Apple Jr, R. S. Staron, G. E. Campos, and G. A. Dudley, "Influence of complete spinal cord injury on skeletal muscle within 6 mo of injury," *Journal of Applied Physiology*, vol. 86, no. 1, pp. 350–358, 1999.

[15] M. A. Regan, R. W. Teasell, D. L. Wolfe, D. Keast, W. B. Mortenson, J.-A. L. Aubut, S. C. I. R. E. R. Team, *et al.*, "A systematic review of therapeutic interventions for pressure ulcers after spinal cord injury," *Archives of physical medicine and rehabilitation*, vol. 90, no. 2, pp. 213–231, 2009.

[16] M. K. Vukobratovic, "When were active exoskeletons actually born?," *International Journal of Humanoid Robotics*, vol. 4, no. 03, pp. 459–486, 2007.

[17] G. Zeilig, H. Weingarden, M. Zwecker, I. Dudkiewicz, A. Bloch, and A. Esquenazi, "Safety and tolerance of the ReWalk exoskeleton suit for ambulation by people with complete spinal cord injury: A pilot study," *The Journal of Spinal Cord Medicine*, vol. 35, pp. 96–101, 03 2012.

[18] E. Read, C. Woolsey, C. A. McGibbon, and C. O'Connell, "Physiotherapists' experiences using the ekso bionic exoskeleton with patients in a neurological rehabilitation hospital: A qualitative study," *Rehabilitation Research and Practice*, vol. 2020, pp. 1–8, Jan. 2020.

[19] "https://exoskeletonreport.com/product/indego/,"

[20] "https://www.indego.com/parkerimages/promosite/indego/united

[21] G. S. Sawicki and D. P. Ferris, "Powered ankle exoskeletons reveal the metabolic cost of plantar flexor mechanical work during walking with longer steps at constant step frequency," *Journal of Experimental Biology*, vol. 212, no. 1, pp. 21–31, 2009.

[22] R. Griffin, T. Cobb, T. Craig, M. Daniel, N. van Dijk, J. Gines, K. Kramer, S. Shah, O. Siebinga, J. Smith, *et al.*, "Stepping forward with exoskeletons: Team ihmc? s design and approach in the 2016 cybathlon," *IEEE Robotics & Automation Magazine*, vol. 24, no. 4, pp. 66–74, 2017.

[23] C. Mummolo, W. Z. Peng, S. Agarwal, R. Griffin, P. D. Neuhaus, and J. H. Kim, "Stability of mina v2 for robot-assisted balance and locomotion," *Frontiers in neurorobotics*, vol. 12, p. 62, 2018.

[24] M. M. van der Krogt, S. L. Delp, and M. H. Schwartz, "How robust is human gait to muscle weakness?," *Gait & posture*, vol. 36, no. 1, pp. 113–119, 2012.

[25] C. F. GEISER, K. M. O'CONNOR, and J. E. EARL, "Effects of isolated hip abductor fatigue on frontal plane knee mechanics," *Medicine & Science in Sports & Exercise*, vol. 42, pp. 535–545, Mar. 2010.

[26] S. Wang, L. Wang, C. Meijneke, E. van Asseldonk, T. Hoellinger, G. Cheron, Y. Ivanenko, V. L. Scaleia, F. Sylos-Labini, M. Molinari, F. Tamburella, I. Pisotta, F. Thorsteinsson, M. Ilzkovitz, J. Gancet, Y. Nevatia, R. Hauffe, F. Zanow, and H. van der Kooij, "Design and control of the MINDWALKER exoskeleton," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, pp. 277–286, Mar. 2015.

[27] C. Meijneke, G. van Oort, V. Sluiter, E. van Asseldonk, N. L. Tagliamonte, F. Tamburella, I. Pisotta, M. Masciullo, M. Arquilla, M. Molinari, A. R. Wu, F. Dzeladini, A. J. Ijspeert, and H. van der Kooij, "Symbitron exoskeleton: Design, control, and evaluation of a modular exoskeleton for incomplete and complete spinal cord injured individuals," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pp. 1–1, 2021.

[28] S.-H. Hyon, J. Morimoto, T. Matsubara, T. Noda, and M. Kawato, "Xor: Hybrid drive exoskeleton robot that can balance," in *2011 IEEE/RSJ International conference on intelligent robots and systems*, pp. 3975–3981, IEEE, 2011.

[29] B. Ugurlu, C. Doppmann, M. Hamaya, P. Forni, T. Teramae, T. Noda, and J. Morimoto, "Variable ankle stiffness improves balance control: Experiments on a bipedal exoskeleton," *IEEE/ASME Transactions on Mechatronics*, vol. 21, pp. 79–87, Feb. 2016.

[30] G. Barbareschi, R. Richards, M. Thornton, T. Carlson, and C. Holloway, "Statically vs dynamically balanced gait: Analysis of a robotic exoskeleton compared with a human," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 6728–6731, IEEE, Aug. 2015.

[31] K. Thakkar, V. Paredes, and A. Hereid, "Adaptive feedback regulator for powered lower-limb exoskeleton under model uncertainty," *arXiv preprint arXiv:2104.11775*, 2021.

[32] T. Gurriet, S. Finet, G. Boeris, A. Duburcq, A. Hereid, O. Harib, M. Masselin, J. Grizzle, and A. D. Ames, "Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2804–2811, 2018.

[33] R. J. Farris, H. A. Quintero, and M. Goldfarb, "Preliminary evaluation of a powered lower limb orthosis to aid walking in paraplegic individuals," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, pp. 652–659, Dec. 2011.

[34] L. Lonini, N. Shawen, K. Scanlan, W. Z. Rymer, K. P. Kording, and A. Jayaraman, "Accelerometry-enabled measurement of walking performance with a robotic exoskeleton: a pilot study," *Journal of NeuroEngineering and Rehabilitation*, vol. 13, no. 1, pp. 1–10, 2016.

[35] O. Harib, A. Hereid, A. Agrawal, T. Gurriet, S. Finet, G. Boeris, A. Duburcq, M. E. Mungai, M. Masselin, A. D. Ames, K. Sreenath, and J. W. Grizzle, "Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking," *IEEE Control Systems Magazine*, vol. 38, pp. 61–87, Dec. 2018.

[36] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *2010 IEEE International Conference on Robotics and Automation*, pp. 3733–3738, 2010.

[37] M. Vukobratović and B. Borovac, "Zero-moment point—thirty five years of its life," *International journal of humanoid robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[38] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, IEEE.

[39] B. Ugurlu, I. Havoutis, C. Semini, K. Kayamori, D. G. Caldwell, and T. Narikiyo, "Pattern generation and compliant feedback control for quadrupedal dynamic trot-walking locomotion: experiments on robocat-1 and hyq," *Autonomous Robots*, vol. 38, no. 4, pp. 415–437, 2015.

[40] M. C. Yildirim, A. T. Kansizoglu, S. Emre, M. Derman, S. Coruk, A. F. Soliman, P. Sendur, and B. Ugurlu, "Co-ex: A torque-controllable lower body exoskeleton for dependable human-robot co-existence," in *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pp. 605–610, IEEE, June 2019.

[41] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302, IEEE, Nov. 2014.

[42] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *International Journal of Humanoid Robotics*, vol. 13, p. 1550039, Mar. 2016.

[43] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal, "Inverse kinematics with floating base and constraints for full body humanoid robot control," in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pp. 22–27, IEEE, 2008.

[44] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3103–3109, IEEE, May 2013.

[45] L. Righetti, J. Buchli, M. Mistry, and S. Schaal, "Control of legged robots with optimal distribution of contact forces," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pp. 318–324, IEEE, Oct. 2011.

[46] B. Ugurlu, H. Oshima, E. Sariyildiz, T. Narikiyo, and J. Babic, "Active compliance control reduces upper body effort in exoskeleton-supported walking," *IEEE Transactions on Human-Machine Systems*, vol. 50, pp. 144–153, Apr. 2020.

[47] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics Robots in Unstructured Environments*, IEEE, 1991.

[48] M. Mistry, J. Nakanishi, and S. Schaal, "Task space control with prioritization for balance and locomotion," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 331–338, IEEE, Oct. 2007.

[49] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, pp. 473–491, Aug. 2015.

[50] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, pp. 1006–1028, May 2014.

[51] B. Henze, A. Dietrich, and C. Ott, "An approach to combine balancing with hierarchical whole-body control for legged humanoid robots," *IEEE Robotics and Automation Letters*, vol. 1, pp. 700–707, July 2016.

[52] A. F. Soliman and B. Ugurlu, "Robust locomotion control of a self-balancing and underactuated bipedal exoskeleton: Task prioritization and feedback control," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5626–5633, 2021.

[53] S. Traversaro, S. Brossette, A. Escande, and F. Nori, "Identification of fully physical consistent inertial parameters using optimization on manifolds," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5446–5451, 2016.

[54] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, "Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 746–768, 1998.

[55] W. Li and J.-J. Slotine, "Indirect adaptive robot control," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 704–709 vol.2, 1988.

[56] B. Ugurlu, E. Sariyildiz, A. T. Kansizoglu, E. C. Ozcinar, and S. Coruk, "Benchmarking torque control strategies for a torsion-based series elastic actuator," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 85–96, 2022.

[57] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[58] W.-H. Ho, T.-Y. Shiang, C.-C. Lee, and S.-Y. Cheng, "Body segment parameters of young chinese men determined with magnetic resonance imaging," *Medicine & Science in Sports & Exercise*, vol. 45, pp. 1759–1766, Sept. 2013.

[59] P. de Leva, "Adjustments to zatsiorsky-seluyanov's segment inertia parameters," *Journal of Biomechanics*, vol. 29, pp. 1223–1230, Sept. 1996.

[60] M. C. Yildirim, P. Sendur, O. Bilgin, B. Gulek, G. G. Yapici, and B. Ugurlu, "An integrated design approach for a series elastic actuator: Stiffness formulation, fatigue analysis, thermal management," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 384–389, IEEE, 2017.

[61] "https://www.jetbrains.com/clion/,"

[62] M. Galassi, F. Rossi, B. Gough, M. Booth, G. Jungman, J. Theiler, and J. Davies, *GNU Scientific Library: Reference Manual*. Network Theory Limited, 2001.

[63] "https://www.kollmorgen.cn/onlinebooks/tbm/mobile/index.html,"

[64] B. Ugurlu, M. Nishimura, K. Hyodo, M. Kawanishi, and T. Narikiyo, "Proof of concept for robot-aided upper limb rehabilitation using disturbance observers," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 1, pp. 110–118, 2014.

[65] P. Baerlocher and R. Boulic, "An inverse kinematics architecture enforcing an arbitrary number of strict priority levels," *The Visual Computer*, vol. 20, no. 6, pp. 402–417, 2004.

[66] A. B. Zoss, H. Kazerooni, and A. Chu, "Biomechanical design of the berkeley lower extremity exoskeleton (bleex)," *IEEE/ASME Transactions on mechatronics*, vol. 11, no. 2, pp. 128–138, 2006.

[67] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," in *2010 IEEE International Conference on Robotics and Automation*, pp. 2406–3412, IEEE, May 2010.

[68] N. Nazir, H. Izu, S. Nakaura, and M. Sampei, "An analysis of zmp control problem of humanoid robot with compliances in sole of the foot," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 337–342, 2005.

[69] S. C. Chapra and R. P. Canale, *Numerical methods for engineers*. New York, NY: McGraw-Hill Professional, 7 ed., Jan. 2014.

[70] S. Ploen, "A skew-symmetric form of the recursive newton-euler algorithm for the control of multibody systems," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 6, pp. 3770–3773 vol.6, 1999.

[71] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *The International Journal of Robotics Research*, vol. 5, pp. 101–119, Sept. 1986.

[72] C. H. An, C. G. Atkeson, and J. Hollerbach, *Model-Based Control of a Robot Manipulator*. MIT Press, 08 1988.

[73] E. D. W. Khalil, *Modeling, Identification and Control of Robots*. Elsevier Science, 2004.

[74] G. Venture, P.-J. Ripert, W. Khalil, M. Gautier, and P. Bodson, "Modeling and identification of passenger car dynamics using robotics formalism," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 349–359, 2006.

[75] J.-A. Ting, M. N. Mistry, J. Peters, S. Schaal, and J. Nakanishi, "A bayesian approach to nonlinear parameter identification for rigid body dynamics.," in *Robotics: Science and systems*, pp. 32–39, Philadelphia, USA, 2006.

[76] N. Y. Ayusawa K, Venture G, "Identifiability and identification of inertial parameters using the underactuated base-link dynamics for legged multibody systems," *The International Journal of Robotics Research*, vol. 33, no. 3, pp. 446–468, 2014.

[77] R. Bellman and K. Åström, "On structural identifiability," *Mathematical Biosciences*, vol. 7, pp. 329–339, 1970.

[78] M. Gautier and W. Khalil, "[1991] proceedings of the 30th ieee conference on decision and control," pp. 494–499 vol.1, 1991.

[79] J. Swevers, C. Ganseman, D. Tukel, J. de Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 730–740, 1997.

[80] F. Bennis, W. Khalil, and M. Gautier, "Calculation of the base inertial parameters of closed-loops robots," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pp. 370–375 vol.1, 1992.

[81] H. Mayeda, K. Yoshida, and K. Osuka, "Base parameters of manipulator dynamic models," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 1367–1372 vol.3, 1988.

[82] B. F. Khalil W, "Symbolic calculation of the base inertial parameters of closed-loop robots," *The International Journal of Robotics Research*, vol. 14, no. 2, pp. 112–128, 1995.

[83] H. Mayeda, K. Yoshida, and K. Osuka, "Base parameters of manipulator dynamic models," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 312–321, 1990.

[84] M. Gautier, "Numerical calculation of the base inertial parameters of robots," in *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 1020–1025 vol.2, 1990.

[85] P. M. Wensing, S. Kim, and J.-J. E. Slotine, "Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 60–67, 2018.

[86] K. Ayusawa, Y. Nakamura, and G. Venture, "Optimal estimation of human body segments dynamics using realtime visual feedback," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1627–1632, 2009.

[87] K. Ayusawa, G. Venture, and Y. Nakamura, "Real-time implementation of physically consistent identification of human body segments," in *2011 IEEE International Conference on Robotics and Automation*, pp. 6282–6287, 2011.

[88] J. Gondzio, "Interior point methods 25 years later," *European Journal of Operational Research*, vol. 218, no. 3, pp. 587–601, 2012.

[89] K. Engel and S. Engel, "Recursive least squares with linear inequality constraints," *Optimization and Engineering*, vol. 16, no. 1, p. 1–26, 2015.

[90] K. Yoshida, K. Osuka, H. Mayeda, and T. Ono, "When is the set of base parameter values physically impossible?," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, vol. 1, pp. 335–342 vol.1, 1994.

[91] K. S. Miller, "On the inverse of the sum of matrices," *Mathematics Magazine*, vol. 54, no. 2, pp. 67–72, 1981.

VITA

Ahmed Fahmy Soliman is an Egyptian researcher. He received his B.Sc. in Mechatronics engineering from Helwan University, Cairo, Egypt, in 2009. He received his M.Sc. in Electrical engineering from Cairo University, Cairo, Egypt, in 2015. He joined the Biomechatronics Lab Research Group as a Research Assistant in 2017, working toward his doctorate under the supervision of Asst. Prof. Barkan Uğurlu at Özyeğin University, Istanbul, Turkey. His current research interests include legged robot kinematics, dynamics analysis and design, assistive devices, robot dynamics and control, compliant manipulation, human-robot interaction, path and trajectory planning, biomechanics, mechatronics systems design and development, control algorithms, and artificial intelligence.

Ahmed Fahmy Soliman serves as a reviewer for IEEE/ASME Transactions on Mechatronics and IEEE Robotics and Automation Letters. He periodically serves as a reviewer for IEEE RAS-related conferences, mainly IEEE Int. Conf. on Robotics and Automation (ICRA), IEEE Int. Conf. on Intelligent Robots and Systems (IROS), IEEE Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob), IEEE Int. Conf. on Rehabilitation Robotics (ICORR), and IEEE Int. Conf. on Humanoid Robots.